

Juniper Networks® Steel-Belted Radius® Carrier

Administration and Configuration Guide

Published
2021-11-29

Release
8.6.0

Juniper Networks, Inc.
 1133 Innovation Way
 Sunnyvale, California 94089
 USA
 408-745-2000
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. and/or its affiliates in the United States and other countries. All other trademarks may be property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Products made or sold by Juniper Networks or components thereof might be covered by one or more of the following patents that are owned by or licensed to Juniper Networks: U.S. Patent Nos. 5,473,599, 5,905,725, 5,909,440, 6,192,051, 6,333,650, 6,359,479, 6,406,312, 6,429,706, 6,459,579, 6,493,347, 6,538,518, 6,538,899, 6,552,918, 6,567,902, 6,578,186, and 6,590,785.

Ulticom, Signalware, Programmable Network, Ultimate Call Control, and Nexworx are registered trademarks of Ulticom, Inc. Kineto and the Kineto Logo are registered trademarks of Kineto Wireless, Inc. Software Advancing Communications and SignalCare are trademarks and service marks of Ulticom, Inc. CORBA (Common Object Request Broker Architecture) is a registered trademark of the Object Management Group (OMG). Raima, Raima Database Manager, and Raima Object Manager are trademarks of Raima, Inc. Sun, Sun Microsystems, the Sun logo, Java, Solaris, MySQL, and all trademarks and logos that contain Sun, Solaris, MySQL, or Java are trademarks or registered trademarks of Oracle America, Inc. in the United States and other countries. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners. All specifications are subject to change without notice.

Contains software copyright 2000–2014 by Oracle America, Inc., distributed under license.

Steel-Belted Radius uses Thrift, licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License.

You may obtain a copy of the license at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

Steel-Belted Radius uses Xerces XML DOM, from the Apache Group. It has the following terms.

The Apache Software license, Version 1.1

Copyright © 1999-2003 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).” Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.
4. The names “Xerces” and “Apache Software Foundation” must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.
5. Products derived from this software may not be called “Apache”, nor may “Apache” appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation and was originally based on software copyright © 1999, International Business Machines, Inc., <http://www.ibm.com>. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Steel-Belted Radius uses the LDAP v2 Server from the University of Michigan. It has the following terms.

Copyright © 1991 Regents of the University of Michigan. All rights reserved.

Redistribution and use in source and binary forms are permitted provided that this notice is preserved and that due credit is given to the University of Michigan at Ann Arbor. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. This software is provided “as is” without express or implied warranty.

Portions of this software copyright 2003-2009 Lev Walkin <vlm@lionet.info> All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Portions of this software copyright 1989, 1991, 1992 by Carnegie Mellon University

SBR includes NetSNMP under the following licenses: Derivative Work-1996, 1998-2009 Copyright 1996, 1998-2009. The Regents of the University of California All Rights Reserved. Permission to use, copy, modify and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of CMU and The Regents of the University of California not be used in advertising or publicity pertaining to distribution of the software without specific written permission.

CMU AND THE REGENTS OF THE UNIVERSITY OF CALIFORNIA DISCLAIM ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL CMU OR THE REGENTS OF THE UNIVERSITY OF CALIFORNIA BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM THE LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Portions of this software copyright © 2001-2009, Networks Associates Technology, Inc. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the Networks Associates Technology, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR

OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Portions of this software are copyright © 2001–2009, Cambridge Broadband Ltd. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of Cambridge Broadband Ltd. may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Steel-Belted Radius uses Jaxen, a “Java XPath Engine” from The Werken Company under the following license:

Copyright 2003 © The Werken Company. All Rights Reserved.

Redistribution and use of this software and associated documentation (“Software”), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name “jaxen” must not be used to endorse or promote products derived from this Software without prior written permission of The Werken Company. For written permission, please contact bob@werken.com.
4. Products derived from this Software may not be called “jaxen” nor may “jaxen” appear in their names without prior written permission of The Werken Company. “jaxen” is a registered trademark of The Werken Company.
5. Due credit should be given to The Werken Company. (<http://jaxen.werken.com/>).

THIS SOFTWARE IS PROVIDED BY THE WERKEN COMPANY AND CONTRIBUTORS “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE WERKEN COMPANY OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR

CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

HTTPClient package Copyright © 1996–2009 Ronald Tschalär (ronald@innovation.ch)

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. For a copy of the GNU Lesser General Public License, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.

Steel-Belted Radius uses OpenSSL version 1.1.1, which have the following terms:

Copyright ©1998-2018 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment:

"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit.
(<http://www.openssl.org/>)"

4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.

5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.

6. Redistributions of any form whatsoever must retain the following acknowledgment:

"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit
(<http://www.openssl.org/>)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

OpenSSL is also subject to the following terms.

Copyright ©1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com).

The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are aheared to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed.

If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used.

This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

"This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)"

The word 'cryptographic' can be left out if the rouines from the library being used are not cryptographic related :

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement:

"This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,

INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

SBR contains software copyright © 2000–2009 by The Legion Of The Bouncy Castle (<http://www.bouncycastle.org>)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Steel-Belted Radius uses modified source from OpenSolaris (now Oracle) under the CDDL, which can be found at http://hub.opensolaris.org/bin/view/Main/opensolaris_license. Modified source is available. Please refer to the SBR Carrier release notes.

SBR includes Spider Monkey libraries under Mozilla Public License Version 2.0

1. Definitions

1.1. “Contributor” means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.

1.2. “Contributor Version” means the combination of the Contributions of others (if any) used by a Contributor and that particular Contributor’s Contribution.

1.3. “Contribution” means Covered Software of a particular Contributor.

1.4. “Covered Software” means Source Code Form to which the initial Contributor has attached the notice in Exhibit A, the Executable Form of such Source Code Form, and Modifications of such Source Code Form, in each case including portions thereof.

1.5. “Incompatible With Secondary Licenses” means that the initial Contributor has attached the notice described in Exhibit B to the Covered Software; or that the Covered Software was made available under the terms of version 1.1 or earlier of the License, but not also under the terms of a Secondary License.

1.6. “Executable Form” means any form of the work other than Source Code Form.

1.7. “Larger Work” means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software.

1.8. “License” means this document.

1.9. “Licensable” means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently, any and all of the rights conveyed by this License.

1.10. “Modifications” means any of the following: any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered Software; or any new file in Source Code Form that contains any Covered Software.

1.11. “Patent Claims” of a Contributor means any patent claim(s), including without limitation, method, process, and apparatus claims, in any patent Licensable by such Contributor that would be infringed, but for the grant of the License, by the making, using, selling, offering for sale, having made, import, or transfer of either its Contributions or its Contributor Version.

1.12. “Secondary License” means either the GNU General Public License, Version 2.0, the GNU Lesser General Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or any later versions of those licenses.

1.13. “Source Code Form” means the form of the work preferred for making modifications.

1.14. “You” (or “Your”) means an individual or a legal entity exercising rights under this License. For legal entities, “You” includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, “control” means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants and Conditions

2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license: under intellectual property rights (other than patent or trademark) Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.

2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor: for any code that a Contributor has removed from Covered Software; or for infringements caused by: (i) Your and any other third party’s modifications of Covered Software, or (ii) the combination

of its Contributions with other software (except as part of its Contributor Version); or under Patent Claims infringed by Covered Software in the absence of its Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and

You may distribute such Executable Form under the terms of this License, or sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.

3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Termination

5.1. The rights granted under this License will terminate automatically if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You of the non-compliance by some reasonable means prior to 60 days after You have come back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this is the first time You have received notice of non-compliance with this License from such Contributor, and You become compliant prior to 30 days after Your receipt of the notice.

5.2. If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

6. Disclaimer of Warranty

Covered Software is provided under this License on an “as is” basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.

7. Limitation of Liability

Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall any Contributor, or anyone who distributes Covered Software as permitted above, be liable to You for any direct, indirect, special, incidental, or consequential damages of any character including, without limitation, damages for lost profits, loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such party's negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.

8. Litigation

Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party's ability to bring cross-claims or counter-claims.

9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

10. Versions of the License

10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

Exhibit A - Source Code Form License Notice

This Source Code Form is subject to the terms of the Mozilla Public License v.2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

Exhibit B - "Incompatible With Secondary Licenses" Notice

This Source Code Form is "Incompatible With Secondary Licenses", as defined by the Mozilla Public License, v. 2.0.

The "inih" library is distributed under the New BSD license:

Copyright © 2009, Brush Technology
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of Brush Technology nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY BRUSH TECHNOLOGY "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL BRUSH TECHNOLOGY BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contains software copyright 2007-2014, by Sencha, Inc., distributed under license.

Steel-Belted Radius uses Jetty 9 under the Apache License 2.0, You may obtain a copy of the license at <http://www.apache.org/licenses/LICENSE-2.0>

Steel-Belted Radius uses Google Web Toolkit (GWT) under the Apache License 2.0, You may obtain a copy of the license at <http://www.apache.org/licenses/LICENSE-2.0>

Steel-Belted Radius uses Apache HTTP components under the Apache License 2.0, You may obtain a copy of the license at <http://www.apache.org/licenses/LICENSE-2.0>

Steel-Belted Radius uses OpenJDK

GNU General Public License, version 2, with the Classpath Exception

The GNU General Public License (GPL)

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program

itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this

License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

One line to give the program's name and a brief idea of what it does.

Copyright © <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright © year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.
signature of Ty Coon, 1 April 1989

Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

"CLASSPATH" EXCEPTION TO THE GPL

Certain source files distributed by Oracle America and/or its affiliates are subject to the following clarification and special exception to the GPL, but only where Oracle has expressly included in the particular source file's header the words "Oracle designates this particular file as subject to the "Classpath" exception as provided by Oracle in the LICENSE file that accompanied this code."

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

SBR uses Gecko SDK 1.4b

Mozilla Public License Version 2.0

1. Definitions

1.1. "Contributor" means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.

1.2. "Contributor Version" means the combination of the Contributions of others (if any) used by a Contributor and that particular Contributor's Contribution.

1.3. "Contribution" means Covered Software of a particular Contributor.

1.4. "Covered Software" means Source Code Form to which the initial Contributor has attached the notice in Exhibit A, the Executable Form of such Source Code Form, and Modifications of such Source Code Form, in each case including portions thereof.

1.5. "Incompatible With Secondary Licenses" means

(a) that the initial Contributor has attached the notice described in Exhibit B to the Covered Software; or

(b) that the Covered Software was made available under the terms of version 1.1 or earlier of the License, but not also under the terms of a Secondary License.

1.6. "Executable Form" means any form of the work other than Source Code Form.

1.7. "Larger Work" means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software.

1.8. "License" means this document.

1.9. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently, any and all of the rights conveyed by this License.

1.10. "Modifications" means any of the following:

(a) any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered Software; or

(b) any new file in Source Code Form that contains any Covered Software.

1.11. "Patent Claims" of a Contributor means any patent claim(s), including without limitation, method, process, and apparatus claims, in any patent Licensable by such Contributor that would be infringed, but for the grant of the License, by the making, using, selling, offering for sale, having made, import, or transfer of either its Contributions or its Contributor Version.

1.12. "Secondary License" means either the GNU General Public License, Version 2.0, the GNU Lesser General Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or any later versions of those licenses.

1.13. "Source Code Form" means the form of the work preferred for making modifications.

1.14. "You" (or "Your") means an individual or a legal entity exercising rights under this License. For legal entities, "You" includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants and Conditions

2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and

(b) under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.

2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

(a) for any code that a Contributor has removed from Covered Software; or

b) for infringements caused by: (i) Your and any other third party's modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or

(c) under Patent Claims infringed by Covered Software in the absence of its Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

(a) such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and

(b) You may distribute such Executable Form under the terms of this License, or sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.

3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the

maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Termination

5.1. The rights granted under this License will terminate automatically if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You of the non-compliance by some reasonable means prior to 60 days after You have come back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this is the first time You have received notice of non-compliance with this License from such Contributor, and You become compliant prior to 30 days after Your receipt of the notice.

5.2. If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

6. Disclaimer of Warranty

Covered Software is provided under this License on an "as is" basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.

7. Limitation of Liability

Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall any Contributor, or anyone who distributes Covered Software as permitted above, be liable to You for any direct, indirect, special, incidental, or consequential damages of any character including, without limitation, damages for lost profits, loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such party's negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.

8. Litigation

Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party's ability to bring cross-claims or counter-claims.

9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

10. Versions of the License

10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

Exhibit A - Source Code Form License Notice

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

Exhibit B - "Incompatible With Secondary Licenses" Notice

This Source Code Form is "Incompatible With Secondary Licenses", as defined by the Mozilla Public License, v. 2.0.

SBR uses Mozilla LDAP C SDK 5.17

MOZILLA PUBLIC LICENSE

Version 1.1

1. Definitions

1.0.1. "Commercial Use" means distribution or otherwise making the Covered Code available to a third party.

1.1. "Contributor" means each entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. "Covered Code" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.

1.4. "Electronic Distribution Mechanism" means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. "Executable" means Covered Code in any form other than Source Code.

1.6. "Initial Developer" means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.

1.7. "Larger Work" means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. "License" means this document.

1.8.1. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:

A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.

B. Any new file that contains any part of the Original Code or previous Modifications.

1.10. "Original Code" means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.10.1. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.11. "Source Code" means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code

of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant.

The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

(b) under Patents Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

(c) the licenses granted in this Section 2.1(a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices.

2.2. Contributor Grant.

Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) the licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party

modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Application of License.

The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.

Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications.

You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters

(a) Third Party Claims. If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs. If Contributor's Modifications include an application programming interface and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the LEGAL file.

(c) Representations. Contributor represents that, except as disclosed pursuant to Section 3.4(a) above, Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices.

You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients' rights or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions. You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works. You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License.

6.1. New Versions. Netscape Communications Corporation ("Netscape") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions. Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Netscape. No one other than Netscape has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works. If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must (a) rename Your license so that the phrases "Mozilla", "MOZILLAPL", "MOZPL", "Netscape", "MPL", "NPL" or any confusingly similar phrase do not appear in your license (except to note that your license differs from this License) and (b) otherwise make it clear that Your version of the license contains terms which differ from the Mozilla Public License and Netscape Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

8.2. If You initiate litigation by asserting a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that:

(a) such Participant's Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either: (i) agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of Modifications made by such Participant, or (ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant. If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the 60 day notice period specified above.

(b) any software, hardware, or device, other than such Participant's Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant's Contributor Version directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

13. MULTIPLE-LICENSED CODE.

Initial Developer may designate portions of the Covered Code as "Multiple-Licensed". "Multiple-Licensed" means that the Initial Developer permits you to utilize portions of the Covered Code under Your choice of the MPL or the alternative licenses, if any, specified by the Initial Developer in the file described in Exhibit A.

EXHIBIT A -Mozilla Public License.

"The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is _____.

The Initial Developer of the Original Code is _____. Portions created by _____ are Copyright (C) _____. All Rights Reserved.

Contributor(s): _____.

Alternatively, the contents of this file may be used under the terms of the _____ license (the "[_____] License"), in which case the provisions of [_____] License are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of the [_____] License and not to allow others to use your version of this file under the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the [_____] License. If you do not delete the provisions above, a recipient may use your version of this file under either the MPL or the [_____] License."

[NOTE: The text of this Exhibit A may differ slightly from the text of the notices in the Source Code files of the Original Code. You should use the text of this Exhibit A rather than the text found in the Original Code Source Code for Your Modifications.]

Steel-Belted Radius Carrier 8.6.0 Administration and Configuration Guide
Release 8.6.0

Revision History
August 2019—Revision 1
April 2020—Revision 2

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <https://support.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Abbreviated Table of Contents

About This Guide | lxxix

1

Product Overview

Chapter 1 Steel-Belted Radius Carrier Overview | 2

2

Web GUI Overview

Chapter 2 Using Web GUI | 16

3

RADIUS Operations

Chapter 3 RADIUS Basics | 34

Chapter 4 Administering RADIUS Clients and Client Groups | 104

Chapter 5 Administering RADIUS Location Groups | 114

Chapter 6 Administering Users | 120

Chapter 7 Administering Profiles | 144

Chapter 8 Administering Proxy RADIUS | 153

Chapter 9 Administering RADIUS Tunnels | 168

Chapter 10 Administering Address Pools | 182

Chapter 11 Setting Up Administrator Accounts | 195

Chapter 12 Configuring Realm Support | 199

Chapter 13 Setting Up Filters | 217

Chapter 14 Setting Up Authentication Policies | 232

Chapter 15 Setting Up EAP Methods | 253

Chapter 16 Configuring Replication | 316

Chapter 17 3GPP Support | 336

4

Diameter Operations

Chapter 18 Diameter Basics | 339

Chapter 19 Administering the Local Network Element | 349

Chapter 20 Administering Diameter Remote Network Elements | 358

Chapter 21 Administering the Diameter Policy | 376

Chapter 22 Administering Request Routing Rules | 417

Chapter 23 Displaying Diameter Statistics | 429

5

Back-End Authentication and Accounting Methods

Chapter 24 Configuring SQL Authentication | 442

Chapter 25 Configuring SQL Accounting | 458

Chapter 26 Configuring LDAP Authentication | 471

Chapter 27 Signalware SIGTRAN Gateway Support | 484

Chapter 28 Proxy RADIUS Authentication and Accounting | 485

Chapter 29 HSS-Subscriber Database | 486

6

Management Interfaces

Chapter 30 Simple Network Management Protocol | 488

Chapter 31 Using the LDAP Configuration Interface | 497

7

Optional Authentication Modules

Chapter 32 SIM Authentication Module | 536

Chapter 33 Summary of Configuration Tasks for the SIM Authentication Module | 550

Chapter 34 SIM Authentication Module Configuration with a SIGHUP (1) Signal | 553

Chapter 35 Overview of the WiMAX Mobility Module | 554

Chapter 36 Configuring the WiMAX Mobility Module | 580

8

Optional Session State Register (High Availability) Module for a Clustered Environment

Chapter 37 Session State Register Overview | 610

Chapter 38 Session State Register Administration | 628

Chapter 39 Managing User Concurrency with Session State Register | 682

Chapter 40 Managing Concurrency with Attributes in Session State Register | 688

Chapter 41 Introduction to Managing and Controlling Sessions in SBR Carrier | 695

Chapter 42 Overview of the Optional Session Control Module | 700

Chapter 43 Using Web GUI to Manage and Control Sessions | 718

Chapter 44 Using the Command Line Utility to Manage and Control Sessions | 730

Chapter 45 Configuring the deviceModels.xml File | 743

Chapter 46 XML over HTTPS Interface | 760

Chapter 47 Example CoA/DM Configuration | 806

11

Statistics and Reporting

Chapter 48 **Displaying Statistics | 817**

Chapter 49 **Logging and Reporting | 834**

12

Optional Scripting Module

Chapter 50 **Introduction to Scripting | 861**

Chapter 51 **Creating Scripts | 866**

Chapter 52 **Debugging Scripts | 876**

Chapter 53 **Creating LDAP Scripts | 881**

Chapter 54 **Creating Realm Selection Scripts | 895**

Chapter 55 **Creating Attribute Filter Scripts | 908**

Chapter 56 **Working with Data Accessors | 919**

Chapter 57 **Script Reference | 947**

Part 13 Appendixes

Appendix A When and How to Stop and Restart Steel-Belted Radius Carrier | 974

Appendix B Authentication Protocols | 979

Appendix C Importing and Exporting Data | 981

Appendix D Technical Bulletins | 986

Appendix E SIR.sh Script | 998

Appendix F Thread and Flood Control Mechanism | 1002

Table of Contents

About This Guide | lxxix

Objective | lxxix

Audience | lxxix

Documentation Conventions | lxxx

Related Documentation | lxxxii

Obtaining Documentation | lxxxviii

Documentation Feedback | lxxxviii

Requesting Technical Support | lxxxix

1

Product Overview

Steel-Belted Radius Carrier Overview | 2

Introduction to Steel-Belted Radius Carrier | 2

SBR Carrier Core Features | 3

3rd Generation Partnership Project (3GPP) Support | 5

Native Support for Structured Attributes | 5

Adding NAS Location Information to Access-Requests | 6

Support for Additional EAP Authentication Protocols | 6

Statistics and Reporting Capabilities | 6

Management Interfaces | 7

Web GUI | 7

LDAP Configuration Interface (LCI) | 7

Command Line Utility | 7

XML/HTTPS Interface | 7

SNMP | 7

Optional SIM Authentication Module | 8

Optional WiMAX Mobility Module Features | 8

Optional Session Control Module | 10

Optional Scripting Module | 10

Optional Session State Register (High Availability) Module | 11

Optional Concurrency Module | 11

Optional 3GPP AAA Module | 11

Licensing | 12

2

Web GUI Overview

Using Web GUI | 16

Running the Web GUI | 16

Tested Browsers | 19

Navigating in the Web GUI | 20

Web GUI Menus | 20

Home Menu | 20

RADIUS Configuration Menu | 21

Diameter Configuration Menu | 22

Tools Menu | 23

Help Menu | 24

Logout Menu | 24

Web GUI Pages | 24

Adding an Entry | 25

Editing an Entry | 27

Cloning an Entry | 27

Resizing Columns | 28

Changing Column Sequence | 29

Sorting Information | 29

Searching Entries | 29

Hiding or Restoring Columns | 29

Adding License Keys | 29

Displaying Version Information | 31

Closing the Web GUI | 31

3

RADIUS Operations

RADIUS Basics | 34

RADIUS Overview | 34

RADIUS Packets | 36

RADIUS Ports | 37

RADIUS Configuration	38
RADIUS Server Configuration	38
RADIUS Client Configuration	38
Multiple RADIUS Servers	38
Shared Secrets	39
RADIUS Secret	40
Replication Secret	41
Accounting	41
Attributes	41
Dictionaries	41
Vendor-Specific Attributes	42
Dictionaries and the Make/Model Field	42
Updating Attribute Information	42
Structured Attributes	42
User Attribute Lists	43
Check List Attributes	43
Return List Attributes	44
Structured Attributes in Check Lists and Return Lists	44
Attribute Values	44
Single- and Multi-Valued Attributes	44
Orderable Multi-Valued Attributes	45
System Assigned Values	45
Echo Property	45
Default Values	45
Wildcard Support	46
Attribute Filtering	47
Adding NAS Location Attributes to Access-Requests	47
Specifying IPv4 Address Classes	48
Centralized Configuration Management	48
Proxy RADIUS	50
Proxy RADIUS Authentication	50
Proxy RADIUS Accounting	50

Authentication | 51

Authentication Methods | 52

- Native User Authentication | 52
- Pass-Through Authentication | 52
- Proxy RADIUS Authentication | 52
- External Authentication | 52
- Directed Authentication | 53
- Authenticate-Only Requests | 53

Configuring the Authentication Sequence | 53

Configuring Authentication Methods | 53

Advanced Options | 54

- Account Lockout | 54
- Account Redirection | 55
- Blocklisting | 56
- Allowed Access Hours | 56

Two-Factor Authentication | 57

Password Protocols | 58

- Password Authentication Protocol | 59
- Challenge Handshake Authentication Protocol | 59
- MS-CHAP v2 | 59

Accounting | 60

Accounting Sequence | 61

- Comma-Delimited Log Files | 61
- Proxy RADIUS Accounting | 61
- External Accounting | 61
- Tunneled Accounting | 62
- Directed Accounting | 63

Accounting Spooling | 63

Request Routing | 64

Match Rules | 64

User-Names with a Single Delimiter | 67

- User-Names with a Single Tunnel Delimiter | 67
- User-Names with a Single Realm Delimiter | 68

User-Names with Multiple Suffix Delimiters | 69

User-Names with Multiple Prefix Delimiters	69
Undecorated User-Names	70
Configuring Undecorated User-Name Support	71
Example	71
Request Routing by DNIS	72
Request Routing by Any Attribute	73
Local Services	73
Control over Routing Methods	73
Radius Client Groups	73
IP Address Assignment	74
Address Pools and Replication	75
Hints	76
Resource Management	78
Network Address Assignment	78
How Address Assignment Works	78
Setting Return List Attributes	78
Handling Address Leaks	78
Address Leakage upon Stopping and Starting the Server	79
Overlapping Address Ranges	79
Order of Address Assignment	80
Concurrent Network Connections	80
Concurrent User Connections	80
Concurrent Tunnel Connections	81
Attribute Value Pooling	81
Phantom Records	84
IPv6 Support	84
IPv6 and Steel-Belted Radius Carrier	84
IPv6 Features	84
IPv6 Addressing	85
Address Notation	85
Address Prefixes	86
Address Interface IDs	87
IPv6 Network Numbers	88
IPv6 Support in Steel-Belted Radius Carrier	89

RADIUS IPv6 Attributes | 95**NAS-IPv6-Address | 96****Framed-Interface-Id | 97****Framed-IPv6-Prefix | 97****Login-IPv6-Host | 98****Framed-IPv6-Pool | 98****Framed-IPv6-Route | 99****Framed-IPv6-Address | 99****DNS-Server-IPv6-Address | 100****Route-IPv6-Information | 100****Delegated-IPv6-Prefix-Pool | 101****Stateful-IPv6-Address-Pool | 101****Enabling IPv6 Networking | 101****Configuring IPv6 Scope IDs | 102****Configuring IPv6 Addresses for RADIUS Client Connections | 102****Configuring DNSv6 Support | 102****Administering RADIUS Clients and Client Groups | 104****Overview | 104****Adding a RADIUS Client or Client Group | 104****Editing a RADIUS Client or Client Group | 112****Deleting a RADIUS Client or Client Group | 113****Administering RADIUS Location Groups | 114****About RADIUS Location Groups | 114****Adding a Location Group | 115****Editing a Location Group | 117****Deleting a Location Group | 119****Administering Users | 120****Users Overview | 120****Allowed Access Hours | 120****User Files | 121**

Setting Up Native Users | 122**Adding a Native User | 123****Adding Subattributes to a Structured Attribute | 128****Editing a Native User | 133****Deleting a Native User | 135****Setting Up UNIX Users or Groups | 135****Adding a UNIX User or Group | 136****Editing a UNIX User or Group | 141****Deleting a UNIX User or Group | 143****Administering Profiles | 144****About Profiles | 144****Adding a Check List or Return List Attribute to a Profile | 144****Resolving Profile and User Attributes | 145****Adding a Profile | 146****Editing a Profile | 150****Removing a Profile | 151****Administering Proxy RADIUS | 153****Proxy RADIUS Overview | 153****Proxy RADIUS Authentication | 154****Proxy RADIUS Accounting | 154****Proxy RADIUS Realms | 154****Target Selection within a Realm | 155****Message-Authenticator Support | 156****Proxy Fast-Fail | 156****Static Proxy Accounting | 157****Proxy AutoStop Feature | 158****Routed Proxy Authentication | 158****Operation | 159****Adding a Proxy Target | 160****Editing a Proxy Target | 164****Deleting a Proxy Target | 166**

Steel-Belted Radius Carrier as a Target | 166

 Dictionaries When Steel-Belted Radius Carrier is the Target | 166

 Accepting Packets from Any Proxy | 167

Administering RADIUS Tunnels | 168

About RADIUS Tunnels | 168

 Tunnel Authentication Sequence | 169

 Configuring Tunnel Support | 170

 Called Station ID | 170

 Dictionaries for Tunnel Support | 170

 Concurrent Tunnel Connections | 170

Configuring RADIUS Tunnels | 171

 Adding a Tunnel | 172

 Editing a Tunnel | 177

 Deleting a Tunnel | 179

Configuring Tunnel Name Parsing | 179

Administering Address Pools | 182

Address Pools for Standalone Servers versus Servers in a SSR Cluster | 182

Address Pool Files | 183

Adding an IPv4 Address Pool | 183

Editing an IPv4 Address Pool | 186

Deleting an IPv4 Address Pool | 188

Specifying an IP Address Pool for User/Profile Records | 188

NAD-Specific IP Address Pools | 189

Service-Level IP Address Pools | 190

Specifying IP Address Assignment from a DHCP Server | 191

 Address Allocation | 192

 Address Renewal | 192

 Address Release | 193

 DHCP Option Mapping | 193

 Using Multiple Servers | 194

Setting Up Administrator Accounts | 195

Local Administrator or Group Overview | 195

Adding a Local Administrator or Group | 196

Deleting a Local Administrator or Group | 197

Administrator Configuration Files | 198

Configuring Realm Support | 199

Realm Configuration Files | 199

Stage One of Realm Configuration | 200

Configuring a Proxy RADIUS Realm | 201

Configuring a Directed Realm | 207

Editing the proxy.ini File | 211

Setting Up Smart Static Accounting | 212

Setting Up Proxy RADIUS Realms | 213

Configuration Tasks | 214

Setting Up Directed Realms | 214

How to Update Realm Configuration | 215

Setting Up Filters | 217

Overview | 217

Order of Filter Rules | 219

Values in Filter Rules | 220

Referencing Attribute Filters | 223

Adding a Filter | 224

Searching the Filter List | 227

Editing a Filter | 229

Deleting a Filter | 231

Setting Up Authentication Policies | 232

Authentication Policy Overview | 232

Order of Authentication Methods | 233

Adding EAP Methods to an Authentication Policy | 235

Enabling EAP Methods | 235

Activating an EAP Method | 236

Certificates | 238**Certificate Chains | 239****Certificate Revocation Lists | 239****Configuring Server Certificates | 241****Creating a Certificate | 241****Adding a Certificate | 243****Deleting a Certificate | 245****Trusted Root Certificates | 246****Adding a Trusted Root Certificate | 246****Configuring a CRL Distribution Point Web Proxy | 248****Configuring Authentication Rejection Messages | 250****Configuring the Server | 252****Configuring External Databases | 252****Setting Up EAP Methods | 253****About the Extensible Authentication Protocol | 253****Handling EAP Requests | 254****Automatic EAP Helpers | 254****Authentication Request Routing | 256****EAP-Only Setting | 256****First-Handle-Via-Auto-EAP Setting | 256****EAP-NAK Notifications | 258****Reauthenticating Connections | 258****Certificates | 259****Certificate Chains | 260****Certificate Revocation Lists | 260****EAP-TLS Authentication Protocol | 261****Configuring EAP-TLS as an EAP Authentication Method | 263****Configuring Client Certificate Validation—EAP-TLS | 266****Configuring Session Resumption—EAP-TLS | 268****Configuring Advanced Server Settings—EAP-TLS | 271****Configuring EAP-TLS as an Automatic EAP Helper | 275****Configuring Client Certificate Validation—EAP-TLS Helper | 277****Configuring Secondary Authentication—EAP-TLS Helper | 279**

Configuring Session Resumption—EAP-TLS Helper | 282

Configuring Advanced Server Settings—EAP-TLS Helper | 284

EAP-TTLS Authentication Protocol | 286

Configuring EAP-TTLS as an EAP Authentication Method | 288

Configuring Client Certificate Validation—EAP-TTLS | 289

Configuring Request Filters—EAP-TTLS | 292

Configuring Response Filters—EAP-TTLS | 294

Configuring Session Resumption—EAP-TTLS | 296

Configuring Inner Authentication Settings—EAP-TTLS | 298

Configuring Advanced Server Settings—EAP-TTLS | 299

EAP-PEAP Authentication Protocol | 301

Configuring EAP-PEAP as an EAP Authentication Method | 302

Configuring Request Filters—EAP-PEAP | 304

Configuring Response Filters—EAP-PEAP | 306

Configuring Session Resumption—EAP-PEAP | 308

Configuring Inner Authentication Settings—EAP-PEAP | 310

Configuring Advanced Server Settings—EAP-PEAP | 312

EAP-MD5-Challenge Authentication Protocol | 314

EAP-MS-CHAP-V2 Authentication Protocol | 314

EAP-SIM and EAP-AKA Authentication Protocols | 315

Configuring Replication | 316

Overview of Replication | 316

Replication Requirements | 319

Adding a Replica Server | 319

Enabling a Replica Server | 323

Editing a Replica Server | 325

Deleting a Replica Server | 325

Publishing Server Configuration Information | 326

Notifying Replica RADIUS Servers | 326

Designating a New Primary Server | 327

Making a Standalone Server the Primary Server | 328

Making a Standalone Server a Replica Server | 328

Verifying the Primary and Replica Servers Are Enabled | 329

Demote a Primary or Replica Server to a Standalone Server | 329

Recovering a Replica After a Failed Configuration Package Download | 330

Changing the Name or IP Address of a Server | 330

Replication Error Messages | 331

 Error Messages on Replica Servers | 331

 Error Messages on Primary Servers | 334

3GPP Support | 336

Overview | 336

 Data Connection Process | 336

 Accounting Process | 337

3GPP Configuration | 337

Diameter Operations

Diameter Basics | 339

Diameter Overview | 339

 Diameter Application | 340

Communication between SBR Carrier Server and the Elements in LTE Network | 341

 Communication with Trusted Non-3GPP Network | 342

 Communication with Untrusted Non-3GPP Network | 342

 Communication with HSS | 343

 Communication with Proxy Servers | 343

 Communication with ePDG | 344

 Communication with PDG or PGW | 345

Diameter Authentication Process | 347

Diameter Authorization Process | 347

RADIUS to Diameter Translation | 347

Administering the Local Network Element | 349

Local Network Element Overview | 349

Configuring SBR Carrier Server Identification | 349

 Setting Up the Local Identity for SBR Carrier | 349

 Configuring Local Addresses for the SBR Carrier Server | 351

 Configuring Local Realm for the SBR Carrier Server | 352

Configuring the Diameter Message Transport | 353

Adding a Diameter Message Transport Entry | 354

Editing a Diameter Message Transport Entry | 355

Deleting a Diameter Message Transport Entry | 357

Administering Diameter Remote Network Elements | 358

Remote Network Element Overview | 358

Creating and Configuring a New Diameter Remote Network Element | 359

Adding Diameter Connections to the Diameter Remote Network Element | 361

Assigning Functions to the Diameter Remote Network Element | 366

Configuring Implicit Routing Rules | 369

Editing a Diameter Remote Network Element | 372

Deleting a Diameter Remote Network Element | 375

Administering the Diameter Policy | 376

Policy Overview | 376

Configuring a Local Profile | 376

Creating a Local Profile | 377

Configuring Authorization Attributes | 380

Configuring a Non-3GPP Interworking Policy for SWa or STa Reference Point | 386

Configuring a Non-3GPP Interworking Policy for SWm Reference Point | 389

Configuring a Non-3GPP Interworking Policy for S6b Reference Point | 398

Editing a Local Profile | 405

Deleting a Local Profile | 408

Configuring Local Profile Selection | 408

Creating a New Profile Selection Rule Set | 410

Creating New Matching Rules | 412

Editing Profile Selection Rule Sets | 415

Deleting Profile Selection Rule Sets | 416

Administering Request Routing Rules | 417

Request Routing Rules Overview | 417

- Routing Rule Evaluation and Rule Priorities | 417

Configuring Request Routing Rules | 419

- Defining Explicit Routing Rules | 421

- Defining Conditions for the Explicit Routing Rule | 423

Displaying Diameter Statistics | 429

Statistics Overview | 429

Diameter Statistics | 429

- Local Diameter Host Statistics | 429

- Diameter Peer Statistics | 431

Routing Rule Statistics | 439

Back-End Authentication and Accounting Methods

Configuring SQL Authentication | 442

Overview of SQL Authentication | 442

- SQL Authentication Process | 443

- Stored Procedures | 444

- Connectivity Issues | 444

Configuring SQL Authentication | 445

- Files | 445

- Using the SQL Authentication Configuration File | 446

- Using Multiple SQL Authentication Methods | 446

Connecting to the SQL Database | 446

SQL Statement Construction | 447

- Overlapped Execution of SQL Statements | 450

- %result Parameter | 451

- SQL Authentication and Password Format | 452

- Hashed Passwords | 452

- Automatic Parsing | 453

Working with Stored Procedures in Oracle | 453

Working with Stored Procedures in MS-SQL | 455

Example 1 | 455

Example 2 | 456

Tips on Using SQL Stored Procedures | 456

Calling Stored Procedures | 456

Using the Insert Function | 457

Configuring SQL Accounting | 458

SQL Accounting Overview | 458

Stored Procedures | 459

Connectivity Issues | 460

Configuring SQL Accounting | 461

Files | 461

Using the SQL Accounting Configuration File | 461

Using Multiple SQL Databases | 461

Connecting to the SQL Database | 462

SQL Statement Construction | 463

INSERT Statement and VALUES Section | 463

Using Multiple SQL Statements | 467

Overlapped Execution of SQL Statements | 467

SQL Accounting Return Values | 468

Accounting Stored Procedure Example | 468

Configuring LDAP Authentication | 471

LDAP Authentication Overview | 471

LDAP Variable Table | 472

Types of LDAP Authentication | 473

BindName Authentication | 473

Bind Authentication | 474

Attributes and LDAP Authentication | 474

Configuring LDAP Authentication | 474

Supporting Secure Sockets Layer | 475

Files | 476

LDAP Database Schema | 476

LDAP Authentication and Password Format | 478

Hashed Passwords | 478

Automatic Parsing | 479

LDAP Authentication Sequence | 479

LDAP Authentication Examples | 480

Bind Authentication with Default Profile | 481

BindName Authentication with Callback Number Returned | 482

LDAP Bind with Profile Based on Network Access Server | 482

Signalware SIGTRAN Gateway Support | 484

Overview of Signalware SIGTRAN Protocol Stacks | 484

Proxy RADIUS Authentication and Accounting | 485

Proxy RADIUS Accounting | 485

HSS-Subscriber Database | 486

6

Management Interfaces

Simple Network Management Protocol | 488

SNMP and Steel-Belted Radius Carrier Overview | 488

The SBR Carrier SNMP Package | 488

Supported MIBs | 489

Configuring the SNMP Agent | 491

Running the SNMP Agent | 492

Starting the SNMP Agent | 492

Stopping the SNMP Agent | 493

Rereading the jnprsnmpd.conf File | 493

Logging Behavior of the SNMP Agent | 493

Verifying SNMP Agent Operation | 494

Running the testagent.sh Script | 494

Using the snmpget Command | 494

Using the snmpwalk Command | 495

Resetting Rate Statistics | 496

Troubleshooting | 496

Using the LDAP Configuration Interface | 497

LDAP Configuration Interface File | 497

LDAP Configuration Interface Overview | 498

LDAP Utilities | 499

LDAP Requests | 500

Downloading the LDAP Utilities | 500

LDAP Version Compliance | 501

Configuring the LDAP TCP Port | 501

Example | 502

Configuring the LCI Password | 502

LDAP Virtual Schema | 503

LDAP Rules and Limitations | 509

Using the LCI to Define Structured Attributes in Check Lists and Return Lists | 510

LCI XML Format | 512

LDAP Command Examples | 513

Searching for Records | 513

Modifying Records | 515

Importing Records from Another LDAP Database | 519

Deleting Records | 520

Searching for Active Sessions | 521

```
# ldapsearch -p 667 -h 10.14.1.2 -D cn=admin,o=radius -w radius -b
user=5c:0a:5b:77:7d:8a,radiusstatus=sessions_by_user,o=radius generic1=* | 521
```

LDIF File Examples | 522

Adding RADIUS Clients with LDIF | 522

Adding Users with LDIF | 523

Adding Proxy Targets with LDIF | 526

Adding Tunnels with LDIF | 526

Adding IP Address Pools with LDIF | 527

Configuring a RADIUS Server with LDIF | 528

Statistics Variables | 529

Counter Statistics | 529

stattype: server | 529

stattype: authentication | 529

stattype: accounting | 530

stattype: proxy | 530

Rate Statistics | 531

Optional Authentication Modules

SIM Authentication Module | 536

SIM Authentication Module Component Overview | 536

SIMAuth | 536

Signalware SIGTRAN Protocol Stacks | 537

MAP Gateway (authGateway) Application | 537

GWrelay Application | 537

CDR Accounting | 538

Data Accessors | 538

Operation Overview | 538

SIM Card-Based Authentication | 539

EAP-SIM/EAP-AKA Authorization/Service Delivery | 541

EAP-SIM/EAP-AKA Identities | 542

EAP-SIM/EAP-AKA Fast Reauthentication | 542

SIM Authentication Module Configuration | 543

Special Attribute Handling Features | 543

Assigning IP Addresses Based on Access Point Name (APN) | 543

Overview | 543

Configuration Tasks for Assigning IP Address Based on Access Point Name | 544

Adding Attributes to an Access-Accept | 544

Overview | 544

Data Flow | 545

Configuration Tasks for Adding Attributes to Access-Accept | 546

Files to Configure for Adding Attributes to Access-Accept | 546

Activating the Authentication Method | 546

Kineto S1 Support | 549

Summary of Configuration Tasks for the SIM Authentication Module | 550

Summary of Configuration Tasks for the SIM Authentication Module | 550

SIM Authentication Module Configuration with a SIGHUP (1) Signal | 553

Overview of the WiMAX Mobility Module | 554

Supported Features of the WiMAX Mobility Module | 554

WiMAX Network Reference Model | 555

Home Network Communication Flow Example | 556

AAA-Generated Cryptographic Keys | 559

Home Agent Root Key (HA-RK) | 559

Allowing the VAAA to Assign the HA-RK | 559

DHCP Server Root Key (DHCP-RK) | 560

EAP Authentication Methods and EAP-Derived Cryptographic Keys | 560

Master Session Key (MSK) | 561

Extended Master Session Key (EMSK) | 561

EMSK-Derived Key Generation and Identification | 561

MSK and EMSK-Derived Key Lifetime and Deprecation | 562

EMSK-Derived Key Storage and Retrieval | 562

WiMAX Vendor Specific Attribute (VSA) Format | 562

Structured Attributes | 564

WiMAX Capabilities Negotiation | 565

WiMAX-Capability Attribute | 565

WiMAX-Capability Structured Attribute | 565

Enabling WiMAX Capabilities Negotiation | 566

Home Agent and DHCP Server Assignment | 567

Assignment Using Return List Attributes | 567

Assignment Using Statically Weighted Round-Robin Groups | 567

Assignment Using the Smart Dynamic Home Agent Assignment Feature | 567

WiMAX Post-Paid (Offline) Accounting | 568

Flow-Based Accounting | 568

IP-Session-Based Accounting | 569

WiMAX Prepaid Accounting | 570

Prepaid Scenarios | 571

Single-Service Prepaid Solution | 571

Multi-Service Prepaid Solution | 571

Data Flow for Prepaid Accounting in SBR Carrier | 572

Data Flow for Single-Service Prepaid Accounting Model | 572

Data Flow for Multi-Service Prepaid Accounting Models | 573

Categorizing Access-Requests from Different Devices | 578

- Access-Request from the ASN-GW | 579

- Access-Request from the Home Agent | 579

- Access-Request from the DHCP Server | 579

- Categorization Rules | 579

Configuring the WiMAX Mobility Module | 580**Before You Begin | 580****Configuring the radius.ini File for WiMAX | 581**

- Configuring Support for Authorize-Only Requests | 582

- Enabling the WiMAX Module and Configuring What Request Types Are Supported | 583

Configuring the Home Agent and DHCP Server Assignment | 585

- Define the List of Home Agents and DHCP Servers | 585

- Configuring Return List Attributes to Assign the Home Agent and DHCP Server | 585

 - Assignment When Acting as the HAAA Server | 585

 - Assignment When Acting as the VAAA Server | 586

- Configuring Statically Weighted Round-Robin Groups to Assign the Home Agent and DHCP Server | 586

- Configuring the Smart Dynamic Home Agent Assignment Feature | 588

 - Smart Dynamic Home Agent Assignment Configuration Overview | 588

 - Operation of the Smart Dynamic Home Agent Assignment Feature | 591

 - Access-Request Processing | 591

Configuring WiMAX Clients | 592**Configuring WiMAX Users and Profiles | 593**

- Configuring the WiMAX-Capabilities Negotiation | 594

 - Example Configuration for New Session Hotlining | 595

Configuring the EAP Methods for WiMAX | 607

- EAP-TTLS Secondary Authentication Support | 608

Optional Session State Register (High Availability) Module for a Clustered Environment

Session State Register Overview | 610

SSR Cluster Overview | 610

Data Replication Between Two Different or Remote SSR Clusters | 611

Configuring the Client Component and Server Component | 612

List of CST Fields That Can Be Replicated Across SSR Clusters | 613

Possible Uses and Limitations of the Geo-redundancy Feature | 614

SSR Cluster Concepts and Terminology | 615

Session State Register Servers | 615

Session State Register Nodes | 616

SSR Data Entities | 617

Cluster Configurations | 618

Session State Register Scaling | 619

Adding a Data Node Expansion Kit | 619

Adding a Third Management Node | 619

Adding More SBR Carrier Front End Servers | 619

Cluster Network Requirements | 620

Supported SBR Carrier SSR Cluster Configurations | 621

Failover Overview | 622

Failover Examples | 623

Session State Register Database Tables | 625

IP Address Pools | 625

Subscriber Session Data Controls | 625

Application Support | 626

Session State Register Administration | 628

SSR Administration Overview | 628

Overview of Starting and Stopping a Session State Register Cluster | 629

Starting the Cluster | 629

Stopping the Cluster | 630

Stopping a Single Node | 631

Starting a Single Node | 632

sbrd | 632

Running sbrd on Session State Register Nodes | 632

When to Stop, Start, or Restart SBR Carrier Nodes | 634

Administration Scripts Overview | 637

SSR Database Management Scripts | 639

Using the Monitor Script | 639

Monitor.sh | 639

Creating and Destroying the SSR Database | 641

CreateDB.sh | 641

DestroyDB.sh | 643

Creating a Demonstration Database | 643

DemoSetup.sh | 643

Steel-Belted Radius Carrier Node Administration Scripts | 646

Using IP Address and IP Address Pool Scripts | 646

Using Management Mode | 647

ClearCache.sh | 649

ShowCaches.sh | 650

AddPool.sh | 652

RenamePool.sh | 654

DelPool.sh | 655

ShowPools.sh | 656

AddRange.sh | 658

DelRange.sh | 660

ShowRanges.sh | 661

KillZombieAddrs.sh | 662

ShowAddrs.sh | 664

BackupIP.sh | 667

RestoreIP.sh | 668

SSR Session Management | 670

Session Management Scripts | 670

ShowSessions.sh | 670

DelSession.sh | 673

User Concurrency Scripts | 675

ShowUserConc.sh | 675

DelUserConc.sh | 678

Administration Script Control Files | 678

Optional Concurrency Module

Managing User Concurrency with Session State Register | 682

Overview | 682

How User Concurrency Works | 683

UserConcurrencyID Construction | 683

Username | 684

Authentication Method | 684

Authentication Method Consistency | 685

Retrospective Dynamicity | 686

Consequences of the No-Retrospective-Dynamicity Model | 686

Benefits of the No-Retrospective-Dynamicity Model | 686

Managing Concurrency with Attributes in Session State Register | 688

Overview | 688

How Attribute-Based Concurrency Works | 689

Configuring Attribute-Based Concurrency | 690

Setting the Size of the ID Field in the User Concurrency Table | 690

Specifying the User Attribute | 691

Distributing the Files | 693

Managing and Controlling Sessions

Introduction to Managing and Controlling Sessions in SBR Carrier | 695

Overview of Managing and Controlling Sessions in SBR Carrier | 695

Introduction | 695

Storing Sessions in the CST in a Standalone Server versus the SSR Cluster | 695

Storing Sessions in the CST of a Standalone Server | 696

Storing Sessions in the CST of the SSR Cluster | 696

Session Management and Control Capabilities | 697

Viewing and Deleting Sessions | 697

Disconnecting or Changing the State of Active Sessions | 697

Available User Interfaces for Managing and Controlling Sessions | 698

Web GUI | 698

Command Line Utility | 698

XML over HTTPS Interface | 698

Administrative Scripts | 699

Overview of the Optional Session Control Module | 700

Change of Authorization/Disconnect Messages Overview | 700

How Disconnect Messages Work | 702

How Change of Authorization Messages Work | 702

How Steel-Belted Radius Carrier Processes CoA/DM Messages | 702

Current Sessions Table | 702

Formatting and Sending CoA/DM Requests with the Correct Attributes | 703

Converting .dct Files to .dic Files | 705

Controlled Devices and Actions | 705

Sequence and Flow of CoA/DM Requests Through Steel-Belted Radius Carrier | 706

Implementing CoA/DM Support | 711

Step 1: Develop a Deployment Plan | 711

Step 2: Consult Your NAS-Specific Documentation | 712

Step 3: Configure Each NAS as a Client in Steel-Belted Radius Carrier | 712

Step 4: Configure the deviceModels.xml File | 713

Step 5: Configure the Current Sessions Table (CST) for Your Environment | 713

Processing Dynamic Authorization (CoA/DM) Messages as a Proxy Server | 713

Attributes and CoA/DM Forwarding Methods | 714

Processing Dynamic Authorization (CoA/DM) Messages as a Proxy Target | 715

Settings to Support the Proxy CoA/DM Functionality | 715

Using Web GUI to Manage and Control Sessions | 718

Current Sessions Overview | 718

Searching for Sessions Using Web GUI | 719

Session Query Fields and Searchable Attributes | 720

Viewing Session Detail | 723

Deleting Sessions | 724

Setting Session Limits with Web GUI | 724

Factors Affecting the Number of Sessions Returned | 724

Number of Sessions Returned | 725

Executing CoA and Disconnect Requests Using Web GUI | 725

Example of Executing a Disconnect Action | 726

Using the Command Line Utility to Manage and Control Sessions | 730

Command Line Utility Overview | 730

Starting the Command Line Utility | 731

Example | 731

Using Command Line Arguments | 731

Access Control Arguments | 731

Syntax | 731

Arguments | 732

Example | 733

Action Arguments | 733

Syntax | 733

Arguments | 733

IP Address Ranges | 736

Unique Session IDs | 737

Setting Session Limits Using the Command Line Utility | 737

Factors Affecting the Number of Sessions Returned | 737

Number of Sessions Returned | 737

Examples of Limiting the Number of Sessions Returned Using the Command Line Utility | 738

Examples of Issuing CoA/DM Requests Using the Command Line Utility | 739

Query Example Using Wildcard | 739

Disconnect Example | 739

Lawful Intercept Example | 740

Shortcut Arguments | 741

Syntax | 741

Arguments | 741

Disconnect Example with Shortcut | 742

Finding All Sessions Using the Command Line Utility | 742

Example of Finding All Sessions | 742

Configuring the deviceModels.xml File | 743

Summary of Allowed Elements in the deviceModels.xml File | 744

Element: action | 747

XML Instance Representation | 747

Schema Component Representation | 747

Element: actions | 748

XML Instance Representation | 748

Schema Component Representation | 748

Element: attributes | 748

XML Instance Representation | 748

Schema Component Representation | 749

Element: controlledDeviceModel | 749

XML Instance Representation | 750

Schema Component Representation | 750

Element: controlledDeviceModels | 751

XML Instance Representation | 751

Schema Component Representation | 751

Element: defaultAttribute | 752

XML Instance Representation | 752

Schema Component Representation | 752

Element: localSessionQuery | 752**XML Instance Representation | 752****Schema Component Representation | 753****Element: onFailure | 753****XML Instance Representation | 753****Schema Component Representation | 753****Element: onSuccess | 753****XML Instance Representation | 754****Schema Component Representation | 754****Element: onTimeout | 754****XML Instance Representation | 754****Schema Component Representation | 754****Element: overrideAttribute | 755****XML Instance Representation | 755****Schema Component Representation | 755****Element: radiusPort | 755****XML Instance Representation | 756****Schema Component Representation | 756****Element: radiusPorts | 756****XML Instance Representation | 756****Schema Component Representation | 756****Element: radiusRequest | 757****XML Instance Representation | 757****Schema Component Representation | 757****Element: requiredAttribute | 758****XML Instance Representation | 758****Schema Component Representation | 758****Element: sessionStop | 758****XML Instance Representation | 758****Schema Component Representation | 759**

XML over HTTPS Interface | 760

XML over HTTPS Interface Overview | 760

Transport Protocol | 761

XML Statement Construction | 761

Client Request Schema Example | 762

Client Request Elements | 763

Element: attribute | 763

XML Instance Representation | 763

Schema Component Representation | 764

Element: attributes | 764

XML Instance Representation | 764

Schema Component Representation | 764

Element: body | 764

XML Instance Representation | 765

Schema Component Representation | 765

Element: envelope | 765

XML Instance Representation | 765

Schema Component Representation | 765

Element: header | 766

XML Instance Representation | 766

Schema Component Representation | 766

Element: request | 766

XML Instance Representation | 766

Schema Component Representation | 767

Client Request Examples | 767

Example: Query | 767

Example: Query | 767

Example: RADIUS Disconnect | 768

Example: RADIUS Disconnect | 768

Example: RADIUS Disconnect | 769

Example: (CoA) Action Called Intercept | 769

Client Response Schema Example | 770

Client Response Elements | 774**Element: attribute | 774**

XML Instance Representation | 774

Schema Component Representation | 774

Element: attributes | 775

XML Instance Representation | 775

Schema Component Representation | 775

Element: body | 775

XML Instance Representation | 775

Schema Component Representation | 776

Element: clientRequest | 776

XML Instance Representation | 776

Schema Component Representation | 776

Element: clientResponse | 776

XML Instance Representation | 777

Schema Component Representation | 777

Element: clientResult | 777

XML Instance Representation | 777

Schema Component Representation | 777

Element: clientResults | 778

XML Instance Representation | 778

Schema Component Representation | 778

Element: defaultAttribute | 778

XML Instance Representation | 778

Schema Component Representation | 779

Element: deviceRequest | 779

XML Instance Representation | 779

Schema Component Representation | 779

Element: deviceRequestSpec | 780

XML Instance Representation | 780

Schema Component Representation | 780

Element: deviceResponse | 780

XML Instance Representation | 782

Schema Component Representation | 782

Element: deviceResult | 782

XML Instance Representation | 782

Schema Component Representation | 783

Element: deviceResults | 783

XML Instance Representation | 783

Schema Component Representation | 783

Element: envelope | 784

XML Instance Representation | 784

Schema Component Representation | 784

Element: header | 784

XML Instance Representation | 784

Schema Component Representation | 784

Element: optionalAttribute | 785

XML Instance Representation | 785

Schema Component Representation | 785

Element: overrideAttribute | 785

XML Instance Representation | 785

Schema Component Representation | 785

Element: requiredAttribute | 786

XML Instance Representation | 786

Schema Component Representation | 786

Element: sessionData | 786

XML Instance Representation | 786

Schema Component Representation | 787

Element: sessionRequest | 787

XML Instance Representation | 787

Schema Component Representation | 787

Element: sessionResponse | 787

XML Instance Representation | 787

Schema Component Representation | 788

Element: sessionResult | 788

XML Instance Representation | 788

Schema Component Representation | 788

Element: sessionResults | 789

XML Instance Representation | 789

Schema Component Representation | 789

Client Response Examples | 789

Example: Client Response to Query for Username 'bob' | 790

Example: Client Response to Query for Any Username Using Wildcard | 790

Example: Client Response to Request for Action Called "foo" on Username TestUser9 | 793

Example: Client Response to Request for Action Called "foo" on Username TestUser99 | 795

Example: Client Response to RADIUS Disconnect | 797

Example: Client Response to Action Intercept | 799

Example: Client Response to Action Intercept | 800

Example: Client Response to Action Intercept | 802

Example CoA/DM Configuration | 806

Requirements of the CoA/DM Requests | 806

Requirements for the Disconnect Message Request | 806

Requirements for the CoA (Hotline) Request | 807

Requirements for Supporting the Attributes in CoA/DM Requests | 808

Dictionaries | 808

deviceModels.xml | 809

Configuring the Attribute Handling Parameters | 811

radius.ini | 811

classmap.ini | 811

Example Result | 812

Configuring Lawful-Intercept between SBR Carrier and ERX Device | 813

Statistics and Reporting

Displaying Statistics | 817

Displaying Authentication Statistics | 817

Displaying Accounting Statistics | 820

Displaying Proxied Request Statistics | 823

Displaying RADIUS Client Statistics | 825

Displaying RADIUS Proxy Targets Statistics | 829

Displaying IP Address Pool Statistics | 833

Logging and Reporting | 834

Logging Files | 834

Displaying Authentication Log Files | 835

File Permissions for Log Files | 836

Security Groups and Permissions | 836

Using the User File Creation Mode Mask | 837

Implementing Default File Permissions in SBR Carrier | 838

Implementing Override File Permissions in SBR Carrier | 839

Enabling or Disabling the Authentication Log Files | 840

Viewing the Authentication Log Files | 842

Saving the Log Files | 843

Searching the Log Files | 844

Using the Locked Accounts List | 846

Configuring Locked Account Settings | 846

Unlocking a Locked Account | 847

Configuring the Log Retention Period | 848

Cluster Management Nodes | 849

SSR Data Nodes | 849

Using the Server Log File | 850

Level of Logging Detail | 851

Using the Authentication Log File | 852

Authentication Log File Format | 852

First Line Headings | 854

Comma Placeholders | 854

Using the Accounting Log File | 854

Accounting Log File Format | 855

First Line Headings | 856

Comma Placeholders | 857

Standard RADIUS Accounting Attributes | 857

Optional Scripting Module

Introduction to Scripting | 861

Scripting Overview | 861

Script Types | 862

LDAP Authentication | 863

Realm Selection | 863

Attribute Filter | 864

About JavaScript | 865

Creating Scripts | 866

Script Development Steps | 866

JavaScript Initialization Files | 867

[Settings] Section | 867

[Script] Section | 868

[ScriptTrace] Section | 869

[Failure] Section | 870

Writing Steel-Belted Radius Carrier Scripts in JavaScript | 870

Programming in JavaScript | 870

Hidden Wrapper Function | 871

Script Return Values | 871

Initializing Reusable Data Objects | 872

General Recommendations | 872

Saving the Script File | 873

Sample Script | 873

Debugging Scripts | 876

SbrWriteToLog() | 876

SbrTrace and ScriptTraceLevel | 877

scriptcheck | 879

Unpacking the scriptcheck Utility | 879

Running the scriptcheck Utility | 880

Creating LDAP Scripts | 881

LDAP Basics | 881

LDAP Request Life Cycle | 882

Unscripted LDAP Searches | 883

LDAP Script Basics | 884

Working with the Variable Table | 884

Invoking LDAP Queries | 885

Writing to the Steel-Belted Radius Carrier Log | 885

Choosing the Return Code | 885

Script Return Codes | 886

SCRIPT_RET_SUCCESS | 886

SCRIPT_RET_DO_NOT_AUTHENTICATE | 886

SCRIPT_RET_TRY_NEXT_AUTH_METHOD | 886

SCRIPT_RET_NOT_AUTHENTICATED | 886

SCRIPT_RET_FAILURE | 887

SCRIPT_RET_INVALID_CODE | 887

LDAP Script Return Codes | 887

LDAP Script Examples | 888

Example 1: Simple Authentication | 888

Example 2: Profile Assignment | 889

Example 3: Received Attribute Normalization | 890

Example 4: Conditional Profile Assignment from User Attribute | 891

Creating Realm Selection Scripts | 895

Realm Selection Script Functions | 896

Enabling Built-In Realm Selection Methods | 896

Choosing the Return Code | 897

Configuring Realm Selection Scripts | 898

Core Realm Selection Scripts | 898

[Processing] Section | 898

Tunneled Authentication Plug-in Realm Selection Scripts | 899

Realm Selection Script Examples | 902

Example 1: Querying Multiple SQL Databases | 902

Example 2: Using JavaScript to Manipulate Request Attributes | 904

Creating Attribute Filter Scripts | 908

Using Attribute Filter Scripts | 908

Attribute Filter Script Functions | 909

Choosing the Return Code | 910

Configuring Attribute Filter Scripts | 910

 Defining Scripted Filters | 910

Attribute Filter Script Examples | 914

 Example 1: Using an LDAP Query to Select a Static Filter to Execute | 914

 Example 2: Adding Values to Multi-Valued Attributes | 916

Working with Data Accessors | 919

Data Accessor Overview | 919

Variable Containers | 920

Internal Variable Table (LDAP Only) | 921

Data Accessor Configuration | 921

 SQL Data Accessor Configuration | 921

 [Bootstrap] Section | 921

 [Results] Section | 922

 [Settings] Section | 923

 [VariableTypes] Section | 925

 LDAP Data Accessor Configuration | 926

 [Bootstrap] Section | 926

 [Attributes/name] Sections | 926

 [Response] Section | 927

 [Search/name] Sections | 928

 [Request] Section | 930

 [Defaults] Section | 931

 [Server/name] Sections | 931

 [Server] Section | 934

 [Settings] Section | 935

 [VariableTypes] Section | 937

Data Conversion Rules | 938

 Output Container | 938

 Input Container | 939

Examples | 939**Example 1 | 939****Example 2 | 940****Example 3 | 941****Example 4 | 941****Supported Data Types and Conversions | 942****Data Accessor Configuration File Examples | 944****Example: LDAP Data Accessor Configuration File | 944****Example: SQL Data Accessor Configuration File | 945****Script Reference | 947****JavaScript Types | 947****API Method Support by Script Type | 948****Local and Global Variable Declarations | 949****Global Object | 950****Logging and Diagnostic Methods | 950****SbrWriteToLog() | 950****SbrWriteToLogEx() | 951****SbrTrace() | 952****Ldap Object | 952****Ldap Methods | 952****Ldap.Search() | 952****LdapVariables Object | 954****LdapVariables Methods | 954****LdapVariables.Get() | 954****LdapVariables.Add() | 955****LdapVariables.Reset() | 955****RealmSelector Object | 956****Constructor | 956****new RealmSelector() | 956****new CSTAccessor() | 956**

new SessionControl()	957
RealmSelector Methods	957
Execute()	958
SetAuthUserName()	958
SetAuthProfile()	959
SetLocationGroupProfile()	959
CSTAccessor Methods	960
Get()	960
SetAuthUserName()	960
SetAuthProfile()	961
SetLocationGroupProfile()	961
SessionControl Object	962
Properties	962
SUCCESS	962
FAILURE	962
TIME_OUT	962
MISSING_INFO	963
SessionControl Methods	963
AddAttribute()	963
Execute()	963
AttributeFilter Object	964
Constructor	964
new AttributeFilter()	964
AttributeFilter Methods	965
Get()	965
Add()	965
Reset()	966
Replace()	967
Execute()	967
AttributeFilter API	968
DataAccessor Object	969
Properties	969
FOUND	969
NOTFOUND	969

	FAILED 969
	PASSWORDFAILED 969
	Constructor 969
	new DataAccessor() 970
	Methods 970
	SetInputVariable() 970
	GetOutputVariable() 971
	Execute() 971
	Clear() 972

Part 13

Appendixes

Appendix A

When and How to Stop and Restart Steel-Belted Radius Carrier | 974

Stopping the Steel-Belted Radius Carrier Server | 977

Starting the Steel-Belted Radius Carrier Server | 977

Appendix B

Authentication Protocols | 979

Appendix C

Importing and Exporting Data | 981

Importing Information from an XML File | 981

Exporting Information to an XML File | 984

Appendix D

Technical Bulletins | 986

Service Type Mapping | 986

 Configuration | 987

 Local User Database Entries | 987

 servtype.ini File | 988

Ascend Filter Translation | 994

 Configuration | 995

 Syntax | 995

Changing IP Addresses in an SSR Cluster Without Redefining the Cluster | 997

Appendix E

SIR.sh Script | 998

Syntax | 999

Options | 999

Example | 1000

Appendix F

Thread and Flood Control Mechanism | 1002

Thread Control Settings | 1003

Flood Control Settings | 1003

SNMP Trap | 1004

Logging Information | 1004

Glossary | 1006

About This Guide

IN THIS SECTION

- Objective | lxxix
- Audience | lxxix
- Documentation Conventions | lxxx
- Related Documentation | lxxxii
- Obtaining Documentation | lxxxviii
- Documentation Feedback | lxxxviii
- Requesting Technical Support | lxxxix

This preface provides the following guidelines for using the *SBR Carrier Administration and Configuration Guide*:

Objective

This guide describes how to configure and administer Steel-Belted Radius Carrier software running on Solaris and Red Hat Enterprise Linux operating systems.

Audience

This guide is intended for network administrators working for wireline and wireless carriers. It provides the information that administrators need to implement and maintain authentication, authorization, and accounting (AAA) services.

This guide assumes that you are familiar with general RADIUS and networking concepts, as well as the network environment that includes Steel-Belted Radius Carrier.

If you use Steel-Belted Radius Carrier with third-party products such as Oracle, this guide assumes you are familiar with the installation, configuration, and use of those products.

Documentation Conventions

Table 1 on page lxxx defines notice icons used in this guide.

Table 1: Notice Icons

Icon	Meaning	Description
NOTE:	Informational note	Indicates important features or instructions.
CAUTION:	Caution	Indicates a situation that might result in loss of data or hardware damage.
WARNING	Warning	Alerts you to the risk of personal injury.

Table 2 on page lxxx describes the text conventions used throughout this manual.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Text Conventions		
Bold text like this	Represents commands and keywords in text.	<ul style="list-style-type: none"> Issue the clock source command. Specify the keyword exp-msg.
Bold text like this	Represents text that the user must type.	host1(config)#traffic class low-loss1
Fixed-width text like this	Represents information as displayed on your terminal's screen.	<pre>host1#show ip ospf 2 Routing Process OSPF 2 with Router ID 5.5.0.250 Router is an Area Border Router (ABR)</pre>
<i>Italic text like this</i>	<ul style="list-style-type: none"> Emphasizes words. Identifies variables. Identifies chapter, appendix, and book names. 	<ul style="list-style-type: none"> There are two levels of access, <i>user</i> and <i>privileged</i>. <i>clusterId</i>, <i>ipAddress</i>. <i>Appendix A, System Specifications</i>.
Plus sign (+) linking key names	Indicates that you must press two or more keys simultaneously.	Press Ctrl+b.

Table 2: Text and Syntax Conventions (continued)

Convention	Description	Examples
<i>radiusdir</i>	Represents the directory into which Steel-Belted Radius Carrier has been installed. The default location is /opt/JNPRsbr/radius on Solaris systems, but any location may be specified during installation.	Change directories to /radiusdir/radiusdir
Syntax Conventions		
Plain text like this	Represents keywords.	terminal length
<i>Italic text like this</i>	Represents variables.	<i>mask</i> , <i>accessListName</i>
< > (angle brackets)	Enclose a list of possible selections.	<add replace>
(pipe symbol)	Represents a choice to select one keyword or variable in a list of choices that is separated by the pipe symbol.	diagnostic line In this example, you must specify <i>add</i> or <i>replace</i> but cannot specify both: <add replace>
[] (brackets)	Represent optional keywords or variables.	[internal external], or <add replace> = Attribute [,Attribute], where the second attribute is identified as optional by the brackets. When they are used in a configuration files brackets identify a section of the file. In scripts or in operating system commands, brackets indicate the default response or entry.
[]* (brackets and asterisk)	Represent optional keywords or variables that can be entered more than once.	[level1 level2 l1]*
{ } (braces)	Represent required keywords or variables.	{ permit deny } { in out } { clusterId ipAddress }

Related Documentation

Table 3 on page lxxxii lists and describes the Steel-Belted Radius Carrier documentation set:

Table 3: Steel-Belted Radius Carrier Documentation

Document	Description
<i>Steel-Belted Radius Carrier Installation Guide</i>	Describes how to install the Steel-Belted Radius Carrier software on the server.
<i>Steel-Belted Radius Carrier Administration and Configuration Guide</i>	Describes how to configure and operate the Steel-Belted Radius Carrier and its separately licensed modules.
<i>Steel-Belted Radius Carrier Reference Guide</i>	Describes the settings and valid values of the Steel-Belted Radius Carrier configuration files.
<i>Steel-Belted Radius Carrier Performance, Planning and Tuning Guide</i>	Provides tips, use cases, and tools you need to: <ul style="list-style-type: none"> • Improve SBRC performance through planning, analysis, and configuration • Increase SBRC throughput and reliability • Analyze specific use cases, in the lab or in the production environment, to identify areas of potential performance enhancement and to limit the impact of resource constraints and failure scenarios
<i>Steel-Belted Radius Carrier Release Notes</i>	Contains the latest information about features, changes, known problems, and resolved problems.

NOTE: If the information in the Release Notes differs from the information in any guide, follow the Release Notes.

Requests for Comments (RFCs)

The Internet Engineering Task Force (IETF) maintains an online repository of Request for Comments (RFCs) online at <http://www.ietf.org/rfc.html>.

Table 4 on page lxxxiii lists the RFCs that apply to Steel-Belted Radius Carrier.

Table 4: RFCs Related to Steel-Belted Radius Carrier

RFC Number	Title
RFC 1035	<i>Domain Names - Implementation and Specification</i> . P. Mockapetris. November 1987.
RFC 1155	<i>Structure and Identification of Management Information for TCP/IP-based Internets</i> . M. Rose, K. McCloghrie, May 1990.
RFC 1213	<i>Management Information Base for Network Management of TCP/IP-based internets: MIB-II</i> . K. McCloghrie, M. Rose, March 1991.
RFC 2006	<i>The Definitions of Managed Objects for IP Mobility Support using SMIv2</i> . D. Cong and others. October 1996.
RFC 2104	<i>HMAC: Keyed-Hashing for Message Authentication</i> . H. Krawczyk, M. Bellare, R. Canetti. February 1997.
RFC 2246	<i>The TLS Protocol</i> . T. Dierks, C. Allen. January 1999.
RFC 2271	<i>An Architecture for Describing SNMP Management Frameworks</i> . D. Harrington, R. Presuhn, B. Wijnen, January 1998.
RFC 2284	<i>PPP Extensible Authentication Protocol (EAP)</i> . L. Blunk, J. Vollbrecht, March 1998.
RFC 2433	<i>Microsoft PPP CHAP Extensions</i> . G. Zorn, S. Cobb, October 1998.
RFC 2548	<i>Microsoft Vendor-specific RADIUS Attributes</i> . G. Zorn. March 1999.
RFC 2607	<i>Proxy Chaining and Policy Implementation in Roaming</i> . B. Aboba, J. Vollbrecht, June 1999.
RFC 2618	<i>RADIUS Authentication Client MIB</i> . B. Aboba, G. Zorn. June 1999.
RFC 2619	<i>RADIUS Authentication Server MIB</i> . G. Zorn, B. Aboba. June 1999.
RFC 2620	<i>RADIUS Accounting Client MIB</i> . B. Aboba, G. Zorn. June 1999.
RFC 2621	<i>RADIUS Accounting Server MIB</i> . G. Zorn, B. Aboba. June 1999.
RFC 2622	<i>PPP EAP TLS Authentication Protocol</i> . B. Aboba, D. Simon, October 1999.

Table 4: RFCs Related to Steel-Belted Radius Carrier (continued)

RFC Number	Title
RFC 2719	<i>Framework Architecture for Signaling Transport</i> . L. Ong et al., October 1999
RFC 2809	<i>Implementation of L2TP Compulsory Tunneling via RADIUS</i> . B. Aboba, G. Zorn. April 2000.
RFC 2865	<i>Remote Authentication Dial In User Service (RADIUS)</i> . C. Rigney, S. Willens, A. Rubens, W. Simpson. June 2000.
RFC 2866	<i>RADIUS Accounting</i> . C. Rigney. June 2000.
RFC 2867	<i>RADIUS Accounting Modifications for Tunnel Protocol Support</i> . G. Zorn, B. Aboba, D. Mitton. June 2000.
RFC 2868	<i>RADIUS Attributes for Tunnel Protocol Support</i> . G. Zorn, D. Leifer, A. Rubens, J. Shriver, M. Holdrege, I. Goyret. June 2000.
RFC 2869	<i>RADIUS Extensions</i> . C. Rigney, W. Willats, P. Calhoun. June 2000.
RFC 2882	<i>Network Access Servers Requirements: Extended RADIUS Practices</i> . D. Mitton. July 2000.
RFC 2960	<i>Stream Control Transmission Protocol</i> . R. Stewart and others. October 2000.
RFC 3046	<i>DHCP Relay Agent Information Option</i> . M. Patrick. January 2001.
RFC 3118	<i>Authentication for DHCP Messages</i> . R.Droms and others. June 2001.
RFC 3162	<i>RADIUS and IPv6</i> . B. Aboba, G. Zorn, D. Mitton. August 2001.
RFC 3344	<i>IP Mobility Support for IPv4</i> . C. Perkins. August 2002.
RFC 3539	<i>Authentication, Authorization, and Accounting (AAA) Transport Profile</i> . B. Aboba, J. Wood. June 2003.
RFC 3575	<i>IANA Considerations for RADIUS (Remote Authentication Dial-In User Service)</i> . B. Aboba, July 2003.
RFC 3576	<i>RFC3576 - Dynamic Authorization Extensions to Remote to Remote Authentication Dial In User Service</i> . Network Working Group, 2003

Table 4: RFCs Related to Steel-Belted Radius Carrier (continued)

RFC Number	Title
RFC 3579	<i>RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)</i> . B. Aboba, P. Calhoun, September 2003.
RFC 3580	<i>IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines</i> . P. Congdon, B. Aboba, A. Smith, G. Zorn, J. Roese, September 2003.
RFC 3588	<i>Diameter Base Protocol</i> . P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko. September 2003.
RFC 3748	<i>Extensible Authentication Protocol</i> . B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz. June 2004.
RFC 3957	<i>Authentication, Authorization, and Accounting (AAA) Registration Keys for Mobile IPv4</i> . C. Perkins and P. Calhoun. March 2005.
RFC 4005	<i>Diameter Network Access Server Application</i> . P. Calhoun, G. Zorn, D. Spence, D. Mitton. August 2005.
RFC 4017	<i>Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs</i> . D. Stanley and others. March 2005.
RFC 4072	<i>Diameter Extensible Authentication Protocol (EAP) Application</i> . P. Eronen, G. Zorn, T. Hiller. August 2005.
RFC 4186	<i>Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)</i> . H. Haverinen, J. Salowey. January 2006.
RFC 4187	<i>Extensible Authentication Protocol Method for Global System for 3rd Generation Authentication and Key Agreement (EAP-AKA)</i> . J. Arkko, H. Haverinen. January 2006.
RFC 4282	<i>The Network Access Identifier</i> . B. Aboba and others. December 2005.
RFC 4284	<i>Identity Selection Hints for the Extensible Authentication Protocol (EAP)</i> . F. Adrangi, V. Lortz, F. Bari, P. Eronen. January 2006.
RFC 4306	<i>Internet Key Exchange (IKEv2) Protocol</i> . C. Kaufman. December 2005.

Table 4: RFCs Related to Steel-Belted Radius Carrier (continued)

RFC Number	Title
RFC 4372	<i>Chargeable User Identity</i> . F. Adrangi and others. January 2006.
RFC 4510	<i>Lightweight Directory Access Protocol (LDAP) Technical Specification Road Map</i> . K. Zeilenga, June 2006.
RFC 4666	<i>Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) - User Adaptation Layer (M3UA)</i> . K. Morneault, J. Pastor-Balbas. September 2006.
RFC 4668	<i>RADIUS Authentication Client MIB for IPv6</i> . D. Nelson. August 2006.
RFC 4669	<i>RADIUS Authentication Server MIB for IPv6</i> . D. Nelson. August 2006.
RFC 4670	<i>RADIUS Accounting Client MIB for IPv6</i> . D. Nelson. August 2006.
RFC 4671	<i>RADIUS Accounting Server MIB for IPv6</i> . D. Nelson. August 2006.
RFC 5281	<i>Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)</i> . P. Funk, S. Blake-Wilson. August 2008.
RFC 5448	<i>Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')</i> . J. Arkko, V. Lehtovirta, P. Eronen. May 2009.
RFC 5997	<i>Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol</i> . A. DeKok. August 2010.
RFC 6733	<i>Diameter Base Protocol</i> . V. Fajardo, J. Arkko, J. Loughney, G. Zorn. October 2012.
RFC 6911	<i>RADIUS Attributes for IPv6 Access Networks</i> . W. Dec, B. Sarikaya, G. Zorn, D. Miles, B. Lourdelet. April 2013.

3GPP Technical Specifications

The Third-Generation Partnership Project (3GPP) and 3GPP2 maintains an online repository of Technical Specifications and Technical Reports at <http://www.3gpp.org> and <http://www.3gpp2.org>, respectively.

Table 5 on page lxxxvii lists the 3GPP Technical Specifications that apply to Steel-Belted Radius Carrier.

Table 5: 3GPP Technical Specifications

3GPP TS Number	Title	Applicable Sections
3GPP TS 22.234 Version 12.0.0	<i>Requirements on 3GPP system to Wireless Local Area Network (WLAN) interworking</i>	<ul style="list-style-type: none"> • Section 5.1.7: Interworking between PLMN and WLANs
3GPP TS 23.003 Version 12.6.0	<i>Numbering, addressing, and identification</i>	<ul style="list-style-type: none"> • Section 2.2: Composition of IMSI
3GPP TS 23.008 Version 12.6.0	<i>Organization of subscriber data</i>	<ul style="list-style-type: none"> • Section 3B: Definition of subscriber data I-WLAN domain
3GPP TS 23.234 Version 12.0.0	<i>3GPP system to Wireless Local Area Network (WLAN) interworking; System description</i>	<ul style="list-style-type: none"> • Section 6.1: Reference Model • Section 6.2: Network Elements
3GPP TS 23.402 Version 12.8.0	<i>Architecture enhancements for non-3GPP accesses</i>	<ul style="list-style-type: none"> • Section 4.1: Concepts • Section 4.3: Network Elements
3GPP TS 24.302 Version 14.4.0	<i>Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks; Stage 3</i>	<ul style="list-style-type: none"> • Section 6: UE – EPC Network protocols • Section 8: PDUs and parameters specific to the present document
3GPP TS 29.002 Version 12.7.0	<i>Mobile Application Part (MAP) specification</i>	<ul style="list-style-type: none"> • Section 6: Requirements concerning the use of SCCP and TC • Section 7.1: Terminology and definitions • Section 7.2: Modelling principles • Section 7.3: Common MAP service
3GPP TS 29.273 Version 12.7.0	<i>Evolved Packet System (EPS); 3GPP EPS AAA interfaces</i>	<ul style="list-style-type: none"> • Section 4: SWa Description • Section 5: STa Description • Section 6: SWd Description • Section 7: SWm Description • Section 8: SWx Description • Section 9: S6b and H2 Description • Section 10: Result-Code and Experimental-Result Values

Table 5: 3GPP Technical Specifications (*continued*)

3GPP TS Number	Title	Applicable Sections
3GPP TS 33.402 Version 14.2.0	<i>3GPP System Architecture Evolution (SAE); Security aspects of non-3GPP accesses</i>	<ul style="list-style-type: none"> • Section 6: Authentication and key agreement procedures • Section 7: Establishment of security contexts in the target access system • Section 8: Establishment of security between UE and ePDG • Section 9: Security for IP based mobility signalling • Section 14: Temporary identity management

WiMAX Technical Specifications

The WiMAX Forum Networking Group (NWG) maintains a repository of technical documents and specifications online at <http://www.wimaxforum.org>. You can also view the WiMAX IEEE standards, 802.16e-2005 for mobile WiMAX and 802.16-2004 for fixed WiMAX, online at <http://www.ieee.org>.

Third-Party Products

For information about configuring your Ulticom software and hardware, or your access servers and firewalls, consult the manufacturer's documentation.

Obtaining Documentation

To obtain the most current version of all Juniper Networks technical documentation, see the products documentation page on the Juniper Networks website at <https://www.juniper.net/>.

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation to better meet your needs. Send your comments to techpubs-comments@juniper.net, or fill out the documentation feedback form at <https://www.juniper.net/documentation/feedback/>. If you are using e-mail, be sure to include the following information with your comments:

- Document name
- Document part number

- Page number
- Software release version

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- **JTAC Policies**—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at <https://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf>
- **Product Warranties**—For product warranty information, visit <https://www.juniper.net/support/warranty/>
- **JTAC Hours of Operation**—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings:
<https://support.juniper.net/support/>
- Find product documentation:
<https://www.juniper.net/documentation/>
- Find solutions and answer questions using our Knowledge Base: <https://kb.juniper.net/>
- Download the latest versions of software and review release notes:
<https://support.juniper.net/support/downloads/>
- Search technical bulletins for relevant hardware and software notifications:
<https://kb.juniper.net/InfoCenter/index?page=subscriptions>, "Manage My Subscriptions"
- Open a case online in the Juniper Networks Customer Service Portal:
<https://my.juniper.net>

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool located at

<https://entitlementsearch.juniper.net/entitlementsearch/>

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Juniper Networks Customer Service Portal at <https://my.juniper.net>
- Call 1-888-314-JTAC (1-888-314-5822 – toll free in the USA, Canada, and Mexico)

For international or direct-dial options in countries without toll-free numbers, visit <https://www.juniper.net/support/requesting-support.html>

When you contact technical support, be ready to provide:

- Your Steel-Belted Radius Carrier release number (for example, Steel-Belted Radius Carrier Release 7.x).
- Information about the server configuration and operating system, including any OS patches that have been applied.
- For licensed products under a current maintenance agreement, your license or support contract number.
- A detailed description of the problem.
- Any documentation that may help in resolving the problem, such as error messages, memory dumps, compiler listings, and error logs.

1

PART

Product Overview

[Steel-Belted Radius Carrier Overview | 2](#)

Steel-Belted Radius Carrier Overview

IN THIS CHAPTER

- Introduction to Steel-Belted Radius Carrier | 2
- SBR Carrier Core Features | 3
- Management Interfaces | 7
- Optional SIM Authentication Module | 8
- Optional WiMAX Mobility Module Features | 8
- Optional Session Control Module | 10
- Optional Scripting Module | 10
- Optional Session State Register (High Availability) Module | 11
- Optional Concurrency Module | 11
- Optional 3GPP AAA Module | 11
- Licensing | 12

This chapter presents an overview of the features in the Steel-Belted Radius Carrier core. In addition, it provides an overview of the optional modules available for Steel-Belted Radius Carrier, and describes licensing. This chapter contains these topics:

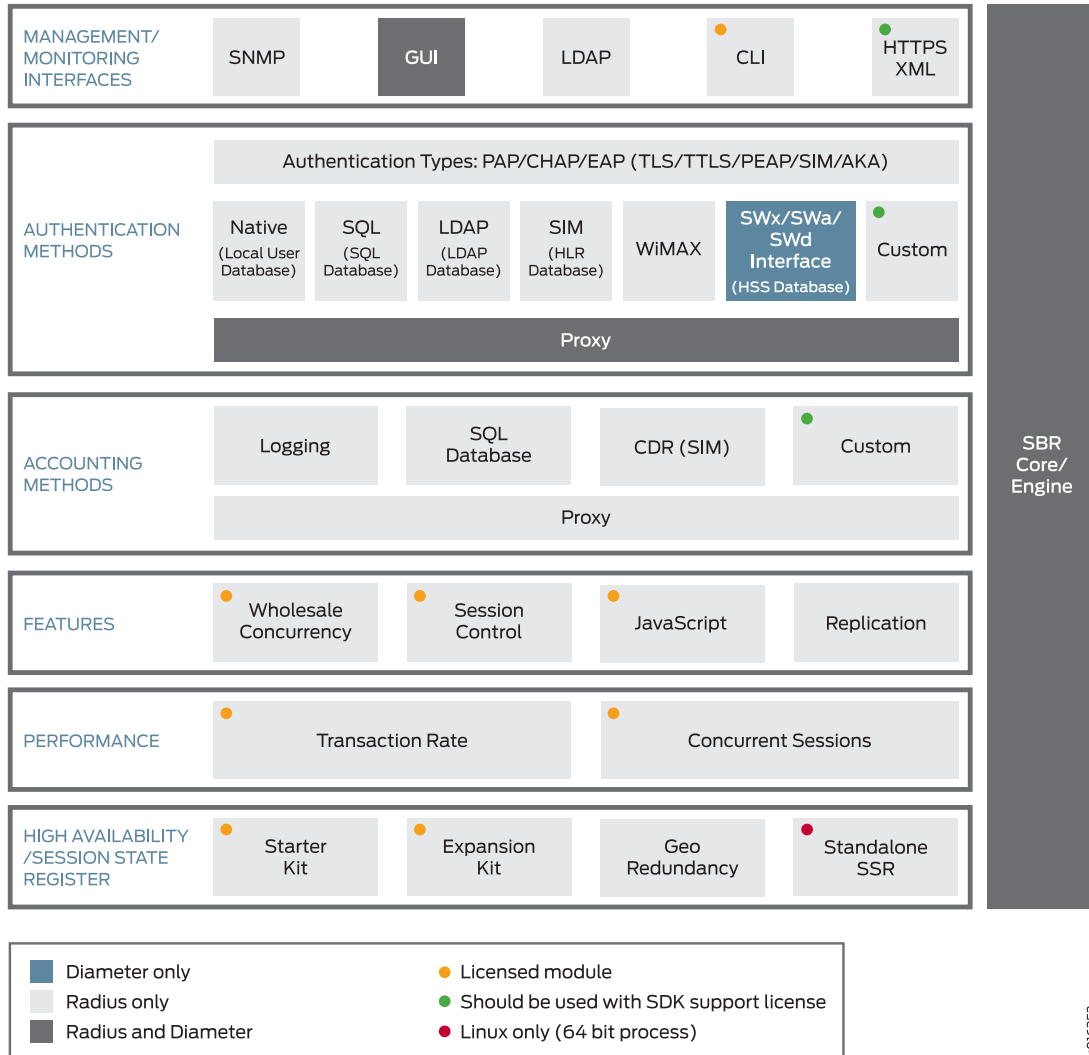
Introduction to Steel-Belted Radius Carrier

Steel-Belted Radius Carrier delivers a total authentication, authorization, and accounting (AAA) solution on the scale required by Internet service providers and carriers. It provides the power and flexibility you need to manage the delivery of enhanced services to your customers. It integrates cleanly with all components of your network operations center (NOC), from customer authentication and service delivery to the back-office accounting and billing system.

Steel-Belted Radius Carrier is an authentication, authorization, and accounting (AAA) platform that provides data services for wireline, wireless, and converged carriers. It interfaces with a wide variety of network access equipment, and authenticates remote and wireless LAN (WLAN) users against numerous back-end databases, allowing you to consolidate the administration of all your remote and WLAN users.

Figure 1 on page 3 shows the architecture of Steel-Belted Radius Carrier.

Figure 1: Steel-Belted Radius Carrier Architecture



The foundation of Steel-Belted Radius Carrier is the SBR Carrier core. All other modules are optional, license-enabled modules. The following section provides a high-level description of SBR Carrier core and each optional module.

SBR Carrier Core Features

Important features and benefits of SBR Carrier core include:

- Centralized management of user access control and security simplifies access administration.
- Flexible, powerful proxy RADIUS features enable you to easily distribute authentication and accounting requests to the appropriate RADIUS server for processing.
 - You have a choice of username format, and you can configure routing based on username decoration, Dialed Number Identification Service (DNIS), or specific attributes.
 - You can selectively modify attributes as proxy packets flow to and from Steel-Belted Radius Carrier.
 - You can specify groups of proxy target servers that handle proxy requests according to load-balancing or retry strategies—for the best performance and reliability.
- External authentication features enable you to authenticate against multiple, redundant Structured Query Language (SQL) or Lightweight Directory Access Protocol (LDAP) databases according to configurable load balancing and retry strategies, ensuring the highest level of service delivery to your users.
- Authentication against a local database enables employee access to the network.
- Flexible authentication options enable you to use your existing OS-based authentication database and external SQL/LDAP databases for remote and WLAN user authentication.
- Support for a wide variety of 802.1X-compliant access points and other network access servers ensures compatibility in your network environment.
- You can control the time periods during which each user is allowed access. An access request is granted only during a user's allowed access hours; otherwise it is refused, even if the user presents valid credentials.
- You can configure Steel-Belted Radius Carrier by using Web GUI or LDAP (either programmatically or at the command line prompt).
- You can define administrative access levels and apply them to user or group accounts on the server. Read, write, and read/write access can be applied selectively to various categories of configuration data.
- Auto-restart enables Steel-Belted Radius Carrier to restart itself automatically if it experiences a shutdown.
- Simple Network Management Protocol (SNMP) support enables you to centrally monitor Steel-Belted Radius Carrier from your SNMP console, in the same manner as you monitor other devices and services on your network. Steel-Belted Radius Carrier offers full SNMP support including SNMP traps and alarms.
- High-performance operation ensures speedy Internet access, with minimal delay for customers.
- Directed authentication and accounting features simplify the hosting of RADIUS services so Steel-Belted Radius Carrier can provide unique services for each of your customers. Incoming requests can be directed to specific authentication or accounting methods based on username decoration or DNIS.

3rd Generation Partnership Project (3GPP) Support

3rd Generation Partnership Project (3GPP) support facilitates the management of mobile sessions and their associated resources through communication with a Gateway GPRS Support Node (GGSN). 3GPP support in Steel-Belted Radius Carrier is based on the specifications given in the *Interworking between the Public Land Mobile Network (PLMN) supporting Packet Based Services and Packet Data Networks (PDN)* documentation (TS 29.061), which is available at www.3gpp.org.

The 3GPP support in Steel-Belted Radius Carrier includes support of multiple Packet Data Protocol (PDP) contexts. In order to transmit or receive General Packet Radio Service (GPRS) data, a mobile station (MS) must activate a Packet Data Protocol context (PDP). The PDP context is a set of parameters that consists of all the information required for establishing an end-to-end data connection. Multiple PDP contexts enable a single MS to access multiple services simultaneously.

Native Support for Structured Attributes

Steel-Belted Radius Carrier natively supports structured attributes that contain subattributes. Subattributes, like normal RADIUS attribute-value pair (AVPs), consist of the raw encoding of a type field (such as 1 for WiMAX-Release, within the WiMAX-Capability VSA) followed by a length value (such as 5) followed by the value of the attribute (such as 1.2).

Subattributes are values in a RADIUS packet that are not stored as a RADIUS AVP, or vendor-specific-attribute (VSA), but rather are packed with other subattributes into a RADIUS VSA. In a RADIUS packet, multiple RADIUS VSAs might contain subattributes. The RADIUS VSA, which consists of multiple subattributes, is sometimes referred to as a *structured attribute* because it contains structured data.

Earlier Steel-Belted Radius products, such as the SIM Server product and the Mobile IP Module (MIM), only interpreted AVPs and not the subattributes contained within the AVP. These earlier products included plug-ins that copied the subattributes from the AVP container and represented them in the RADIUS request as if they were received as separate AVPs. In the response, the RADIUS AVPs were reassembled from their contained subattribute values. This process was known as packet *flattening* and *unflattening*.

The dictionary mechanism of Steel-Belted Radius Carrier has been extended to allow for XML declaration of structured AVP contents. When an AVP with an associated structure definition is received, its internal subattribute values are automatically parsed and become available to any component within Steel-Belted Radius Carrier that processes RADIUS requests. Similarly, any subattribute values that are populated into the RADIUS response are formatted as part of the structured RADIUS AVP according to the same XML structure definition.

In Release 7.2 and lower-numbered releases of Steel-Belted Radius Carrier, if you used the packet flattening/unflattening method, we recommend that you migrate to using subattributes.

Adding NAS Location Information to Access-Requests

Steel-Belted Radius Carrier core provides an attribute handling feature that allows you to add network access server (NAS) location information to proxied Access-Request messages.

When a mobile device is outside the area of its provider, it roams by sending the request to a local foreign AAA (FAAA) server that is owned by another provider. Service providers might require the location of the mobile device requesting access to their network.

You can configure an Access-Request to include the location of the NAS through which the proxied request was processed. Because the NAS is geographically near the mobile device, it closely approximates the location of the mobile device.

For proxied requests, Steel-Belted Radius Carrier can perform a lookup to find a NAS location based on a particular attribute. Steel-Belted Radius Carrier can perform a query to find the value of the attribute that identifies the NAS location. The NAS location is then added to the Access-Request, which is sent to the service provider's home AAA (HAAA) server.

Support for Additional EAP Authentication Protocols

The license for the Steel-Belted Radius Carrier core module includes support for the following EAP authentication protocols: TLS, TTLS, and PEAP. Before release 7.0 these authentication protocols required an additional license.

Statistics and Reporting Capabilities

Steel-Belted Radius Carrier collects and displays summary statistics for authentication, accounting, and proxy forwarding transactions. You can also use statistics to see how long Steel-Belted Radius Carrier has been running and to display a list of the users currently connected through a NAS or tunnel.

Steel-Belted Radius Carrier also includes a user-configurable logging and reporting function that generates log files, which record session statistics for authentication and accounting. Accounting reports can be used for billing, system diagnosis, and usage planning. Using authentication reporting, you can control how and what Steel-Belted Radius Carrier logs and reports for authentication requests including successful authentications, authentication rejections, requests received from unknown clients, and requests that used invalid shared secrets.

You can also control dilutions and thresholds for Steel-Belted Radius Carrier events used to communicate failures, warnings, and other information.

Management Interfaces

Steel-Belted Radius Carrier and its functions can be managed using a number of management interfaces. Following is a brief description of each management interface and a reference to where you can find specific details on each interface.

Web GUI

The front-end interfaces for the Steel-Belted Radius Carrier is the Web GUI. The Web GUI is an application that enables you to configure settings for SBR Carrier using a Web browser. In minutes, you can set up new users, alter standard profiles, or configure new network access servers from any computer on the network. Web GUI is used to configure most of the functions of Steel-Belted Radius Carrier. Refer to the particular function you want to configure, for specific details on how to use Web GUI to configure it. For more information about the Web GUI, see [“Using Web GUI” on page 16](#).

LDAP Configuration Interface (LCI)

The LDAP Configuration Interface (LCI) provided by Steel-Belted Radius Carrier, consists of an LDAP interface in the Steel-Belted Radius Carrier server and an LDAP virtual schema. The LDAP virtual schema presents the structure of the Steel-Belted Radius Carrier session storage in a manner that can be understood by the LDAP client utilities. The LCI uses the virtual schema to retrieve entries from the session storage. For more information about the LCI, see [“Using the LDAP Configuration Interface” on page 497](#).

Command Line Utility

A command line utility can be used to view sessions and issue Change of Authorization and Disconnect Message (CoA/DM) requests. For more information about the command line utility, see [“Using the Command Line Utility to Manage and Control Sessions” on page 730](#).

XML/HTTPS Interface

Steel-Belted Radius Carrier includes an application programming interface (API) which is a proprietary XML request/response interface. This interface is used by Web GUI, command line utility, or a custom developed client management application to issue Change of Authorization and Disconnect Message (CoA/DM) requests. For more information, see [“XML over HTTPS Interface” on page 760](#).

SNMP

SNMP support enables you to monitor Steel-Belted Radius Carrier from your SNMP console, in the same manner as you monitor other devices and services on your network. Steel-Belted Radius Carrier SNMP

support includes SNMP traps and alarms. For more information, see [“Simple Network Management Protocol” on page 488](#).

Optional SIM Authentication Module

The optional SIM Authentication module enables you to provide IP-based services such as public WLAN access, and Unlicensed Mobile Alliance (UMA) and Femtocell access to your subscribers, while leveraging your existing customer care and authentication infrastructure. Appropriate for Global System for Mobile Communications (GSM) infrastructures, the optional SIM authentication module provides AAA services for 802.1X and non-802.1X hotspots and Unlicensed Mobile Access (UMA) networks, enabling several new service offerings. You can offer secure hotspot access via 802.1X and Extensible Authentication Protocol–Subscriber Identity Module (EAP-SIM) or Extensible Authentication Protocol–Authentication and Key Agreement (EAP-AKA) user authentication. Finally, the SIM authentication module extends mobile services over IP access networks for UMA environments, providing the same mobile identity on unlicensed wireless networks as on mobile networks, and enabling roaming and handover between networks.

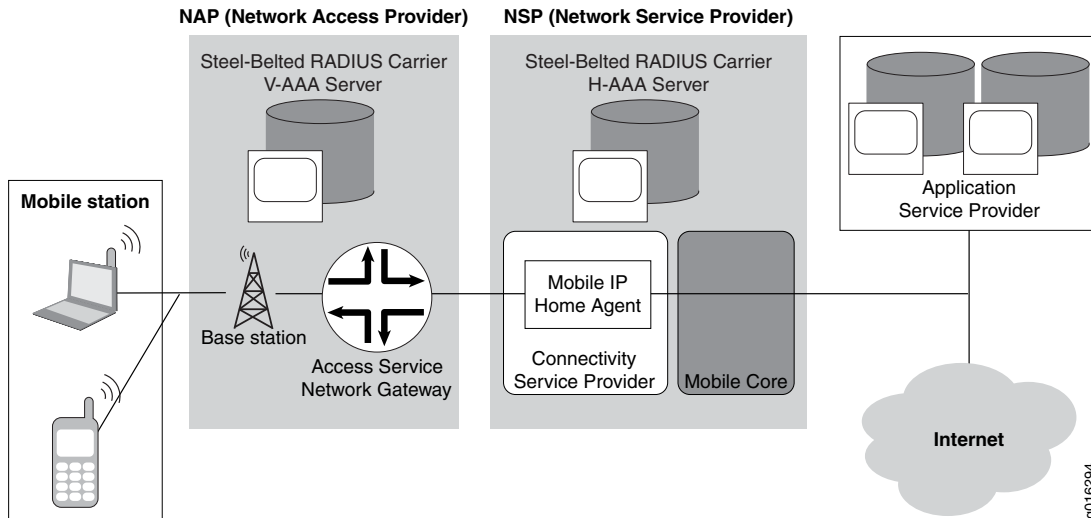
Optional WiMAX Mobility Module Features

Worldwide Interoperability Microwave Access (WiMAX) is an 802.16-based implementation of a standard broadband wireless access technology used for applications that include mobile broadband, fixed broadband connections, hotspot and cellular backhaul, and high speed enterprise connectivity for businesses.

The main components of WiMAX include ([Figure 2 on page 9](#)):

- Mobile Station—The device used by the end user to access the network.
- Access Service Network Gateway (ASN-GW)—Located at the Network Access Provider, the ASN GW provides radio resource management, access control, and mobility management services.
- Mobile IP Home Agent (MIP-HA)—Located at the Connectivity Service Provider (CSP) the MIP-HA acts as mobility anchor, providing access to IP-based services and acting as the policy enforcement point for policy management applications.
- AAA Server—Located at the CSP, the AAA server provides authentication and authorization of subscribers, devices (or both), as well as mobility management.

Figure 2: WiMAX Reference Architecture



The optional WiMAX Mobility module enables Steel-Belted Radius Carrier to act as either the home AAA (HAAA) or visited AAA (VAAA) as shown in [Figure 2 on page 9](#). It also handles EAP-authentication over RADIUS, manages mobility, and is responsible for processing accounting requests for both post and prepaid billing.

The functions the AAA server is responsible for in a WiMAX network include:

- Network attachment—Securely attach a user or device (or both) to the network, and manage its authentication keys throughout the session lifetime.
- Mobility management—Manage a user's mobility throughout the session lifetime.
- Resource Management—Assign and manage a user's network resources
 - User IP addresses
 - Home agent assignment
- Quality of Service—Manage and assign a user's WiMAX QoS flows and authorize their activation
- Billing—Provide user/session and QoS flow-based accounting (service session) and reconciliation
- Roaming—Act as a visited or home AAA in roaming scenarios. Ensure proper authentication and billing

The WiMAX mobility module of Steel-Belted Radius Carrier supports the following features:

- AAA-generated cryptographic keys generation, management, and distribution
- EAP authentication methods and EAP-derived cryptographic keys
- WiMAX VSA support, including Continued and Structured attributes
- Capabilities negotiation
- Access-Request categorization by client type: ASN-GW, home agent, DHCP server or other

- Home agent and DHCP server assignment
- Post-Paid Accounting, including per-flow and continued sessions
- Reauthentication (authenticate-only service type) without handover
- Mobility and handover, including reauthentication and accounting with handover

For more information, see [“Configuring the WiMAX Mobility Module” on page 580](#).

Optional Session Control Module

The optional Session Control module enables you to make changes to active subscriber sessions without requiring the network access server (NAS) to initiate the change. For example, you may want to terminate an active user’s session by issuing a Disconnect Message (DM) request to the NAS, or you may want to modify the authorization level of an active user’s session issuing a Change of Authorization (CoA) request to the NAS. For example, as a service provider, you may be required to provide legal organizations with voice and data intercept capabilities as mandated by law. These might include access to private communications between organizations or individuals such as phone calls, email, VoIP, or instant messaging. These legal intercept capabilities can be performed by issuing a CoA request.

Using the Session Control module, you can customize the CoA/DM requests you want to support in your network. This powerful capability allows you to define *actions* that can be invoked on active sessions such as disconnecting an active session, increasing the bandwidth of an active session, or any other action you want to define.

You can control sessions using Web GUI, a command line utility, or you can develop your own client management application to interface with the Steel-Belted Radius Carrier CoA/DM XML interface.

For more information, see [“Overview of the Optional Session Control Module” on page 700](#).

Optional Scripting Module

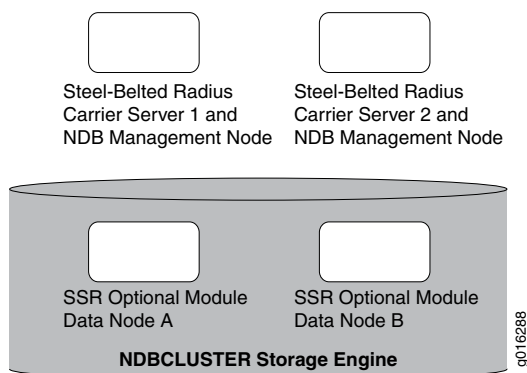
An optional JavaScripting module is available that enables you to incorporate scripts into your Steel-Belted Radius Carrier configuration. It supports fine-tuning of Steel-Belted Radius Carrier server settings and implementation of custom request processing logic. You can use scripts to configure Steel-Belted Radius Carrier to evaluate complex decision logic and manipulate RADIUS request data objects in ways that cannot be expressed through settings in the standard Steel-Belted Radius Carrier initialization files.

Steel-Belted Radius Carrier scripts are written in JavaScript, an industry standard scripting language with a powerful, object-based syntax.

Optional Session State Register (High Availability) Module

The Steel-Belted Radius Carrier Session State Register (SSR) module implements a stateless high-availability AAA platform. Multiple servers of different types (data, management, and SBR Carrier) perform certain aspects of SBR Carrier operation. The servers collaborate to share a common session database and a common IP address pool, and to provide a high level of redundancy. The common shared resources can be accessed simultaneously by up to 20 SBR Carrier servers.

Figure 3: Basic “Starter Kit” Steel-Belted Radius Carrier Session State Register Cluster



Optional Concurrency Module

The optional Concurrency module works in the Session State Register environment and provides tools that can limit the number of active connections on a per-user per-cluster basis on any attribute or realm. Users can also be grouped based on attributes or realms.

Optional 3GPP AAA Module

The 3GPP AAA module enables the Diameter functionality in SBR Carrier to support non-3GPP network services. SBR Carrier performs EAP-AKA and EAP-AKA' based authentication, subscription authorization, and accounting in non-3GPP networks. SBR Carrier converts RADIUS messages received from non-3GPP networks to Diameter messages and forwards Diameter messages to a Home Subscriber Server (HSS). When the HSS replies with Diameter messages, SBR Carrier converts them to RADIUS messages and replies to the non-3GPP client.

Licensing

Steel-Belted Radius Carrier uses five types of licensing.

- The base server license (SBR-CAR-AAA) delivers all core functionality on a single server and supports up to 50,000 concurrent sessions.
- Additional concurrent user licenses are available to increase the number of concurrent sessions that a server may handle. These are cumulative licenses that enhance the base server license. For example, if you start with a base server license (which supports 50,000 sessions) on each server in a cluster and purchase an additional license that also supports 50,000 sessions, you can apply this additional license to all servers in the cluster to increase the total number of sessions of the server to 100,000.

NOTE:

- For a cluster, the additional license should be applied to each of the front ends to increase the total session capacity of the cluster.
- For a standalone system, the additional license can be applied only to the single SBR server.

In both the standalone system and the cluster, each server individually monitors the number of concurrent active sessions every hour. If the number of concurrent active sessions exceeds the license count, then the server generates an SNMP trap and writes a warning message to the log.

The format of the message written to the log is **WARNING: %d active sessions exceed the license limit of %d sessions. Please upgrade your SBR license in order to remain in compliance with your license agreement.** Here, the first %d indicates the number of active sessions and the second %d indicates the maximum number of concurrent sessions set by the license.

NOTE: The message is written to the log regardless of the LogLevel set in the **radius.ini** file.

Licenses are available for different numbers of additional sessions, up to 2,000,000:

- Add 50,000 concurrent sessions (SBR-CAR-ADD-50K)
- Add 100,000 concurrent sessions (SBR-CAR-ADD-100K)
- Add 250,000 concurrent sessions (SBR-CAR-ADD-250K)
- Add 500,000 concurrent sessions (SBR-CAR-ADD-500K)
- Add 1,000,000 concurrent sessions (SBR-CAR-ADD-1M)
- Add 2,000,000 concurrent sessions (SBR-CAR-ADD-2M)
- Optional add-on modules, except Session State Register, are licensed on a per-server basis and requires that the server already have a base server (SBR-CAR-AAA) license. Currently available modules are:

- Optional WiMAX Mobility Module (SBR-CAR-WMM).
- Optional JavaScripting (SBR-CAR-JSC).
- Optional SIM Authentication Module (SBR-CAR-SIM).
- Optional Session Control Module (SBR-CAR-SCM)
- Optional 3GPP AAA Module (SBR-CAR-DIA)

NOTE: The preceding optional add-on modules require unique licenses per front end.

- The Optional Session State Register (high availability) Module has special license offerings.
 - Three kinds of licenses are available for the Steel-Belted Radius Carrier Session State Register cluster starter kit:
 - Regular starter kit license (SBR-SSR-START)—Provides two data node licenses and two management node licenses. It does not impose any restriction on the expansion kit or concurrent sessions. One SBR-SSR-START license is required per cluster, and the same license must be applied to each front end in the cluster.
 - Restricted cluster session license (SBR-SSR-LIMITED)—Provides two data node licenses and two management node licenses. It limits the total number of concurrent sessions to 100,000 regardless of the additional concurrent session licenses you applied. For example, if you start with a base server license (which supports 50,000 sessions) and purchase an additional license that supports 100,000 sessions, the restricted cluster session license allows you to use only 100,000 sessions and ignores the additional 50,000 sessions. The restricted cluster session license does not allow you to add any expansion kit licenses. One SBR-SSR-LIMITED license is required per cluster, and the same license must be applied to each front end in the cluster.
 - Upgrade cluster license (SBR-SSR-UPG)—Is available to enable you to upgrade from a restricted cluster to a regular cluster. It only removes the restriction on the number of concurrent sessions and enables the addition of an expansion kit license. One SBR-SSR-UPG license is required per front end.
 - The Steel-Belted Radius Carrier Session State Register cluster expansion kit (SBR-SSR-EXP) provides two additional data node licenses. One expansion kit may be added to a starter kit, so the SSR cluster can grow beyond the Starter Kit's two data node licenses to include four data nodes. One SBR-SSR-EXP license is required per cluster, and the same license must be applied to each front end in the cluster.
 - Additional single Steel-Belted Radius Carrier Session State Register management node licenses (SBR-SSR-EXP)(SBR-SSR-MNGMT) are available. The SSR cluster can grow beyond the Starter Kit's two combination management node licenses to include a third management node. One SBR-SSR-MNGMT license is required per front end.
 - The optional Concurrency and Wholesale Module (SBR-CAR-CWM) license enables you to manage user concurrency across multiple servers connected to the same session storage cluster. Concurrency

can be managed based on username, as well as any attribute. One SBR-CAR-CWM license is required per front end.

- The optional transaction license controls the transaction rate when no session license is applied. This license is useful when SBR is not used to manage sessions—for example, when SBR is used as a proxy server. By default, a rate limit of 100 transactions per second (TPS) exists in the absence of a session license.

Licenses are available for additional transactions. You can add the following numbers of additional transactions:

- Up to 500 transactions (SBR-CAR-TPS-500)
- Up to 1000 transactions (SBR-CAR-TPS-1000)
- Up to 2000 transactions (SBR-CAR-TPS-2000)
- Up to 5000 transactions (SBR-CAR-TPS-5000)
- Up to 10,000 transactions (SBR-CAR-TPS-10000)
- Unlimited transactions (SBR-CAR-TPS-UNLIM)

Steel-Belted Radius Carrier generates SNMP traps and stores warning messages in the log file when the rate limit of TPS is exceeded.

NOTE:

- The default rate limit of 100 TPS will not add up, when any of the above transaction licenses are applied.
- By default, clusters support unlimited TPS, hence the optional transaction license is not required for the cluster environment.

2

PART

Web GUI Overview

Using Web GUI | 16

Using Web GUI

IN THIS CHAPTER

- Running the Web GUI | 16
- Navigating in the Web GUI | 20
- Adding License Keys | 29
- Displaying Version Information | 31
- Closing the Web GUI | 31

Web GUI is an application that enables you to configure settings for SBR Carrier using a Web browser. This chapter presents an overview of how to use the Web GUI to configure SBR Carrier. This chapter contains the following topics:

NOTE: Administrators making large-scale changes to the SBR Carrier database might prefer to use the LDAP command line interface. See [“Using the LDAP Configuration Interface” on page 497](#).

Running the Web GUI

You launch the Web GUI by running a Web browser on your management workstation, and opening a connection to the SBR Carrier server you want to configure. The Web GUI is designed to support any Web 2.0 browser. For a list of tested browsers, see [“Tested Browsers” on page 19](#).

A Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS) connection is established between the Web browser in the management workstation and the SBR Carrier server. This connection is used to communicate configuration data to the server.

NOTE: The SBR Carrier server must be running in order to use Web GUI. Executing the **sbrd start** command also starts the Web server.

Like other secure websites using HTTPS, SBR Carrier provides a Secure Sockets Layer (SSL) certificate to prove its identity, and requires credentials when accessing the server. The SSL certificate can be autogenerated or you can provide a custom SSL certificate by running the configure script or during initial configuration of SBR Carrier.

NOTE: Digital Signature Algorithm (DSA) certificates are not supported.

Pluggable authentication module (PAM) settings:

PAM provides an extensible mechanism for user authentication. It provides one or more of the following authentication and security services:

- Authentication—These service modules are used to authenticate access to an account or service and set up user credentials.
- Account management—These service modules determine the validity of the current user account. For example, an account management module could check password or account expiration or enforce access-time restrictions.
- Session management—These service modules are responsible for setting up and terminating login sessions.
- Password management—These service modules enforce password strength and previous-use rules and perform authentication updates.

The **Method** parameter in the [Settings] section of the **access.ini** file controls the database against which the user is authenticated for access. See the *SBR Carrier Reference Guide* for more information.

To log in to a SBR Carrier server:

1. Open a browser connection to the SBR Carrier server you want to administer.

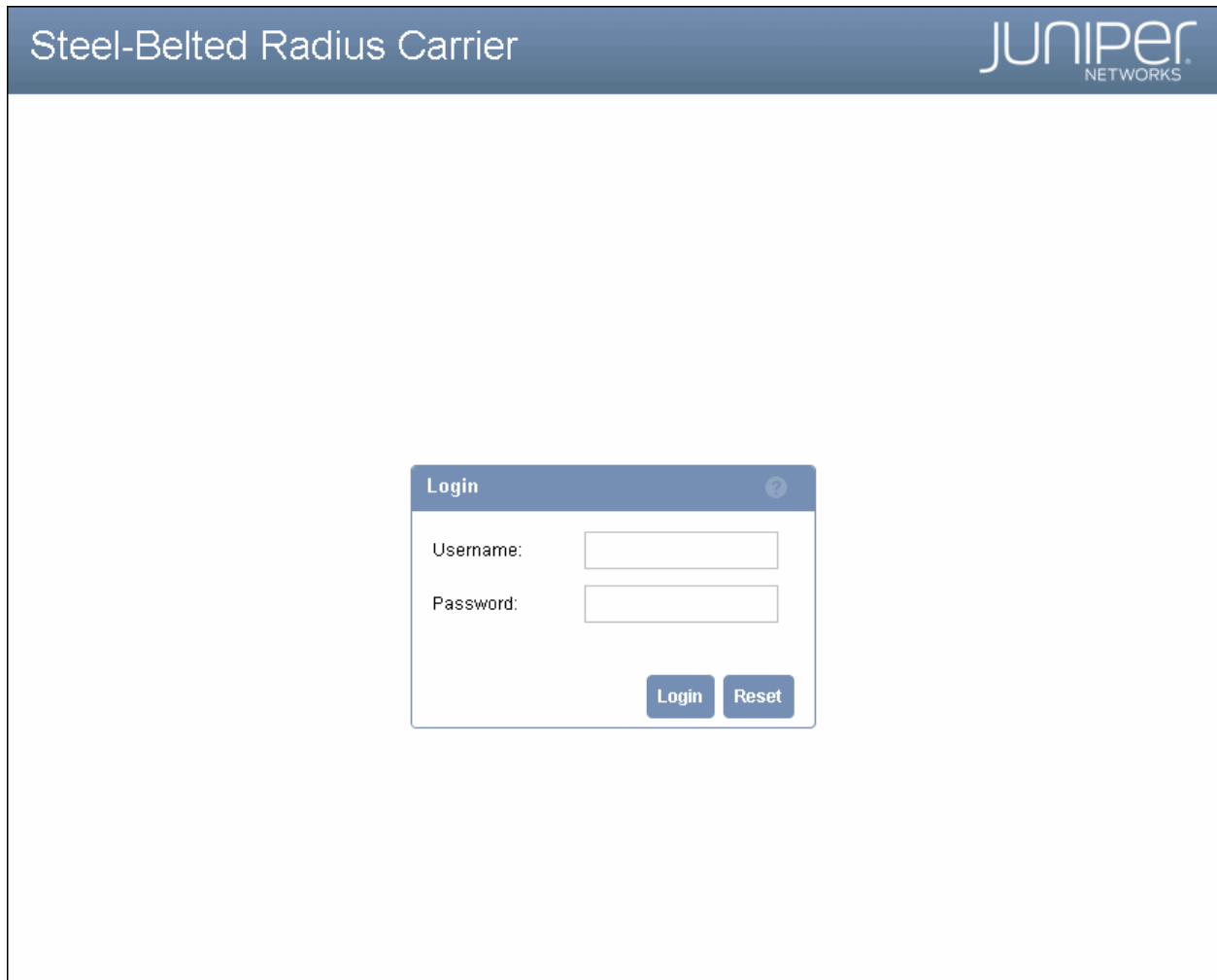
- To administer a SBR Carrier server running on your local host, enter **https://localhost:2909**, where the port assignment of 2909 is the SBR Carrier's default TCP port for administration connections.
- To administer a SBR Carrier server running on a remote host, enter **https://server:2909**, where the port assignment of 2909 is the SBR Carrier's default TCP port for administration connections. For example:

`https://192.168.24.15:2909/`

NOTE: Make sure that you access the Web GUI using HTTPS instead of HTTP.

The **Login** page ([Figure 4 on page 18](#)) appears.

Figure 4: Login Page



Steel-Belted Radius Carrier

JUNIPER NETWORKS

Login

Username:

Password:

Login Reset

NOTE: You can use the **https.port** parameter in the **https.ini** file located in the **/opt/JNPRsbr/radius/website/webserver/jetty/start.d/** path to change the default port.



CAUTION: Using a non-default port or modifying other parameters in the **/opt/JNPRsbr/radius/website/webserver/** directory incorrectly could cause communication problems between the Web GUI and the SBR Carrier server.

2. Enter your administrator username in the **Username** field.
3. Enter your login password in the **Password** field.
4. Click **Login**.

When you click **Login**, Web GUI establishes an HTTPS connection with the local or remote server. The Web GUI displays an error message if the connection cannot be established.

NOTE: If a timeout occurs, verify that the Web browsers are running on the target server using the **sbrd status** command and that it is listening on the administration port you entered in the URL; that the port is not blocked.

Web GUI verifies that the username you entered exists in the **access.ini** file. If the username is found, Web GUI validates the password you entered against a local or remote password database.

When you connect to a server, the **Home** page ([Figure 5 on page 21](#)) lists status of the running server, IP addresses, available authentication methods, license information, and any initialization errors that might have occurred.

Tested Browsers

Web GUI can be launched in different browsers across different platforms. [Table 6 on page 19](#) lists the tested browser versions and the operating systems.

NOTE: Java 1.8.0 or a later version is required to be installed in the server to access the Web GUI.

When you upgrade from an earlier SBR Carrier release to the current release, clear your browser's cache before launching the Web GUI.

Table 6: Web GUI—Tested Browsers

Browser	Version	OS
Google Chrome	36 and later	Windows/UNIX
Internet Explorer	9 and later	Windows
Mozilla Firefox	31 and later	Windows/UNIX
Opera	23 and later	Windows/Mac
Opera	12 and later	UNIX

NOTE: While using Internet Explorer to launch the Web GUI, you may face the following problem:

If a hostname is resolved manually by adding an entry in the **hosts** file under **C:\Windows\System32\drivers\etc**, then Internet Explorer treats any websites from the hostname as an intranet site and applies the corresponding intranet settings. By default, the **Display intranet sites in Compatibility View** check box under **Compatibility View Settings** is enabled, which causes the invisibility of GUI webpages.

It is recommended to clear the **Display intranet sites in Compatibility View** check box for viewing the webpages properly. But, this setting might also impact other intranet sites.

If you still want to view other intranet sites in the Compatibility View but not the Web GUI, set the **Document Mode** setting under **Developer Tools** to the highest version number instead of clearing the **Display intranet sites in Compatibility View** check box, and then relaunch the Web GUI.

Navigating in the Web GUI

This section describes how to use the Web GUI menus and pages.

Web GUI Menus

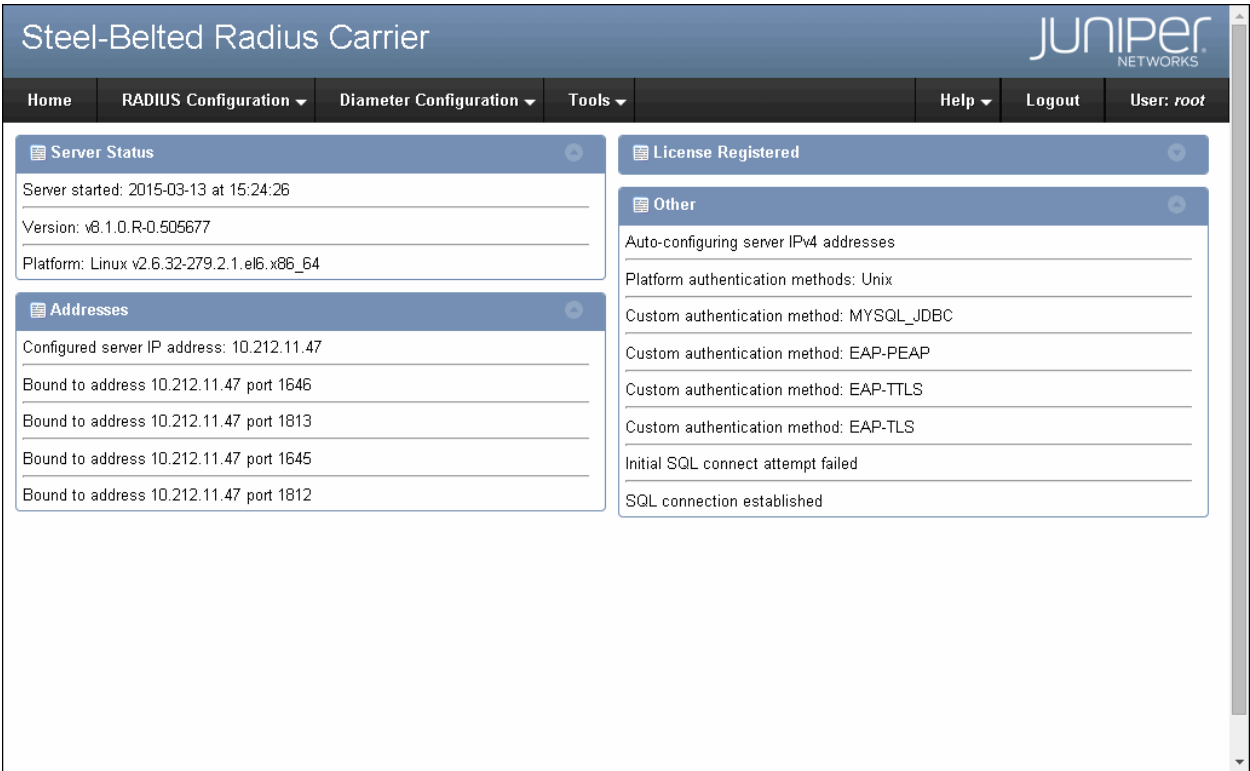
The Web GUI has six menus: Home, RADIUS Configuration, Diameter Configuration, Tools, Help, and Logout.

NOTE: The **Diameter Configuration** menu is available only if you have installed a valid Diameter license.

Home Menu

The **Home** menu displays the **Home** page ([Figure 5 on page 21](#)), which lists status of the running server, IP addresses, available authentication methods, license information, and any initialization errors that might have occurred.

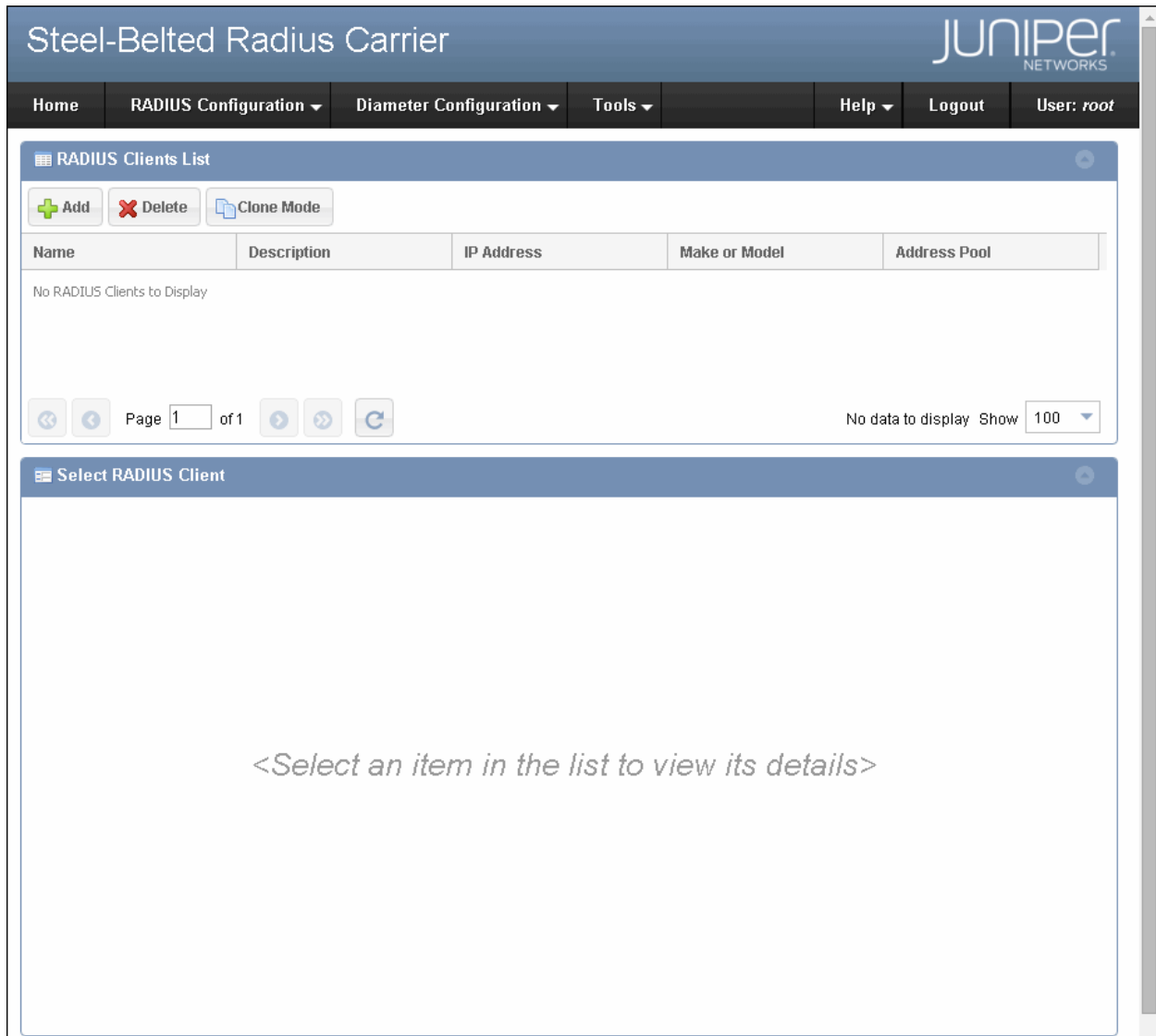
Figure 5: Web GUI Home Page



RADIUS Configuration Menu

You use the entries in the **RADIUS Configuration** menu to configure RADIUS operations and display statistics. For example, to configure RADIUS clients and client groups, select **RADIUS Configuration > RADIUS Clients**. This displays the **RADIUS Clients List** page (Figure 6 on page 22).

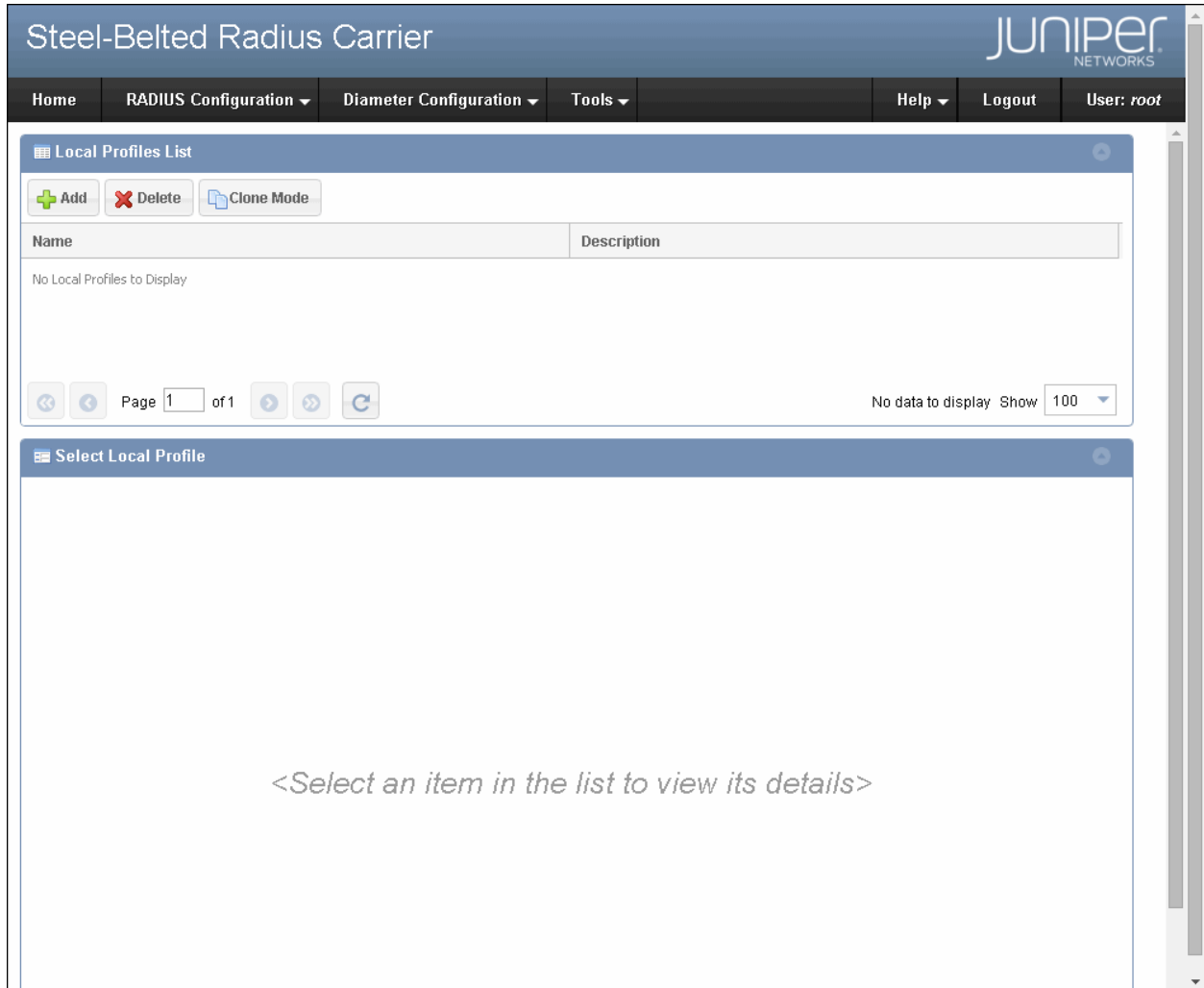
Figure 6: RADIUS Clients List Page



Diameter Configuration Menu

You use the entries in the **Diameter Configuration** menu to configure Diameter operations and display Diameter statistics. For example, to configure local profiles, select **Diameter Configuration > Policy > Local Profiles**. This displays the **Local Profiles List** page (Figure 7 on page 23).

Figure 7: Local Profiles List Page



Tools Menu

Table 7 on page 23 describes the functions of each entry in the **Tools** menu in the Web GUI.

Table 7: Tools Menu Options

Menu Entry	Function
License	Opens the License Registration page, which enables you to add a license string for your SBR Carrier software. For more information, see "Adding License Keys" on page 29.

Table 7: Tools Menu Options (*continued*)

Menu Entry	Function
Import	<p>Opens the Browse Import File page, which enables you to import information from an XML file into the SBR Carrier database.</p> <p>The Import Configuration page is described in “Importing and Exporting Data” on page 981.</p> <p>NOTE: You cannot import Diameter configuration information to the SBR Carrier database.</p>
Export	<p>Opens the Export Configuration page, which enables you to export selected information from the SBR Carrier database to an XML file.</p> <p>The Export Configuration page is described in “Importing and Exporting Data” on page 981.</p> <p>NOTE: You cannot export Diameter configuration information to the XML file.</p>
Backup/Restore	<p>Opens the Backup/Restore page, which enables you to back up the SBR Carrier database to an archive file or restore SBR Carrier from an archive file.</p> <p>NOTE: You cannot back up or restore Diameter configuration information.</p>

Help Menu

[Table 8 on page 24](#) describes the functions of each entry in the **Help** menu in the Web GUI.

Table 8: Help Menu Options

Menu Entry	Function
Contents	Opens the online help for the Web GUI.
About	Displays the version information of SBR Carrier. For more information, see “Displaying Version Information” on page 31 .

Logout Menu

The **Logout** menu terminates your connection to the SBR Carrier server.

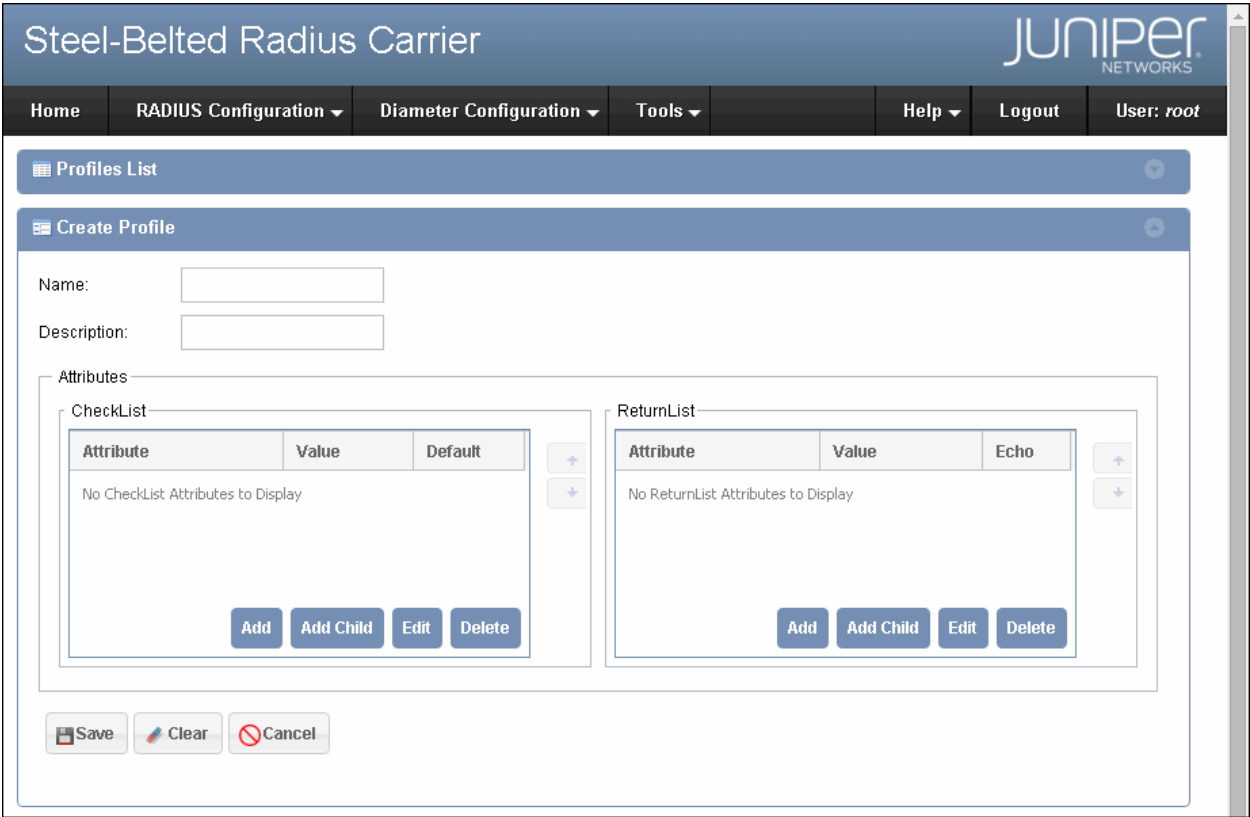
Web GUI Pages

This section summarizes how to use Web GUI pages and controls.

Adding an Entry

To add an entry to the SBR Carrier database, open the appropriate Web GUI page and click the **Add** button. The Web GUI page displays a **Create** pane. [Figure 8 on page 25](#) displays a sample **Create Profile** pane.

Figure 8: Sample Create Pane





Every object of the same type must have a unique name. If the name you assign to an item is already being used by another item of the same type, the Web GUI displays a warning.

[Table 9 on page 25](#) describes the functions of common GUI elements available on the Web GUI pages.

Table 9: Common Elements on Web GUI

GUI Element	Function
Add	Adds an object to the SBR Carrier database.
Edit	Edits an existing object in the SBR Carrier database. NOTE: Active only when an object is selected in the active page.
Delete	Deletes an existing object from the SBR Carrier database.

Table 9: Common Elements on Web GUI (continued)

GUI Element	Function
Clone Mode	Enables the clone feature, which allows you to reuse an existing object and make the required changes to it.
Save	Saves any configurations or changes you have made to the server settings.
<	Displays the previous page entries.
>	Displays the next page entries.
<<	Displays the first page entries.
>>	Displays the last page entries.
	Refreshes the displayed list of items in the Web GUI page.
Page	Displays the entries available on the entered page number.
Show	Select the number of entries to be displayed in a page.
Reset	Restores the default values for controls in the active page.
Load	<p>Loads specific values (such as attributes, called station IDs, and IP addresses) for the cloned object from other objects that exist in the active page.</p> <p>NOTE: The Load button is available only when the clone feature is enabled.</p>
Clear	Clears all the values configured for controls in the active page.
Cancel	Cancels the action and exits from the active pane or dialog box.
	<p>Indicates that an error has occurred in the active page due to one of the following reasons:</p> <ul style="list-style-type: none"> • An invalid value is entered in the highlighted field. • A mandatory field is not filled. <p>You cannot save the changes unless the errors are corrected. You may move your cursor over the red-highlighted warning symbol for more details.</p>

Editing an Entry

To edit an existing entry in the SBR Carrier database, open the appropriate Web GUI page and select the entry you want to change. The Web GUI page displays the settings configured for the selected entry in a **Selected** pane. Depending on the page that you are working in, you may use a **Save** or **OK** button to save your changes.

While editing an existing entry, you cannot change the name associated with the entry. To change the name of an entry, use the **Clone** pane. For more information about the **Clone** pane, see [“Cloning an Entry” on page 27](#).

Cloning an Entry

The clone feature enables you to reuse an existing entry and make the required changes to it instead of creating a new entry from scratch. To clone an existing entry:

1. Open the appropriate Web GUI page and click the **Clone Mode** button.

A message stating that clone mode is enabled pops up.

NOTE: When the clone feature is enabled, the **Add** and **Delete** buttons are disabled. If you want to disable the clone feature when it is enabled, you need to click the **Clone Mode** button again.

2. Select the entry you want to clone.

The Web GUI page displays the settings configured for the selected entry in a **Clone** pane.

[Figure 9 on page 28](#) displays a sample **Clone Profile** pane.

Figure 9: Sample Clone Pane

The screenshot shows the Juniper Steel-Belted Radius Carrier web interface. The top navigation bar includes 'Home', 'RADIUS Configuration', 'Tools', 'Help', 'Logout', and 'User: root'. The main content area is divided into two panes. The top pane, 'Profiles List', shows a table with one entry: 'PROFILE1' with description 'Example profile'. The bottom pane, 'Clone Profile: PROFILE1', contains fields for 'Name' (PROFILE1) and 'Description' (Clone of PROFILE1). Below these are two sections: 'Attributes' and 'ReturnList'. Each section contains a table with columns for 'Attribute', 'Value', and 'Default' (or 'Echo'). Both tables are currently empty, displaying 'No CheckList Attributes to Display' and 'No ReturnList Attributes to Display' respectively. At the bottom of each table are buttons: 'Load', 'Add', 'Add Child', 'Edit', and 'Delete'. The bottom of the 'Clone Profile' pane has 'Save', 'Reset', and 'Cancel' buttons.

3. Edit the name of the entry and the settings you want to change.

The **Load** button is available for some configurations (such as attributes, called station IDs, and IP addresses). You can use the **Load** button to load specific values for the clone entry from other entries that exist in the active page.

4. Click **Save** to save your changes. Some settings of the same entry type must be unique. If any one of those settings you configured for the duplicate entry is already being used by the original entry or another entry of the same type, the Web GUI displays a warning.

A confirmation dialog box stating “Would you like to delete the existing object” is displayed.

5. Click **Yes** if you want to delete the original entry or click **No** to skip deleting the original entry.

The duplicated entry is added to the corresponding Web GUI page.

Resizing Columns

You can resize columns in a Web GUI page by dragging the column header boundary to the left or right.

Changing Column Sequence

You can change the sequence of columns in a Web GUI page by dragging the column headers left and right.

Sorting Information

By default, entries in Web GUI pages are sorted by names. You can sort the entries in ascending or descending order by clicking a column header.

To sort a table by more than one column (for example, sort first by address pool and then by IP address), click the least-significant column (IP Address), and then click the more significant columns (Address Pool).

Searching Entries

You can search for specific entries by specifying certain criteria to display only the entries that match the criteria. To find a specific entry in a Web GUI page, move your cursor on the column header and click the **Down** arrow. Then, move your cursor over the **Filters** check box and enter the searching criteria in the text box. The Web GUI page displays the entries that match the entered criteria. Clear the **Filters** check box after finding the specific entries.

Hiding or Restoring Columns

To hide or restore columns in a Web GUI page, move your cursor on the column header and click the **Down** arrow. Then, move your cursor over **Columns** and select or clear the corresponding check box of the column name.

Adding License Keys

Depending upon your purchasing arrangements, your SBR Carrier software may require a new license key at some point after its initial installation.

If you are provided with a new license key, you can add the key to an existing SBR Carrier installation as follows:

1. Start the Web GUI program and connect to your SBR Carrier server.
2. Select **Tools > License**.

The **License Registration** page ([Figure 10 on page 30](#)) appears.

Figure 10: License Registration Page

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

License Registration

License String:

Register Clear

Existing Licenses Registered

3. Enter the license key in the **License String** field and click **Register**.

If the license key you have entered is invalid, the server displays an error message. If this occurs, click **OK** in the message dialog box and enter the correct license key.

4. After you enter a valid license key, the server displays a confirmation message and reminds you that you must restart the server. Click **OK**.

NOTE: The server does not restart itself automatically after a new license key is added. You must restart SBR Carrier manually to activate the new license key. See [“When and How to Stop and Restart Steel-Belted Radius Carrier” on page 974](#).

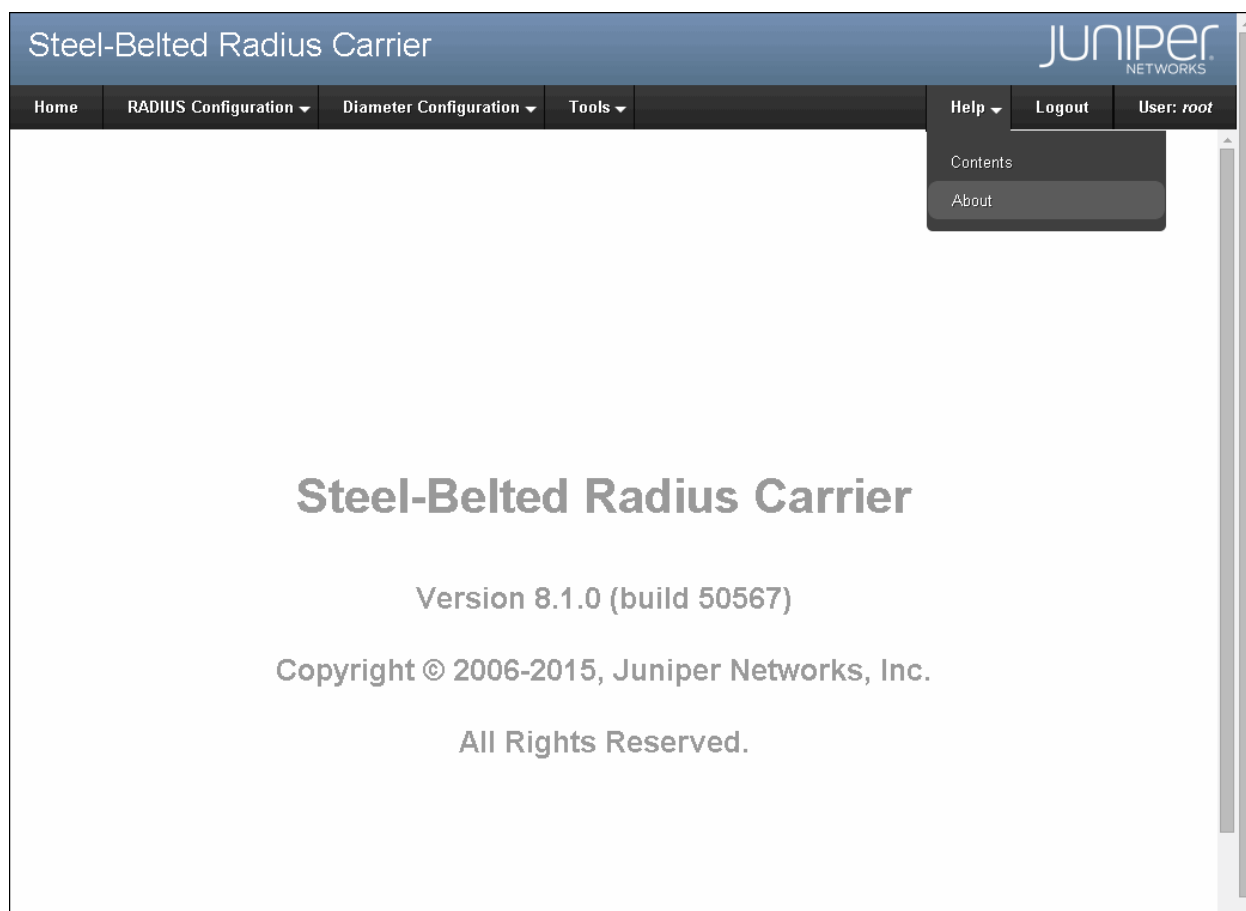
Be sure to refresh and log in again to the Web GUI running on the management workstation to reflect the changes.

NOTE: The SBR Carrier audit log does not record an entry when you enter license keys through Web GUI.

Displaying Version Information

To identify the current version of SBR Carrier for which the Web GUI is launched (Figure 11 on page 31), select **Help > About**.

Figure 11: Displaying Web GUI Version Information



Closing the Web GUI

To close the Web GUI, click the **Logout** menu and then close the browser.

Closing the Web GUI has no impact on the SBR Carrier daemon.

3

PART

RADIUS Operations

[RADIUS Basics | 34](#)

[Administering RADIUS Clients and Client Groups | 104](#)

[Administering RADIUS Location Groups | 114](#)

[Administering Users | 120](#)

[Administering Profiles | 144](#)

[Administering Proxy RADIUS | 153](#)

[Administering RADIUS Tunnels | 168](#)

[Administering Address Pools | 182](#)

[Setting Up Administrator Accounts | 195](#)

[Configuring Realm Support | 199](#)

[Setting Up Filters | 217](#)

[Setting Up Authentication Policies | 232](#)

[Setting Up EAP Methods | 253](#)

[Configuring Replication | 316](#)

[3GPP Support | 336](#)

RADIUS Basics

IN THIS CHAPTER

- RADIUS Overview | 34
- Attributes | 41
- Centralized Configuration Management | 48
- Proxy RADIUS | 50
- Authentication | 51
- Password Protocols | 58
- Accounting | 60
- Request Routing | 64
- Radius Client Groups | 73
- IP Address Assignment | 74
- Resource Management | 78
- IPv6 Support | 84

This chapter presents a conceptual overview of RADIUS authentication, authorization, and accounting services. This chapter contains these topics:

RADIUS Overview

RADIUS is an industry-standard protocol for providing authentication, authorization, and accounting services.

- **Authentication** is the process of verifying a user's identity and associating additional information (attributes) to the user's login session.
- **Authorization** is the process of determining whether the user is allowed on the network and controlling network access values based on a defined security policy.
- **Accounting** is the process of generating log files that record session statistics used for billing, system diagnosis, and usage planning.

A RADIUS-based remote access environment typically involves four types of components:

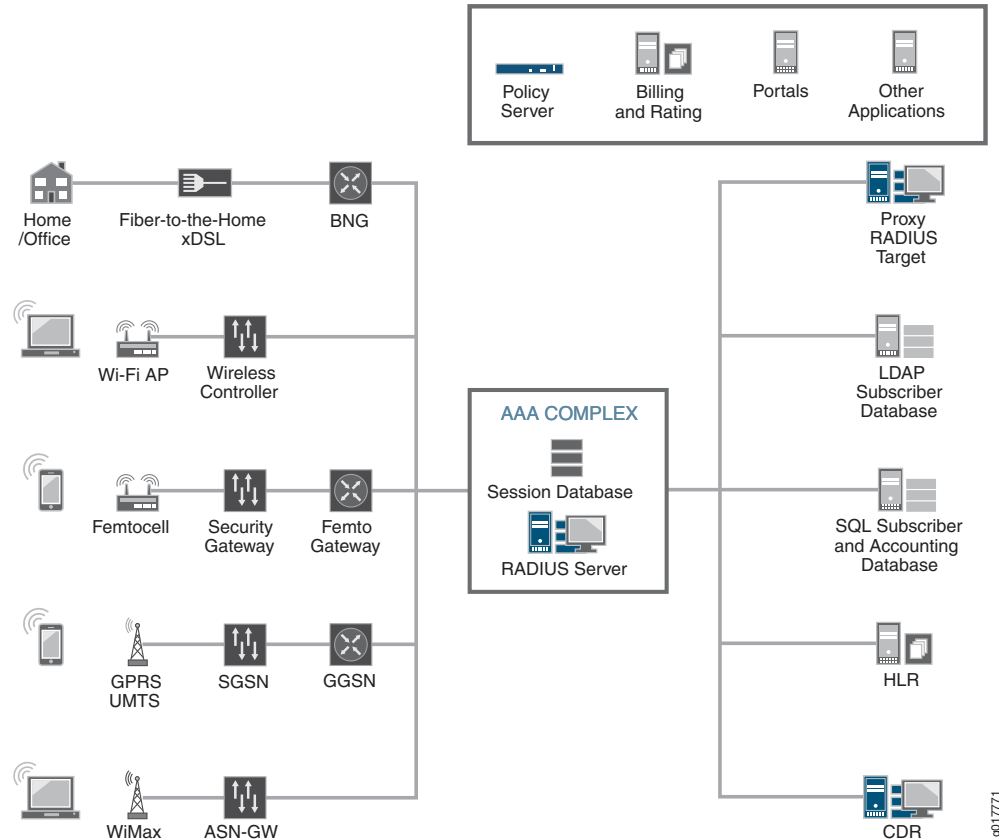
- An *access client* is a user who initiates a network connection. An access client might be a user dialing into a service provider network, a router at a small office/home office connecting to an enterprise network to provide network access, or a wireless client connecting to an 802.1X access point.
- A *network access server* (NAS), also called a RADIUS client, is a device that recognizes and processes connection requests from outside the network edge. A NAS can be a wireless access point, a modem pool, a network firewall, or any other device that needs to authenticate users. When the NAS receives a user's connection request, it may perform an initial access negotiation with the user to obtain identity/password information. The NAS then passes this information to the RADIUS server as part of an authentication/authorization request.

NOTE: The terms *network access device* (NAD), *remote access server* (RAS), and *network access server* (NAS) are interchangeable. This guide primarily uses the term NAS.

- The *RADIUS server* matches data from the authentication/authorization request with information in a trusted database, such as the database on the Steel-Belted Radius Carrier server or a back-end database server. If a match is found and the user's credentials are correct, the RADIUS server sends an Access-Accept message to the NAS. If a match is not found or a problem is found with the user's credentials, the server returns an Access-Reject message. The NAS then establishes or terminates the user's connection. The NAS may then forward accounting information to the RADIUS server to document the transaction; the RADIUS server may store or forward this information as needed to support billing for the services provided.
- In some networks, a *back-end authentication server*, such as SQL or LDAP database; or some other RADIUS server for which this server is a proxy, stores the information against which the authentication request is compared. In some cases, the back-end server passes information to the RADIUS server, which determines whether a match exists. In other cases, the matching is performed on the back-end server, which then passes an accept/reject result to the RADIUS server.

Figure 12 on page 36 illustrates a simple RADIUS environment.

Figure 12: RADIUS Authentication



RADIUS Packets

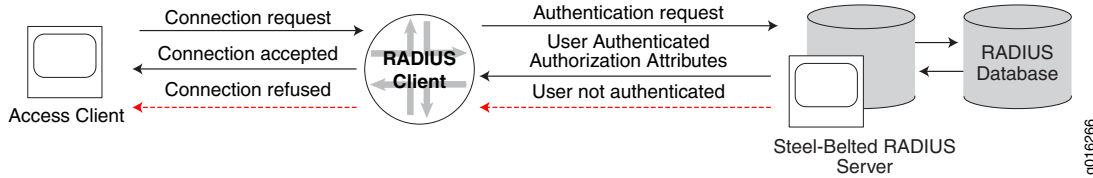
A RADIUS client and RADIUS server communicate by means of RADIUS packets. RADIUS packets carry messages between the RADIUS client and RADIUS server in a series of request/response transactions: the client sends a request and expects a response from the server. If the response does not arrive, the client can retry the request periodically.

Each RADIUS packet supports a specific purpose: authentication or accounting. A packet can contain values called *attributes*. The specific attributes to be found in each packet depend upon the type of packet (authentication or accounting) and the device that sent it (for example, the specific make and model of the NAS acting as a RADIUS client).

For information about RADIUS authentication packet structures and attributes, see RFC 2865, *Remote Authentication Dial-In User Service (RADIUS)*. For information about RADIUS accounting packet structures and attributes, see RFC 2866, *RADIUS Accounting*.

Figure 13 on page 37 illustrates a simple RADIUS authentication/authorization sequence.

Figure 13: RADIUS Authentication



1. The RADIUS access client sends an authentication request containing identification and connection information to the network access server (RADIUS client).
2. When the NAS receives a user's connection request, it typically performs an initial access negotiation with the user to establish connection information (username, password, network access server identifier, NAS port number, and so on). The NAS then forwards the user information in an authentication request to the RADIUS server.
3. The RADIUS server looks up the user information in a local or remote RADIUS authentication database. The RADIUS server verifies that the user's name and password are valid. It can also enforce fine-grained security rules by using an *access check list* to verify specific attributes in the authentication request.
4. If a match is found, the RADIUS server returns an Access-Accept message. The RADIUS server might also send *return list* information stored in the database, such as the user's authorization or connection parameters, back to the NAS.

If a match is not found, the RADIUS server returns an Access-Reject message.

5. Based on the information it receives from the RADIUS server, the NAS accepts or refuses the connection request.

After the user is authenticated and the connection established, the NAS may forward accounting data to the RADIUS server to document the transaction; the RADIUS server can store or forward this data to support billing for the services provided.

RADIUS Ports

The RADIUS standard initially used UDP ports 1645 and 1646 for RADIUS authentication and accounting packets. The RADIUS standards group later changed the port assignments to 1812 and 1813, but many organizations still use the old 1645/1646 port numbers for RADIUS.

Any two devices that exchange RADIUS packets must use compatible UDP port numbers. That is, if you are configuring a NAS to exchange authentication packets with a RADIUS server, you must find out which port the server uses to receive authentication packets from its clients (1812, for example). You must then configure the NAS to send authentication packets on the same port (1812). The same is true for RADIUS accounting.

Steel-Belted Radius Carrier can listen on multiple ports. For compatibility, the server listens to the old and new default RADIUS ports: ports 1645 and 1812 for authentication, and ports 1646 and 1813 for

accounting. To add, change, or disable the ports on which Steel-Belted Radius Carrier listens, modify the **radius.ini** file or edit the **/etc/services** file. The **radius.ini** file is described in the *SBR Carrier Reference Guide*.

RADIUS Configuration

You must configure a RADIUS client and RADIUS server before they can communicate. If the client and server are on the same network, one administrator may be able to configure both sides of the RADIUS communication. If the client and server are not administered by the same person, you may have to coordinate RADIUS configuration details with the administrators of other networks.

RADIUS Server Configuration

To configure Steel-Belted Radius Carrier to respond to RADIUS clients, open the **RADIUS Clients List** page in Web GUI and enter the following information for each RADIUS client:

- The IP address of the client device.
- The RADIUS shared secret used by Steel-Belted Radius Carrier and the client device. For information about RADIUS shared secrets, see [“Shared Secrets” on page 39](#).
- The make and model of the client device, selected from a list of devices that Steel-Belted Radius Carrier supports. If a specific make and model is not listed, select - **Standard Radius** -.

Additionally, you must configure the UDP ports the server uses when sending and receiving RADIUS authentication and accounting packets. The UDP ports you configure on the RADIUS server must match the UDP ports that the RADIUS client is using for the same purposes. For more information, see [“RADIUS Ports” on page 37](#).

RADIUS Client Configuration

You must tell each RADIUS client how to contact its RADIUS server. To configure a client to work with a Steel-Belted Radius Carrier server, log in to the client device, run its administration program, bring up its RADIUS configuration interface, and enter the following information:

- The IP address of the Steel-Belted Radius Carrier server.
- The RADIUS shared secret to be used by Steel-Belted Radius Carrier and the client device. For information about RADIUS shared secrets, see [“Shared Secrets” on page 39](#).
- The UDP ports on which to send and receive RADIUS authentication and accounting packets. These must match the UDP ports that Steel-Belted Radius Carrier is using for the same purposes. For more information, see [“RADIUS Ports” on page 37](#).

Multiple RADIUS Servers

You can distribute the RADIUS workload among several servers, as follows:

- You can set up separate servers for RADIUS authentication and accounting services. When RADIUS authentication and accounting services are performed by separate servers, each client device must be

configured to send its authentication packets to one RADIUS server and its accounting packets to another.

- You can provide redundancy by pairing RADIUS servers to work in tandem. Most NAS configuration interfaces permit you to designate primary and secondary servers for authentication and accounting.

If both measures for distributing the RADIUS workload are implemented, client configuration involves identifying four servers for each client device: a primary RADIUS accounting server, a secondary RADIUS accounting server, a primary RADIUS authentication server, and a secondary RADIUS authentication server.

Shared Secrets

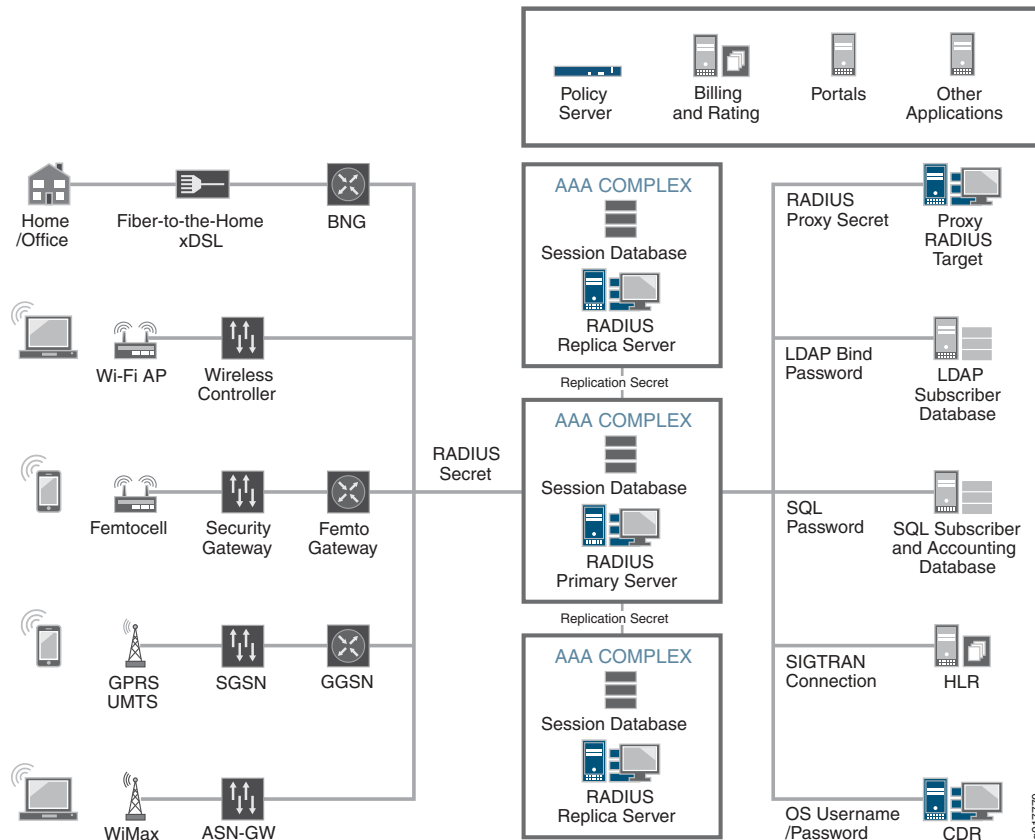
A shared secret is a case-sensitive text string used to validate communications between two RADIUS devices. You should configure shared secrets that are long enough and random enough to resist attack, and you should avoid using the same shared secret throughout your network.

Steel-Belted Radius Carrier uses the following types of shared secrets:

- RADIUS secret—Used to authenticate communication between a RADIUS server and a RADIUS client
- Replication secret—Used to authenticate communication between a primary server and a replica server

[Figure 14 on page 40](#) illustrates the usage of shared secrets in a RADIUS environment.

Figure 14: Shared Secrets



RADIUS Secret

A RADIUS shared secret is a case-sensitive password used to validate communications between a RADIUS server such as Steel-Belted Radius Carrier, and a RADIUS client, such as a network access server. Steel-Belted Radius Carrier supports shared secrets of up to 127 alphanumeric characters, including spaces and the following special characters:

```
~!@#$%^&*()_+|\=-'{}:"';<>?/.,
```

Identical shared secrets must be configured on both sides of the RADIUS communication link.

NOTE: Not all network access servers support shared secrets of up to 127 alphanumeric/special characters. Select shared secrets that are fully supported by RADIUS devices in your network.

Most RADIUS clients allow you to configure different secrets for authentication and accounting. On the server side, the configuration interface allows you to create a list of known RADIUS clients (network access

servers). You can usually identify the authentication shared secret and accounting shared secret that a server uses to communicate with each of the clients on this list.

During an authentication transaction, password information must be transmitted securely between the RADIUS client (network access server) and Steel-Belted Radius Carrier. Steel-Belted Radius Carrier uses the authentication shared secret to encrypt and decrypt password information.

No encryption is involved in transmitting accounting data between a RADIUS client and RADIUS server. However, the accounting shared secret is used by each device to verify that it can *trust* any RADIUS communications it receives from the other device.

Replication Secret

A replication secret is a text string used to authenticate communications between a primary server and a replica server. You do not need to configure the replication secret for a realm: the primary server generates it automatically, and each replica server in a realm receives the replication secret as part of its configuration package.

See [“Overview of Replication” on page 316](#) for information about primary and replica servers.

Accounting

A NAS can issue an Accounting-Request whenever it chooses, for example upon establishing a successful connection. Each time an Accounting-Request message arrives at the Steel-Belted Radius Carrier server, an accounting transaction begins. During this transaction, the server handles the message by examining the Acct-Status-Type and other attributes within the message, and taking the appropriate action.

Attributes

You work with RADIUS attributes while setting up users, profiles, and RADIUS clients in Steel-Belted Radius Carrier. The Web GUI lets you select RADIUS attributes by name from a predefined list. For each attribute, the Web GUI prompts you to enter values using familiar data types such as string, integer, or network address.

Dictionaries

Steel-Belted Radius Carrier uses dictionary files to define RADIUS and Diameter attributes, as well as some attributes defined by Microsoft, 3GPP, and others used internally by SBR. Steel-Belted Radius Carrier uses these definitions to parse authentication/accounting requests and generate responses.

The main Steel-Belted Radius Carrier dictionary files (**radius.dct** and **radius.dic**) list attributes defined by the RADIUS and Diameter standards. The **radius.dct** and **radius.dic** files reside in the same directory as Steel-Belted Radius Carrier (usually **/opt/JNPRsbr/radius** on Solaris and Linux systems).

Vendor-Specific Attributes

In addition to the standard attributes, many network access servers use Vendor-Specific Attributes (VSAs). Steel-Belted Radius Carrier supports a large number of specific network access servers by providing vendor-specific dictionary files. These files also reside in the server directory and use the filename extensions **.dct** and **.dic**.

The Steel-Belted Radius Carrier package contains **juniper.dct** and **juniper.dic** dictionary files that list all Juniper VSAs that are used for fixed and mobile subscriber management (ERX Series and MX Series routers, with vendor ID 4874) and router administration (vendor ID 2636). Dictionaries for many versions of Juniper products, including the E Series and MX Series routers, are available at <https://www.juniper.net/support/downloads/> (in the **Documentation** tab for the specific product). For more information about how to add the dictionaries to SBR Carrier, see the “Dictionary Files” section in the *SBR Carrier Reference Guide*.

Dictionaries and the Make/Model Field

During Steel-Belted Radius Carrier configuration, when you make a selection in the RADIUS Client **Make/Model** field, you are telling the server which dictionary file contains the VSAs for this client device. Thereafter, whenever the server receives a RADIUS packet from this client device, it can consult this dictionary file for any nonstandard attributes that it encounters in the packet. Standard RADIUS attributes are always defined by the **radius.dct** file. If you are not sure which make/model you should specify for a RADIUS Client, choose - **Standard RADIUS** -.

The selections available in the **Make/model** field identify devices whose vendors have provided attribute dictionaries for use with Steel-Belted Radius Carrier.

Updating Attribute Information

If your NAS vendor announces a new product, a new attribute, or a new value for an attribute, you can add this information to your Steel-Belted Radius Carrier configuration. You can edit the dictionary file for that vendor to add new attributes or attribute values, or you can create a new vendor-specific dictionary file that contains new attributes and values. For more information about dictionary files, refer to the *SBR Carrier Reference Guide*.

Structured Attributes

Steel-Belted Radius Carrier natively supports structured attributes that contain subattributes.

Subattributes are values in a RADIUS packet that are not stored as a RADIUS AVP or vendor-specific-attribute (VSA), but rather are packed with other subattributes into a RADIUS VSA. In a RADIUS packet, multiple RADIUS VSAs might contain subattributes. The RADIUS VSA, which consists of multiple subattributes, is sometimes referred to as a *structured attribute* because it contains structured data.

Earlier Steel-Belted Radius products, such as the SIM Server product and Mobile IP Module, only interpreted AVPs and not the subattributes contained within the AVP. These earlier products included plug-ins that copied the subattributes from the AVP container and represented them in the RADIUS request as if they

had been received as separate AVPs. In the response, the RADIUS AVPs were reassembled from their contained subattribute values. This process was known as packet *flattening* and *unflattening*.

The dictionary mechanism of Steel-Belted Radius Carrier has been extended to allow for XML declaration of structured AVP contents. When an AVP with an associated structure definition is received, its internal subattribute values are automatically parsed and become available to any component within Steel-Belted Radius Carrier that processes RADIUS requests. Similarly, any subattribute values that are populated into the RADIUS response are formatted as part of the structured RADIUS AVP according to the same XML structure definition.

Subattribute dictionary definitions are defined in **.jdict** files. The **.jdict** definition files reside in the **radiusdir/subattributes** directory. All files ending in **.jdict** are parsed by Steel-Belted Radius Carrier as subattribute files. Once parsed, the definitions are merged into the **.dct** definitions.

If you used the packet flattening/unflattening method in Steel-Belted Radius Carrier 7.2 and previous versions, we recommend that you migrate to using subattributes.

NOTE:

- *.dic dictionary files do not support structured attribute definitions.
- The Class attribute cannot contain structured attributes.

User Attribute Lists

Each user entry in the Steel-Belted Radius Carrier database provides the information necessary for the server to try to authenticate a connection request using a specific authentication method. When you view a user entry using the Web GUI, this method is identified in the **User type** field.

You can control authentication at finer levels of detail than simple username/ password checking allow. The check list, return list, or profile fields in the user entry in the database provide powerful tools for the authentication and authorization of users. These fields direct the server how to handle RADIUS attributes while authenticating a connection request and can be used to configure the authorization of the session.

Check List Attributes

A *check list* is a set of attributes that must accompany the authentication request before the request can be accepted. The NAS must send attributes that match the check list associated with a user entry; otherwise, Steel-Belted Radius Carrier rejects the user even if the user's name and password are valid.

By including appropriate attributes in the check list, a variety of rules can be enforced. For example, only specific users might be permitted to use ISDN or dial-in connections to a particular NAS, or Caller ID might be used to validate a user against a list of acceptable originating telephone numbers.

A check list is created by selecting attributes from a list of all RADIUS attributes known to the Steel-Belted Radius Carrier server. This list can include a variety of vendor-specific attributes.

During authentication, Steel-Belted Radius Carrier filters the check list based on the dictionary for the RADIUS client that sent the authentication request. The server ignores any check list attribute that is not valid for this device.

Return List Attributes

A *return list* is a set of attributes that Steel-Belted Radius Carrier must return to the NAS after authentication succeeds. The return list usually provides additional parameters that the NAS needs to complete the connection. Return list attributes can thus be considered to be *authorization configuration parameters*.

By including appropriate attributes in the return list, you can create a variety of connection policies. Specific users can be assigned particular IP addresses; IP header compression can be turned on or off; or a time limit can be assigned to the connection.

You create a return list by selecting attributes from a list of all RADIUS attributes known to Steel-Belted Radius Carrier. This list can include a variety of vendor-specific attributes.

During authentication, Steel-Belted Radius Carrier filters the return list based on the dictionary for the RADIUS client that sent the authentication request. The server omits any return list attribute that is not valid for this device.

Structured Attributes in Check Lists and Return Lists

When configuring check list and return list attributes, if the selected attribute has corresponding subattributes defined in the Steel-Belted Radius Carrier dictionaries, an *Add Child* button is available, which enables you to add the corresponding subattributes.

Attribute Values

The value of each RADIUS attribute has a well-defined data type: numeric, string, IP address, time, or hexadecimal. For example, **Callback-Number** is of type **string** and is a telephone number entered by the administrator, whereas **NAS-Port-Type** is an integer attribute that has defined values in the dictionary which appear in the GUI and can be **Sync**, **Async**, and so forth.

NOTE: Steel-Belted Radius Carrier supports signed integers (negative numbers) for attributes received in packets. However, Web GUI do not support signed integers, and treats signed and unsigned integers as unsigned integers.

Single- and Multi-Valued Attributes

Attributes can be single- or multi-valued. Single-valued attributes appear at most once in the check list or return list; multi-valued attributes may appear several times.

If an attribute appears more than once in the check list, this means that any one of the values is valid. For example, you can set up a check list to include multiple telephone numbers for attribute **Calling-Station-ID**.

A user trying to dial into your network would then have to call from one of the designated telephone numbers to be authenticated.

If an attribute appears more than once in the return list, each value of the attribute is sent as part of the response packet.

Orderable Multi-Valued Attributes

Certain multi-valued return list attributes are also orderable, which means the attribute can appear more than once in a RADIUS response, and the order in which the attributes appear is important.

For example, the **Reply-Message** attribute allows text messages to be sent back to the user for display. A multi-line message is sent by including this attribute multiple times in the return list, with each line of the message in its proper sequence.

System Assigned Values

Some attributes do not allow the administrator to set a value. Steel-Belted Radius Carrier retrieves the appropriate value for this attribute when it is needed.

Echo Property

Using the echo property, you can force an attribute from the RADIUS request to be echoed in the RADIUS response. For example, you might add **Callback-Number** to the return list and select the **echo** check box. Steel-Belted Radius Carrier takes the value of the Callback-Number it receives in the RADIUS request and echoes it back to the client in the RADIUS response; if it receives no Callback-Number, it echoes nothing.

You enter **Callback-Number** one or more times into the check list. This indicates that one of the callback numbers you supplied must be present in the RADIUS request, and that number should be echoed in the RADIUS response.

NOTE: The echo property is disabled for multi-valued return list attributes. The echo property is also disabled for the Framed-IPv6-Address attribute regardless of its multi-value setting.

Default Values

Selecting **default** for a check list attribute specifies that, if the RADIUS request does not include this attribute, the request should not be rejected. Instead, the value supplied as the default should be used as if it were received as part of the request. One use for default values is to require that an attribute in a RADIUS request must have one of several values, or must not be present at all. Another use is to provide a default value for an attribute in conjunction with the echo property in the return list.

Steel-Belted Radius Carrier can provide alternate values when an attribute appears in the check list marked as **default**, and the same attribute appears in the return list marked as **echo**. The server echoes the actual value of the attribute in the RADIUS response if the attribute appears in the RADIUS request and echoes

the default value (from the check list) in the response if the attribute does not appear in the RADIUS request.

If you add multiple values of the same attribute to the check list, only one of them can be marked as default.

For example, an administrator adds several **Callback-Number** values to the check list and marks one of them as default. The administrator adds **Callback-Number** to the return list and specifies it as echo.

- If a **Callback-Number** value is present in the RADIUS request, it must match one of the check list values or the user is rejected.
- If it does match, the user is accepted and the value supplied is echoed in the RADIUS response.
- If no **Callback-Number** is supplied in the request, the user is accepted and the default value is echoed in the response.

Other check list attributes are used to provide configuration for the user, such as time-of-day and concurrent-login-limit information.

Wildcard Support

Steel-Belted Radius Carrier supports wildcards (? and *) for string-type attributes in check list items.

To allow backward compatibility with check list items that treat the string literally, a string containing wildcards must be prefixed with a caret (^). When the caret is present, the remainder of the string is parsed using escape rules.

A ? wildcard matches any character and a * wildcard matches the remainder of the string (but can appear only at the end of a string). Wildcard characters can be treated as literals by using escape codes (for example, \?). [Table 10 on page 46](#) lists the non-ASCII characters that can also be present in the wildcard string:

Table 10: Non-ASCII Characters in Wildcard Strings

Code	Meaning
\a	BEL
\b	Backspace
\f	Form feed
\n	Linefeed
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab

Table 10: Non-ASCII Characters in Wildcard Strings *(continued)*

Code	Meaning
\\	Backslash
*	Literal '*' (not wildcard)
\?	Literal '?' (not wildcard)
\xnn	Where <i>nn</i> is a hexadecimal value
\nnn	Where <i>nnn</i> is a decimal value

A '\ ' followed by any other character represents that character's value.

The following is a wildcard example for string type attributes:

Called-Station-ID = ^800*

Where Called-Station-ID indicates any 800 number.

Attribute Filtering

You can filter specific RADIUS attribute/value pairs into and out of RADIUS packets as they travel to and from directed realms and proxy RADIUS realms. Attribute filtering can be useful if there is data in the packets that is needed for routing, but not for authentication or accounting.

Adding NAS Location Attributes to Access-Requests

Steel-Belted Radius Carrier core provides an attribute handling feature, which allows you to add NAS location information to proxied Access-Request messages.

Service providers might require the location of the mobile device requesting access. For example, a service provider might offer weather reports or advertising based on the location of the mobile device.

You can configure an Access-Request to include the location of the NAS through which the proxied request was processed. Because the NAS is geographically near the mobile device, it closely approximates the location of the mobile device.

When a mobile device is outside the area of its provider, it roams by sending the request to a local foreign AAA (FAAA) server that is owned by another provider. The FAAA server proxies (forwards) the request to the appropriate home AAA (HAAA) server for the user.

For proxied requests, Steel-Belted Radius Carrier can perform a lookup to find a NAS location based on an attribute (usually NAS-Identifier or NAS-IP-Address). Steel-Belted Radius Carrier can perform a query

to find the value of the attribute that identifies the NAS location. The NAS location is then added to the Access-Request that is sent to the service provider’s home AAA server. The attribute that is used to look up the NAS location is user-configurable as the AttributeTolIdentifyNAS in the **locspectrl** file.

For details on configuring this feature, see *Adding NAS Location Information to Access-Request Messages* in the *Attribute Processing Files* section of the *SBR Carrier Reference Guide*.

Specifying IPv4 Address Classes

You can specify IPv4 address classes using class-based subnetting. [Table 11 on page 48](#) lists example usages for class-based subnetting.

Table 11: Example Usages for Class-Based Subnetting

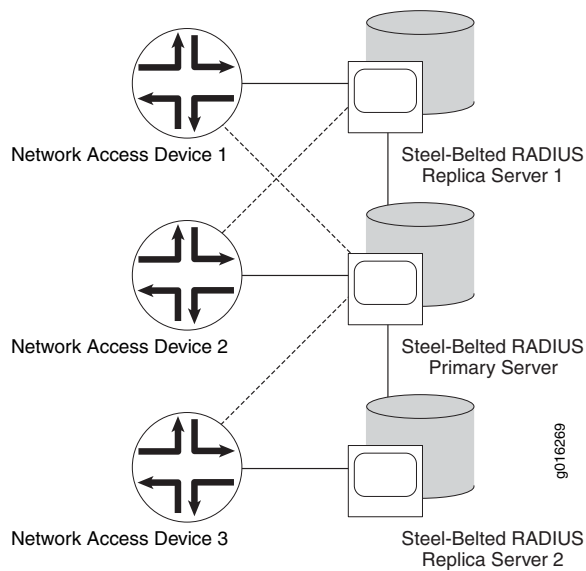
Class	Example IP Address Range	Class-Based Subnetting
A	10.0.0.0–10.255.255.255	10.0.0.0
B	176.1.0.0–176.1.255.255	176.1.0.0
C	192.168.15.0–192.168.15.255	192.168.15.0
D	225.224.224.15	225.224.224.15 Address match required

Centralized Configuration Management

Steel-Belted Radius Carrier supports the replication of RADIUS configuration data from a *primary server* to a maximum of 10 *replica servers* within a *replication realm*. Replica servers help balance the load of authentication requests coming in from RADIUS clients, and ensure that authentication services are not interrupted if the primary or other replica servers stop working.

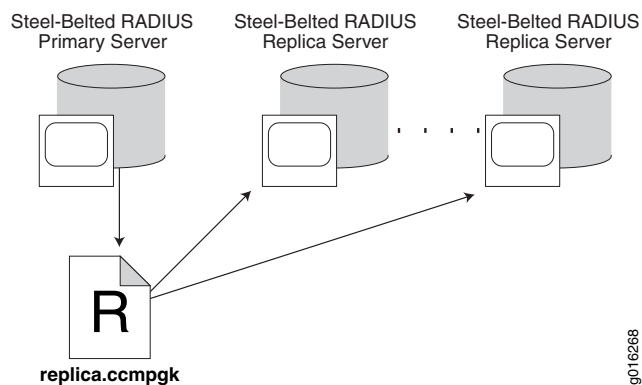
For example, [Figure 15 on page 49](#) illustrates an environment where RADIUS traffic is load-balanced by configuring each network access server (NAS) to authenticate users through a different RADIUS server (solid line). If a RADIUS server becomes unavailable, the NAS can fail over to its backup RADIUS server (dotted line).

Figure 15: Using Replication for Redundancy and Load Balancing



All the servers within a realm reflect the current configuration specified by the network administrator: the network administrator modifies the configuration on the primary server, and the primary server propagates the new configuration to its replica servers. For example, after a network administrator configures a new RADIUS client or profile on the primary server, the network administrator directs the primary server to publish a configuration package file (**replica.cmpkg**) that contains the updated configuration information. After publication, the primary server notifies each replica server that a new configuration package is ready. Each replica then downloads and installs the configuration package to update its settings.

Figure 16: Configuration Package Publication



The primary server maintains a list of the replica servers that have registered with it. The primary server uses this list to track which servers to notify after it publishes an updated configuration package to resynchronize the configuration of replica servers.

If the primary server needs to be taken out of service, the network administrator promotes one of the replica servers to be the new primary server. Thereafter, the other replica servers copy the configuration package from the promoted primary server.

Proxy RADIUS

The Steel-Belted Radius Carrier server can forward a RADIUS request to another server for processing and relay the other server's result back to its client. In such cases, Steel-Belted Radius Carrier is acting as a *proxy* for the *target server*, and Steel-Belted Radius Carrier is *proxy-forwarding* the request to the target server.

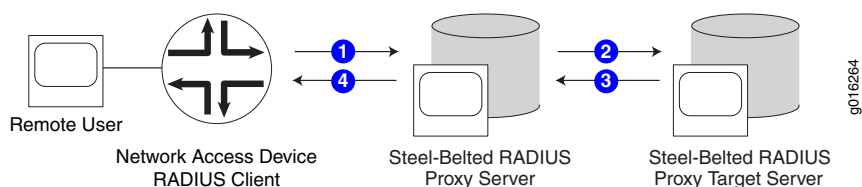
Steel-Belted Radius Carrier supports proxy RADIUS; any Steel-Belted Radius Carrier server can act as proxy or target for authentication or accounting messages.

Proxy RADIUS Authentication

RADIUS authentication messages are forwarded by proxy as follows:

1. An access client requests authentication from a RADIUS client, which sends an authentication request to a RADIUS proxy server.
2. The proxy RADIUS server forwards the message to a RADIUS *target* server.
3. The target RADIUS server performs the authentication services indicated by the message, then returns a response message to the proxy RADIUS server.
4. The proxy RADIUS server relays the acknowledgement response message to the RADIUS client.

Figure 17: RADIUS Proxy Forwarding



Proxy RADIUS Accounting

RADIUS accounting messages are proxyforwarded as follows:

1. A RADIUS server receives an accounting request.
2. Depending on its configuration, the RADIUS proxy server forwards the accounting message to a target accounting server or records accounting attributes locally (or does both).

3. If the proxy server does not receive an acknowledgement of the forwarded accounting message, it re-sends periodically according to its retry policy.
4. When the target server acknowledges the request, the proxy server forwards an acknowledgement to the RADIUS client.

Authentication

RADIUS uses different types of messages during user authentication. [Table 12 on page 51](#) summarizes the conditions under which each type of RADIUS authentication message is issued, and the purpose of any RADIUS attributes the message contains.

Table 12: RADIUS Authentication Messages and Attributes

Message Conditions	Purpose of Message Attributes
When a NAS receives a connection request from a user, the NAS requests authentication by sending an AccessRequest to its RADIUS server.	<p>Identify the user.</p> <p>Describe the type of connection the user is trying to establish.</p>
When a RADIUS server is able to authenticate a user, it returns a RADIUS Access-Accept to the NAS.	<p>Allow the NAS to complete access negotiations.</p> <p>Configure connection details such as providing the NAS with an IP address it can assign to the user.</p> <p>Enforce time limits and other <i>class of service</i> restrictions on the connection.</p>
When a RADIUS server is unable to authenticate a connection request, it returns an Access-Reject to the NAS.	<p>Terminate access negotiations.</p> <p>Identify the reason for the authorization failure.</p>
If initial authentication conditions are met but additional input is needed from the user, the RADIUS server returns an Access-Challenge to the NAS.	<p>Enable the NAS to prompt the user for more authentication data.</p> <p>Complete the current AccessRequest, so the NAS can issue a new one.</p>

Authentication Methods

Each time an `AccessRequest` message reaches the server, an authentication transaction begins. During this transaction, the server attempts to authenticate the request by sequentially trying its configured and enabled authentication methods. The server consults its list of authentication methods to determine which methods to try and in which order to try them.

Native User Authentication

Native user authentication references user accounts stored on the Steel-Belted Radius Carrier server. When trying the native user method, Steel-Belted Radius Carrier searches its database for an entry whose `User-Type` is **Native User**, and whose username matches the username in the `AccessRequest`.

- If the entry cannot be found, or if it is found and the password is invalid, Steel-Belted Radius Carrier tries the next enabled method in the authentication methods list.
- If an entry for the user is found but the entry's check list does not match attributes found in the `AccessRequest`, Steel-Belted Radius Carrier returns an `AccessReject` message to the NAS.
- If the entry is found and its password and check list match perfectly, Steel-Belted Radius Carrier formats an `AccessAccept` message using the entry's return list, and returns it to the NAS.

Pass-Through Authentication

Pass-through authentication methods permit Steel-Belted Radius Carrier to begin the authentication by asking another entity to validate the username and password found in the `AccessRequest`.

Steel-Belted Radius Carrier can pass authentication requests through to a back-end database or a third party Authentication Manager. For example, when using the optional SIM authentication module, you can authenticate users against the credentials in your HLR.

Proxy RADIUS Authentication

Steel-Belted Radius Carrier can convey an `AccessRequest` to some other RADIUS server, which then (1) attempts to authenticate the connection request according to its own conventions and (2) returns a response to Steel-Belted Radius Carrier. Steel-Belted Radius Carrier then relays this response to the NAS. The set of conventions for relaying packets between cooperating RADIUS servers is known as *proxy RADIUS*.

External Authentication

External authentication methods enable Steel-Belted Radius Carrier to authenticate users by referring to external SQL or LDAP databases. During external authentication, Steel-Belted Radius Carrier queries the database for authentication data, and uses the results to format a response packet. Steel-Belted Radius Carrier then relays this response to the NAS.

For information about using Steel-Belted Radius Carrier with SQL databases, see [“Configuring SQL Authentication” on page 442](#). For information about using Steel-Belted Radius Carrier with LDAP databases, see [“Configuring LDAP Authentication” on page 471](#).

Directed Authentication

Every authentication request works its way through the same Authentication Methods list until one of the methods succeeds or the end of the list is reached.

This behavior might not be ideal for every account. If you want requests from certain users or accounts to bypass the primary Authentication Methods list and use an alternate list, you can do so by employing the directed authentication feature. This feature allows you to map the UserName or DNIS information in an incoming authentication request to a specific list of authentication methods. The list can include any native, pass-through, proxy-as-authentication, or external database authentication method configured on the Steel-Belted Radius Carrier server.

You can also direct authentication towards a particular realm using a technique called *attribute mapping*. This allows you to check for the presence or absence of a particular attribute in an authentication request, or for an attribute containing a specific value. Attribute mapping can be used with both proxy realms and directed realms.

Authenticate-Only Requests

Steel-Belted Radius Carrier supports requests to authenticate a user where the server performs no other processing. The NAS specifies this type of request by setting the **Service-Type** field to a value of **Authenticate-Only** (numeric value 8). The server responds with either an Access-Reject or an Access-Accept (without any attributes).

You can disable this feature (so that attributes are always returned in the response packet) by setting the **AuthenticateOnly** field in the [Configuration] section of the **radius.ini** file to 0. For more information about **radius.ini**, refer to the *SBR Carrier Reference Guide*.

Configuring the Authentication Sequence

After you configure authentication methods for Steel-Belted Radius Carrier, the **Authentication Methods** page in Web GUI displays them in the order in which the server tries them. The **Authentication Methods** page in Web GUI displays both active and inactive authentication methods. During an authentication transaction, the server works down the list of Active Authentication Methods.

You can activate or deactivate methods, or reorder methods in the list, by using the controls in the **Authentication Methods** page in Web GUI. For information about setting up authentication sequences, see [“Setting Up Authentication Policies” on page 232](#).

Configuring Authentication Methods

Each authentication method in Steel-Belted Radius Carrier performs a different type of processing on information in an incoming Access-Request. [Table 13 on page 54](#) summarizes what you need to do to configure each authentication method.

Table 13: Authentication Method Configuration

Method	How to Configure	See
Native User	Create native user entries in the Steel-Belted Radius Carrier database.	“Setting Up Native Users” on page 122
UNIX Pass-Through Security	This method assumes that you already have users, groups, and passwords defined in your local security database. Create user entries in the database of the UNIX server hosting Steel-Belted Radius Carrier. Choose User-types as appropriate.	“Administering Users” on page 120
External SQL Database	This method assumes that you have user records stored in a SQL database. Create a Steel-Belted Radius Carrier .aut file that connects to a SQL database and issues a SELECT query based upon the username and password. Give the .aut file a unique InitializationString value. Stop and restart Steel-Belted Radius Carrier. Subsequently, the SQL authentication method appears in the Authentication Methods page, using the InitializationString value as its name. You can use the Authentication Methods page to enable, disable, and re-order the SQL authentication method.	“Configuring SQL Authentication” on page 445
External LDAP Database	This method assumes that you have user records stored in an LDAP database. Create a Steel-Belted Radius Carrier .aut file that validates the username and password based upon Bind and Search requests to an LDAP database. Give the .aut file a unique InitializationString value. Stop and restart Steel-Belted Radius Carrier. Subsequently, the LDAP authentication method appears in the Authentication Methods page, using the InitializationString value as its name. You can use the Authentication Methods page to enable, disable, and re-order the LDAP authentication method as desired.	“Configuring LDAP Authentication” on page 474

Advanced Options

Steel-Belted Radius Carrier provides the following additional authentication control options:

Account Lockout

Account lockout allows you to disable an account after a configurable number of failed login attempts within a configurable period. For example, if a user enters an incorrect password three times within two minutes, Steel-Belted Radius Carrier can lock out the user’s account temporarily. During the lockout period, the user cannot log in, even with the correct password.

When a user account is locked out, the user must wait until the expiration of the lockout period, or a network administrator can clear the lockout status for the account.

For information about displaying and administering locked accounts, see [“Using the Locked Accounts List” on page 846](#).

NOTE: Do not enable account lockout and account redirection at the same time. If account lockout and account redirection are both enabled, account lockout is used and account redirection settings are ignored.

NOTE: Account lockout state is not maintained if Steel-Belted Radius Carrier is restarted. Steel-Belted Radius Carrier enforces lockout locally only, not globally.

Account Redirection

Account redirection allows you to flag an account for special processing after a configurable number of failed login attempts within a configurable period. For example, if a user enters an incorrect password three times within two minutes, Steel-Belted Radius Carrier can accept the user (even with an incorrect password) but limit the user's access to specific network resources, such as a secure webpage that prompts the user to provide other authentication information. If the user can obtain his or her current password (or can create a new one through such a secure webpage), he or she can then reconnect and log in successfully.

When account redirection is enabled and a user repeatedly enters an incorrect password, Steel-Belted Radius Carrier places the user in redirect state. When a user is in redirect state:

- If the user does not submit another authentication request within a specified timeout period, the user account is released from redirect state and returned to normal state.
- If the user submits another authentication request within a specified timeout period, the user is accepted without authentication/authorization processing. The accept message for the user includes the attributes and values specified in a redirection profile, and the user is placed into Access-Pending state. The attributes and values in the Access-Accept message are used by an external customer process, which may prompt the user to enter alternate authentication information to receive a password by email.

When a user is in Accept-Pending state, the next authentication request received determines whether the user is accepted or locked out:

- If the user enters the appropriate authentication information, the user is returned to normal state and Steel-Belted Radius Carrier generates an informational SNMP trap message.
- If the user does not enter the appropriate authentication information, Steel-Belted Radius Carrier issues an Accept-Reject message, locks the user out of the network for a configurable lockout period,

and generates an informational SNMP trap message. During this lockout period, authentication requests for the user are automatically rejected, even if the user enters the correct password.

Optionally, you can identify RADIUS clients that you want to exclude from account redirection processing. Authentication requests from excluded RADIUS clients are processed normally, without use of redirection or account state changes.

NOTE: Do not enable account lockout and account redirection at the same time. If account lockout and account redirection are both enabled, account lockout is used and account redirection settings are ignored.

NOTE: Account redirection state is not maintained if SBR Carrier is restarted.

Blocklisting

Blocklisting allows you to automatically reject authentication requests that contain certain values. You configure blocklisting by setting up a profile that identifies check list attributes that should trigger an automatic authentication failure, and then modifying the **blacklist.ini** file to use that profile. For example, you could set up a profile to block users calling from a specific area code by configuring a **Calling-Station-Id** check list attribute in the blacklist profile. You can use * and ? wildcards in the blacklist profile.

NOTE: You can blacklist individual users by setting their **Concurrency Limit = 0**.

Blocklisting functions on all local authentication requests and can be configured to include proxy-RADIUS requests.

For information about setting up profiles, see [“Administering Profiles” on page 144](#). For information about configuring the **blacklist.ini** file, refer to the *SBR Carrier Reference Guide*.

Allowed Access Hours

Steel-Belted Radius Carrier provides a vendor-specific attribute called **Funk-Allowed-Access-Hours**. This attribute can be placed in the check list for a user or profile entry to control the times during which a user can be allowed access.

NOTE: See [“Allowed Access Hours” on page 120](#) for the format of this value (and how to enter it into a user or profile record).

During authentication, the server processes the current time, the **Funk-Allowed-Access-Hours** value from the check list, and the **Session-Timeout** value from the return list.

- If a **Funk-Allowed-Access-Hours** attribute is present in the check list, and if the present time does not fall within a valid time period according to **Funk-Allowed-Access-Hours**, the server rejects the session.
- If a **Funk-Allowed-Access-Hours** attribute is present in the check list, and if the current time falls within a valid time range according to **Funk-Allowed-Access-Hours**, the server accepts the session and calculates a session end time as follows:
 - If a **Session-Timeout** attribute exists in the user's return list, Steel-Belted Radius Carrier adds this number of seconds to the present time to calculate a proposed session end time. If the proposed session end time falls within the current time period (as defined by **Funk-Allowed-Access-Hours**), then Steel-Belted Radius Carrier returns the proposed session end time to the NAS in the **Session-Timeout** attribute.

If the proposed end time occurs after the end of the current time period, then Steel-Belted Radius Carrier calculates the number of seconds between the present time and the end of the current time period (as defined by **Funk-Allowed-Access-Hours**, and returns this value to the NAS in the **Session-Timeout** attribute.

- If a **Funk-Allowed-Access-Hours** attribute is present in the check list but a **Session-Timeout** attribute does not exist in the user's return list, Steel-Belted Radius Carrier computes a value for **Session-Timeout** based on the number of seconds between the present time and the end of the current time period. Steel-Belted Radius Carrier returns this value to the NAS in the **Session-Timeout** attribute.
- If **Funk-Allowed-Access-Hours** is not present in the check list, the server returns the **Session-Timeout** value from the user's return list.
- If neither attribute is present, no **Session-Timeout** value is returned in the Access-Accept message, and the session is limited by the **StaleSessionTimeoutSecs** setting in the **radius.ini** file.

Two-Factor Authentication

Extensible Authentication Protocol-Tunneled Transport Layer Security (EAP-TTLS) provides for certificate-based mutual authentication between a client and a network through an encrypted tunnel. A typical implementation of EAP-TTLS uses certificates on authentication servers to create a network-to-user encryption tunnel, and then uses EAP inside the TLS tunnel for user-to-network authentication.

An enhanced version of EAP-TTLS uses certificates on the client side to provide *two-factor authentication*: the end user must have both a private key for a valid certificate and the password to an active account to obtain network access.

When client certificate support in EAP-TTLS is enabled on the server, you must provide a list of trusted root certificates from which offered client certificates must derive. These certificates must be provided in DER-encoded form and must be placed in the root subdirectory of the server directory.

Optionally, you can enable certificate revocation list (CRL) checking as part of the EAP-TTLS authentication process. CRL checking verifies that an unexpired certificate has not been revoked by its issuing Certificate Authority (CA) for any reason, such as a suspected security breach. Enabling CRL checking means that, every time the client requests a connection, Steel-Belted Radius Carrier checks the CRL to confirm that the client certificate has not been revoked. This improves security but increases processing overhead.

If client certificate support is not enabled in EAP-TTLS, any trusted root certificates and CRL checking options are ignored.

Password Protocols

During an authentication transaction, password information is transmitted between the NAS and the RADIUS server. This password information originally comes from the user, for example during PPP negotiations between a user and a NAS. Steel-Belted Radius Carrier supports three protocols (PAP, CHAP, and MS-CHAP v2) for receiving the password from the NAS. Steel-Belted Radius Carrier also supports the *Extensible Authentication Protocol* (EAP).

[Table 14 on page 58](#) lists supported protocols according to the authentication methods with which each protocol can be used.

Table 14: Authentication Methods and Password Protocols

Method	PAP	CHAP	MS-CHAP v2
LDAP	Yes	Yes, if BindName is used and the password is in clear-text form or is encrypted with enc-md5.	Yes, if the LDAP server can return a clear-text password, or an MD4 hash Unicode format password.
		No, if Bind is used	No, if Bind is used
Local (Native)	Yes	Yes	Yes
Proxy RADIUS	Yes	Yes	Yes
SQL	Yes	Yes, if the password is available in clear-text form in the database or is encrypted with enc-md5.	Yes, if BindName is used and the password is in clear-text form, is encrypted with enc-md5, or is the MD4 hash of the Unicode form of the password.
UNIX User	Yes	No	No

Table 14: Authentication Methods and Password Protocols (*continued*)

Method	PAP	CHAP	MS-CHAP v2
UNIX Group	Yes	No	No

Password Authentication Protocol

When the Password Authentication Protocol (PAP) is used, no encryption is used to send the password to the NAS during the negotiation process between the remote user and the NAS. After the NAS has enough information from the user to create an Access-Request, the NAS encrypts the password (using its RADIUS shared secret) before sending an Access-Request packet to Steel-Belted Radius Carrier.

Upon receiving the Access-Request, Steel-Belted Radius Carrier looks for attributes within the packet that identify the NAS that sent it. Steel-Belted Radius Carrier decrypts the password by using the shared secret configured for the RADIUS client entry associated with the sending NAS.

Ultimately, Steel-Belted Radius Carrier has the password in clear-text form for authentication.

Challenge Handshake Authentication Protocol

The Challenge Handshake Authentication Protocol (CHAP) avoids sending passwords in clear-text over any communication link. Under CHAP, during password negotiations the NAS generates a *challenge* (a random string) and sends it to the user. The user's PPP client creates a *digest* (the password concatenated with the challenge), encrypts the digest using one-way encryption, and sends the digest to the NAS.

The NAS sends this digest as the password in the Access-Request.

Because the encryption is one-way, Steel-Belted Radius Carrier cannot recover the password from the digest. Instead, it performs an identical operation, using the NAS's challenge value (provided in the Access-Request packet) and its own copy of the user's password to generate its own digest. If the two digests match, the password is the same.

Steel-Belted Radius Carrier must be able to perform the digest operation to support CHAP. Therefore, it must have access to its own copy of the user's password. Native User passwords are stored in the Steel-Belted Radius Carrier database. SQL or LDAP BindName authentication retrieves the password by means of a query to the database; the retrieved password can be used to create a digest if it is in clear-text form.

MS-CHAP v2

MS-CHAP v2 (Microsoft Challenge Handshake Authentication Protocol version 2) is a Microsoft authentication protocol that, like CHAP, avoids sending passwords in clear-text. MS-CHAP v1 is not supported.

Steel-Belted Radius Carrier must be able to perform a digest operation similar to CHAP to support MS-CHAP v2. Therefore, it must have access to its own copy of the user's password. Native User passwords are stored in the Steel-Belted Radius Carrier database. SQL or LDAP BindName authentication retrieves the password by means of a query to the database; the retrieved password can be used to create a digest if it is in clear-text form.

MS-CHAP v2 communicates users' requests to change their passwords to a RADIUS server. Steel-Belted Radius Carrier supports this feature, although it must also be supported by whatever application the user is using to log in. For more information about MS-CHAP v2, see RFC 2433, *Microsoft PPP CHAP Extensions*; RFC 2548, *Microsoft Vendor-specific RADIUS Attributes*; and RFC 2759, *Microsoft PPP CHAP Extensions, Version 2*.

Accounting

To understand the Steel-Belted Radius Carrier accounting sequence, you need an overview of RADIUS accounting messages. [Table 15 on page 60](#) explains the conditions under which each type of message is issued, and the purpose of any RADIUS attributes that a message contains.

Table 15: Message Conditions and Attributes

Message Conditions	Purpose of Message Attributes
<p>The RADIUS client sends accounting data to Steel-Belted Radius Carrier using an Accounting-Request message.</p> <p>The RADIUS client is responsible for verifying that the server receives accounting requests. Most clients retry periodically until the server responds.</p>	<p>Depending on the value of the Acct-Status-Type attribute, the message type is considered to be Start, Stop, Interim-Acct, Accounting-On, or Accounting-Off.</p>
<p>Upon receipt of an Accounting-Request message, the server sends an Accounting-Response.</p>	<p>Complete the request/response cycle.</p>
<p>After receiving an Access-Accept from the server, the NAS completes its access negotiation with the user. The NAS then sends a Start message to the server.</p>	<p>Record connection data, such as username, NAS identifier, NAS port identifier, port type, and connection start time.</p>
<p>At intervals of approximately every six minutes, the NAS sends an Interim-Acct message to the server.</p>	<p>Record a <i>snapshot</i> of statistics regarding the connection. One message contains the current value of every statistic that this NAS is capable of recording about this type of connection.</p>

Table 15: Message Conditions and Attributes *(continued)*

Message Conditions	Purpose of Message Attributes
After a connection is terminated, the NAS sends a Stop message to the server.	Record statistics regarding the connection. One message contains the final value of every statistic that this NAS is capable of recording about this type of connection.
Every time a client device comes online, whether after a crash or after an orderly shutdown, it sends an Accounting-On message to the server.	Identify the device that is going online and clear all session information.
Every time a client device experiences an orderly shutdown, before completing its shutdown sequence it sends an Accounting-Off message to the server.	Identify the device that is going offline and clear all session information.

Accounting Sequence

A NAS can issue an Accounting-Request whenever it chooses, for example upon establishing a successful connection. Each time an Accounting-Request message reaches Steel-Belted Radius Carrier, an accounting transaction begins. During this transaction, the server handles the message by examining the Acct-Status-Type and other attributes within the message, and taking the appropriate action.

Comma-Delimited Log Files

When the Steel-Belted Radius Carrier accounting log is enabled, all of the RADIUS accounting attributes that the server receives are reformatted and logged to a comma-separated value (CSV) text file, which is easily imported into spreadsheets and database programs for report generation and billing.

Proxy RADIUS Accounting

Steel-Belted Radius Carrier can relay an Accounting-Request to some other RADIUS server, which records the data according to its own, locally-configured RADIUS accounting options. (You have the option of specifying that the data also be recorded locally on the Steel-Belted Radius Carrier server.) The set of conventions for relaying packets between cooperating RADIUS servers is known as *proxy RADIUS*, and is defined in the RADIUS standard.

See [“Configuring a Proxy RADIUS Realm” on page 201](#).

External Accounting

External accounting methods permit Steel-Belted Radius Carrier to record accounting data to external databases. Configuration files specify how Steel-Belted Radius Carrier communicates with an external database and how to insert accounting data into that database.

SQL is the only external accounting method currently supported by Steel-Belted Radius Carrier.

See [“SQL Accounting Overview” on page 458](#).

Tunneled Accounting

During authentication, a user is typically identified by attributes such as **User-Name** (in the authentication request) and **Class** (in the authentication accept response). Standard RADIUS accounting requests typically include these attributes in messages flagging Start, Interim, and Stop events so that the user’s identity can be recorded for accounting and auditing purposes.

When an organization uses a tunneled authentication protocol such as EAP/TTLS or EAP/PEAP, the identity of a user requesting authentication may be concealed from the NAS; the **User-Name** attribute carried by the outer authentication protocol is typically a non-unique value such as *anonymous*. As a result, the outer User-Name value included in accounting requests may not be sufficient to determine a user’s identity. Class attributes provided by an authentication server cannot be included in clear-text in an outer Access-Accept message because they might contain clues about the user’s identity, thereby defeating the identity-hiding feature of the tunneled protocol.

Tunneled accounting allows Steel-Belted Radius Carrier to pass user identity information to accounting processes without exposing user identities to a NAS that should not see them. When tunneled accounting is enabled, RADIUS attributes are encrypted and encapsulated in a Class attribute. If the information for a Class attribute exceeds the attribute payload size (253 octets), Steel-Belted Radius Carrier returns more than one Class attribute for a user.

The tunneled accounting transaction sequence is:

1. The Steel-Belted Radius Carrier server acting as the tunnel endpoint for EAP/TTLS or EAP/PEAP encrypts a user’s inner **User-Name** and **Class** attributes when it authenticates the user.
2. The server returns the encrypted information to the NAS encapsulated in a Class attribute in the outer Access-Accept message. The NAS associates this encapsulated identity attribute with the user, and echoes the encapsulated identity attribute whenever it generates an accounting request for the user.
3. When Steel-Belted Radius Carrier receives an accounting request from a network access server, it scans the request for an encapsulated identity attribute.
4. If Steel-Belted Radius Carrier finds an encapsulated identity attribute, it de-encapsulates and decrypts the attributes to reconstitute the original inner **User-Name** and **Class** attributes.
5. Steel-Belted Radius Carrier substitutes the decrypted attributes for the ones returned from the NAS.
6. Steel-Belted Radius Carrier processes the accounting request locally or forwards the accounting request through the proxy to its intended target.

To implement tunneled accounting, you must configure the **classmap.ini** file to specify how attributes should be presented, and you must configure the **spi.ini** file to specify the keys that are used to encrypt and decrypt users’ identity information. The **classmap.ini** file and the **spi.ini** file are described in the *SBR Carrier Reference Guide*.

For an overview of how EAP/TTLS and EAP/PEAP work, refer to [“About the Extensible Authentication Protocol” on page 253](#).

Directed Accounting

The directed accounting feature allows you to map an incoming accounting request to one or more accounting methods, based on routing information found in the request packet. Among the options available with directed accounting is that of establishing an accounting log file that is distinct from the Steel-Belted Radius Carrier accounting log file in the server directory, and that contains entries from only those accounting requests that were specifically directed to the realm.

See [“Configuring a Directed Realm” on page 207](#).

Accounting Spooling

Accounting spooling can improve proxy accounting reliability. When spooling is enabled for a realm, Steel-Belted Radius Carrier immediately acknowledges all accounting requests for that realm to the NAS. Meanwhile, it spools accounting requests to a file while a separate thread unspools requests and sends them to the server responsible for the realm. If the server is unavailable, Steel-Belted Radius Carrier retries at regular intervals until the proxy target acknowledges the request. Even if Steel-Belted Radius Carrier restarts, all spooled requests are preserved until they are completed.

Account spooling offers the following benefits:

- The NAS always gets an immediate ACK (acknowledgement response) for accounting requests.
- Accounting data is never lost if it is sent to a Steel-Belted Radius Carrier server with spooling enabled.

A separate and independent spooler is maintained for each realm for which proxy spooling is configured. When an accounting request is received for a realm implementing proxy spooling, it is written to a file in the target directory and a request is prepared, followed by an acknowledgement returned to the client. The file is then read by the unspooling thread and the prepared request proxied.

Targets, fast-fail, round-robin, and other extended proxy features operate normally, but unspooling continues to retry sending a request until it is successfully acknowledged. Since each spooler is independent, one unresponsive realm does not affect the delivery of spooled requests to other realms.

The Acct-Delay-Time attribute in a request is updated or added as necessary if there is a delay between the spooling and the forwarding of the request.

When the spool file’s rollover interval expires or the file size exceeds the rollover size limit, the current spool file is closed for writing and a new one created. Files are named in the format, **yyyymmdd_hhmm_ssss.psf**, where **yyyy** is the year, **mm** is the month, **dd** is the day, **hh** is the hour, **mm** is the minute, and **ssss** is a sequence number. The configuration of the rollover settings enables spooling to be optimized according to the characteristics of the operating environment.

When Steel-Belted Radius Carrier is shut down, unspooling continues for the configured **ShutdownDelay** time until all spooled packets are sent. If the destination server is down at the time of shutdown, however,

unspooling terminates immediately. After startup, unspooling continues from the beginning of the oldest spool file.

NOTE: Account Spooling guarantees sequential delivery of proxied accounting packets through a single-threaded mechanism. However, the use of Account Spooling can adversely affect performance in systems that may sustain heavy load and high latency, or both of proxy targets. See the *SBR Carrier Reference Guide* for more information on the settings for spooling.

Request Routing

When Steel-Belted Radius Carrier receives a RADIUS authentication or RADIUS accounting request, it examines the attributes in the request to match the request with the service that can best respond to it:

- RADIUS authentication or accounting
- Proxy RADIUS authentication or accounting
- Tunnel authentication
- Directed authentication or accounting

This matching process (request routing) uses the information in the **User-Name** attribute (in which routing information is supplied as a prefix or suffix to the user's account name), the **Called-Station-Id** attribute (DNIS), or other attribute(s) to route the request.

Steel-Belted Radius Carrier follows the order of matching techniques configured in the [Processing] section of proxy.ini. For example, it may check **User-Name** and **Called-Station-Id** attributes, then it checks the attributes you have mapped to realms. It uses the first routing destination it finds.

An exception to this routing logic is when the authentication request is handled by a directed realm and that directed realm also has accounting enabled. When this occurs, the class attribute will have a subattribute called VirtualRealm added to it. If the accounting request received by the radius server has the class attribute sent in the Access Accept, the routing logic above is bypassed and the directed realm listed in the VirtualRealm field will be used to process the accounting request.

Match Rules

In simple cases, each entry in the [Realms] or [Directed] sections of the **proxy.ini** file consists of the name of a domain. For prefix- and suffix-based realm matching, the domain name is taken from the **User-Name** attribute (for example, **other.com** in **user@other.com**). The domain name also supplies the root of the

name of the configuration file that specifies the treatment for that realm (for example, **other.com.pro** or **other.com.dir**).

Match rules extend the format of lines in the [Realms] or [Directed] sections by allowing leading and trailing wildcard support and by allowing multiple entries to be mapped to the same realm.

Match rules in the [Realms] or [Directed] sections of the **proxy.ini** file take the form:

```
RealmName [= match_rule ]
```

- Leading wildcards allow you to match any domain name that ends with the appropriate domain name string and map it to a realm name:

```
realm1 = *.msn.com
```

- Trailing wildcards allow you to match any domain name that begins with the appropriate domain name string and map it to a realm name:

```
realm2 = usa.*
```

- Direct matching allows you to map a domain name to a realm name:

```
realm3 = boston.other.com
```

A standalone wildcard character allows you to match any domain name that does not meet the criteria of a more specific rule:

```
realm4 = *
```

You can use match rules to map more than one entry in the [Realms] or [Directed] section of **proxy.ini** to a realm. When Steel-Belted Radius Carrier performs prefix or suffix processing of the **User-Name**, it isolates the realm portion of the **User-Name** attribute and then searches through the match rules to find the one that matches the realm information most closely. The best match is the one that specifies the largest number of non-wildcard characters in the rule.

For example, assume the following entries are present in the [Realms] or [Directed] section of your **proxy.ini** file:

```
realm1 = *.msn.com
realm2 = usa.msn.com
realm3 = *.uk.msn.com
realm4 = *.com
```

```
realm5 = *
```

Table 16 on page 66 identifies what realms Steel-Belted Radius Carrier chooses for different **User-Name** values.

Table 16: Realm Mapping Example

User-Name	Mapped Realm
bob@usa.msn.com	realm2
alice@scotland.uk.msn.com	realm3
lauren@wales.uk.msn.com	realm3
rich@germany.msn.com	realm1
julia@indiana.usa.msn.com	realm1
ramon@other.com	realm4
seema@other.edu	realm5

Realm names can appear more than once within the [Realms] or [Directed] section of **proxy.ini** to point multiple match rules to the same realm. For example, you can enter the following settings in the [Realms] or [Directed] section of your **proxy.ini** file:

```
realm1 = *.msn.com
realm2 = usa.msn.com
realm2 = uk.msn.com
```

These entries map any username with a domain of usa.msn.com or uk.msn.com to realm2, and map other domain ending in msn.com to realm1.

You cannot enter duplicate realm lines (with or without match rules) in the [Realms] or [Directed] section of your **proxy.ini** file, and you cannot enter the same match rule on multiple lines.

You cannot combine leading and trailing wildcards in the same match rule. If you do so, the trailing match rule is ignored.

User-Names with a Single Delimiter

An incoming User-Name string can be *decorated* with a single delimiter separating the user's name from a destination name. A User-Name decorated in this manner can indicate a proxy RADIUS realm, a directed realm, a tunnel, or a proxy entry that is not a member of any realm.

NOTE: To prevent unexpected routing results, you must ensure that the name of every realm, tunnel, and proxy entry is unique across your entire Steel-Belted Radius Carrier configuration.

User-Names with a Single Tunnel Delimiter

If the delimiter matches the currently configured delimiter for tunnels, and if the current name-parsing convention for tunnels is **suffix**, the User-Name is understood to be:

```
User<SuffixDelimiter>TunnelName
```

If the current name parsing convention for tunnels is **prefix**, the User-Name is understood to be:

```
TunnelName<PrefixDelimiter>User
```

Where:

- **User** is the name of the dial-in user
- **TunnelName** identifies the destination
- **<SuffixDelimiter>** or **<PrefixDelimiter>** is a delimiter character such as @, / or !

If a tunnel entry is found that matches the **TunnelName**, and the request is for authentication, Steel-Belted Radius Carrier proceeds with tunnel authentication.

NOTE: Tunnel delimiters are specified in the **Name Parsing** page (described in [“Configuring RADIUS Tunnels” on page 171](#)). You may use either the prefix or the suffix naming convention for tunnels, but not both. You can also choose the tunnel delimiter character ('@', '/', and so forth). The conventions you define in the **Name Parsing** page apply to all tunnels defined on the server.

User-Names with a Single Realm Delimiter

If the User-Name contains a single delimiter that matches the currently configured suffix delimiter for realm destinations, the User-Name is understood to be:

User<SuffixDelimiter>RealmName

If the User-Name contains a single delimiter that matches the currently configured prefix delimiter for realm destinations, the User-Name is understood to be:

RealmName<PrefixDelimiter>User

Where:

- **User** is the name of the dial-in user
- **RealmName** identifies the destination
- **<SuffixDelimiter>** or **<PrefixDelimiter>** is a delimiter character such as @, / or !

Steel-Belted Radius Carrier attempts to find a destination that matches **RealmName** in one of four places, as summarized in [Table 17 on page 68](#).

Table 17: Realm Name Matching

If a match is found in:	Then Steel-Belted Radius Carrier does this:
[Self] section of the radius.ini file.	Steel-Belted Radius Carrier services the request locally.
The [Directed] section of the proxy.ini file.	Steel-Belted Radius Carrier routes the request to a specific authentication or accounting method on the local server, according to the rules in the corresponding RealmName.dir file.
The [Realms] section of the proxy.ini file.	Steel-Belted Radius Carrier routes the request to the proxy RADIUS realm called RealmName according to the rules in the corresponding RealmName.pro file. See “Target Selection within a Realm” on page 155 .
A proxy entry in the Steel-Belted Radius Carrier database	Steel-Belted Radius Carrier uses the information in the proxy entry (IP address, UDP port, and shared secret) to forward the RADIUS request.

NOTE: Realm delimiters and naming conventions are defined in the **proxy.ini** file. You can define different delimiters for prefixes and suffixes. The conventions you define in **proxy.ini** apply to all types of realm defined on the server (both proxy realms and directed realms).

User-Names with Multiple Suffix Delimiters

If the User-Name contains multiple realm delimiters (**User**<**Delimiter**>**RealmName** <**Delimiter**>**RealmName**<**Delimiter**>**RealmName**) and the delimiter character matches the current RealmSuffix setting in the [Configuration] section of **proxy.ini**, the name parsing strategy is as follows:

1. Steel-Belted Radius Carrier finds the leftmost **RealmName** in the User-Name that is also listed in the [Self] section of its **radius.ini** configuration file.
2. If a matching **RealmName** was found in Step 1, and there is no other **RealmName** to the left of it, then Steel-Belted Radius Carrier services the request locally, without forwarding.
3. If a matching **RealmName** was found in Step 1, but there is another **RealmName** to the left of it, then Steel-Belted Radius Carrier routes the request to the **RealmName** listed immediately to the left of the matching RealmName. The routing is controlled by the corresponding **RealmName .pro** or **RealmName .dir** file.
4. If no **RealmName** match was selected in Steps 1, 2, or 3, then Steel-Belted Radius Carrier routes the request to the rightmost **RealmName** in **User-Name**. The routing is controlled by the corresponding **RealmName .pro** or **RealmName .dir** file.

According to these rules, if the realm suffix **Delimiter** character is '@', and the **User-Name** value matches realm suffix naming conventions, and the [Self] section of **radius.ini** lists one realm called **bigserver**, then incoming **User-Name** values are parsed as described in [Table 18 on page 69](#).

Table 18: Realms in User-Names

Request	Action
fred@bignet@bigserver	Routed to the realm called bignet .
fred@bignet@bigserver@smallnet	Routed to the realm called bignet .
fred@bignet@smallnet	Routed to the realm called smallnet .
fred@bigserver@bignet	Handled locally on bigserver .

User-Names with Multiple Prefix Delimiters

If the User-Name contains multiple realm delimiters:

RealmName<Delimiter>RealmName<Delimiter>RealmName <Delimiter>User

and the **Delimiter** character matches the current RealmPrefix setting in the [Configuration] section of **proxy.ini**, the name parsing strategy is the reverse of the suffix strategy described above. In detail:

1. Steel-Belted Radius Carrier finds the rightmost **RealmName** in the **User-Name** that is also listed in the [Self] section of its **radius.ini** configuration file.
2. If a matching **RealmName** was found in Step 1, and there is no other **RealmName** to the right of it, then Steel-Belted Radius Carrier services the request locally, without forwarding.
3. If a matching **RealmName** was found in Step 1, but there is another **RealmName** to the right of it, then Steel-Belted Radius Carrier routes the request to the **RealmName** listed immediately to the right of the matching **RealmName**. The routing is controlled by the corresponding **RealmName .pro** or **RealmName .dir** file.
4. If no **RealmName** was selected in Steps 1, 2, or 3, then Steel-Belted Radius Carrier routes the request to the leftmost **RealmName** in the **User-Name**. The routing is controlled by the corresponding **RealmName.pro** or **RealmName.dir** file.

According to these rules, if the realm prefix **Delimiter** character is '!', and the User-Name matches realm prefix naming conventions, and the [Self] section of **radius.ini** lists one realm called **bigserver**, then incoming **User-Name** values are parsed as described in [Table 19 on page 70](#).

Table 19: Realms and User-Names

Request	Action
superserver!bignet!fred	Routed to the realm called bignet .
smallnet!bigserver!bignet!fred	Routed to the realm called bignet .
smallnet!bignet!fred	Routed to the realm called smallnet .
bignet!bigserver!fred	Handled locally on bigserver .

Undecorated User-Names

An undecorated User-Name is a username that does not include realm identification information. When realm support for undecorated User-Names is enabled, Steel-Belted Radius Carrier routes authentication requests that contain undecorated **User-Name** attributes to the proxy realm or directed realm designated in the [Realms] or [Directed] section of the **proxy.ini** file.

Undecorated User-Name support allows you to specify a realm to handle any request containing a User-Name that does not contain realm identification information.

Steel-Belted Radius Carrier uses the following logic to determine if a User-Name is undecorated:

1. If **Suffix** is enabled in the [Processing] section of the **proxy.ini** file and the User-Name contains the specified **RealmSuffix** character, the User-Name is not undecorated. By default, @ is the default **RealmSuffix** character and that **Suffix** is enabled if the **proxy.ini** file does not include a [Processing] section.

2. If **Prefix** is enabled in the [Processing] section of the **proxy.ini** file and the User-Name contains the specified **RealmPrefix** character, the User-Name is not undecorated. By default, / is the default **RealmPrefix** character and that **Prefix** is enabled if the **proxy.ini** file does not include a [Processing] section.
3. If neither of the above statements is true, the User-Name is undecorated.

Configuring Undecorated User-Name Support

To configure undecorated User-Name support in Steel-Belted Radius Carrier:

1. Add an **Undecorated** entry to the [Processing] section of **proxy.ini**.
If the **proxy.ini** file does not include a [Processing] section or if the **Undecorated** entry is commented out or not present, undecorated User-Name processing is disabled.
2. Associate an **<undecorated>** marker with a proxy realm (in the [Realms] section of **proxy.ini**) or a directed realm (in the [Directed] section of **proxy.ini**).

Only one realm listed in the [Realms] or [Directed] section of **proxy.ini** can be configured with the = **<undecorated>** setting. If more than one realm is associated with the = **<undecorated>** setting, Steel-Belted Radius Carrier enables the first entry it finds and writes an error message identifying the duplicate realms to the server log file.

3. Verify that the private directory for Steel-Belted Radius Carrier includes a **realm.pro** file (for the proxy realm associated with undecorated User-Name processing) or **realm.dir** file (for the directed realm associated with undecorated User-Name processing) that matches the realm specified in Step [“Configuring Undecorated User-Name Support” on page 71](#)

For example, if you enter **other.com = <undecorated>** in the [Realms] section of **proxy.ini**, you must have an **other.com.pro** file in the Steel-Belted Radius Carrier directory. If the applicable **realm.pro** file is not found, Steel-Belted Radius Carrier writes an error message noting the realm has not been enabled to the server log file.

Example

The following sections of **proxy.ini** illustrate how undecorated User-Name support is configured in Steel-Belted Radius Carrier.

```
[Processing]
Undecorated
Suffix
Prefix
DNIS
Attribute-Mapping
```

```
[Realms]
bigco.com
```

```
partner.com
other.com = <undecorated>
```

In this example, any authentication request that contains undecorated **User-Name** attributes is handled by the **other.com** realm. Successful authentication requests that are handled by the selected realm results in a **Class** attribute that records the name of the realm so that accounting requests resulting from the session can be handled by the same realm.

This is different from cases where a decorated User-Name (that is, a User-Name that contains a prefix or suffix delimiter) does not match explicit realm mapping rules. For example, since no matching rule for **littlecorp.com** is configured, a request containing a **User-Name** value of **user@littlecorp.com** would fall through to local processing and result in a rejection if no matching user is found.

NOTE: All settings are reloaded on receipt of a SIGHUP (1) signal.

Request Routing by DNIS

If the **Called-Station-Id** attribute is found in the RADIUS request, the request can be routed based on DNIS (Dial Number Information Services). Steel-Belted Radius Carrier checks its administration database and server configuration files for a DNIS string that matches the value of the incoming **Called-Station-Id** attribute. If found, the matching string might be in one of three places:

- A tunnel entry's Called Station Id list. If a match is found here, and the request is for authentication, Steel-Belted Radius Carrier performs tunnel authentication. See [“Tunnel Authentication Sequence” on page 169](#).
- The [Called-Station-ID] section of a **RealmName.dir** file. If a match is found here, Steel-Belted Radius Carrier handles the request locally using the authentication or accounting methods identified in the **RealmName.dir** file. See [“Configuring a Directed Realm” on page 207](#).
- The [Called-Station-ID] section of a **RealmName.pro** file. If a match is found here, Steel-Belted Radius Carrier routes the request to the proxy RADIUS realm called **RealmName** using the rules defined in the **RealmName.pro** file. See [“Target Selection within a Realm” on page 155](#).

NOTE: We recommend that you use DNIS strings that are unique across all tunnel entries, all **RealmName.dir** files, and all **RealmName.pro** files. If you duplicate a DNIS string anywhere in your Steel-Belted Radius Carrier configuration, the request routing results might be unexpected.

Request Routing by Any Attribute

You can map the presence or absence of any attribute or attribute/value pair in an incoming packet to a specific realm, by providing an [AuthAttributeMap] or [AcctAttributeMap] section in the **proxy.ini** configuration file.

You can route all of the packets for a session to a realm based on attributes in the Access-Request (the [AuthAttributeMap] section), or you can route the session's accounting packets to a different realm, based on attributes found in these packets (the [AcctAttributeMap] section).

Attribute mapping can be used for proxy RADIUS realms and for directed realms. You cannot use this feature when forwarding packets to a proxy target that is not a member of a realm.

Local Services

If the RADIUS request does not contain routing information (or at least, it does not contain any routing information that Steel-Belted Radius Carrier has been configured to recognize), it is processed locally on the Steel-Belted Radius Carrier server. Authentication follows the authentication methods list in the **Authentication Methods** page. No User-Name parsing is performed; the entire string is understood to be the user's name. Accounting is controlled by the server's main **account.ini** file and (for external database accounting) **.acc** file.

Control over Routing Methods

By default, the rules for determining the destination of a request are applied in the following order by default:

1. Apply Suffix delimiter rules
2. Apply Prefix delimiter rules
3. Apply DNIS rules
4. Apply Attribute Mapping rules

You can specify which methods you want used for request routing and the sequence in which methods are applied.

Radius Client Groups

If your RADIUS clients use the same RADIUS attributes and have contiguous IP addresses, you can configure one or more RADIUS *client groups* and specify an address range consisting of as many as 500 IP addresses for each client group. A RADIUS client group is a collection of network devices or software applications that contacts Steel-Belted Radius Carrier to authenticate a user or to record accounting information about

a network connection. When Steel-Belted Radius Carrier receives a RADIUS request that includes a source IP address in this range, it uses the RADIUS client group to determine the appropriate shared secret, make/model, and IP address pool.

Please note the following when you set up address ranges for RADIUS client groups:

- Address ranges are for IPv4 networks only. Steel-Belted Radius Carrier does not support address ranges for IPv6.
- The address range assigned to one RADIUS client group cannot overlap the address ranges assigned to other RADIUS client groups.
- The starting address of the address range assigned to a RADIUS client group cannot match the IP address of an individual RADIUS client.
- If an individual RADIUS client entry has an IP address that falls within an address range assigned to a RADIUS client group, Steel-Belted Radius Carrier uses the make/model for the individual RADIUS client. For example, if RADIUS client NAS1 is configured with IP address 192.168.21.55 and if RADIUS client group BLDG1NAS is configured with an IP address range 192.168.21.50– 192.168.21.60, Steel-Belted Radius Carrier uses the client information for NAS1 if it receives a RADIUS request from 192.168.21.55, and it uses the client information for BLDG1NAS if it receives a RADIUS request from 192.168.21.56.
- A RADIUS client group cannot use a Class D, E, or F IP address (that is, an address greater than 223.255.255.0).

See [“Adding a RADIUS Client or Client Group” on page 104](#) for information about how to configure IPv4 address ranges for RADIUS clients.

IP Address Assignment

Steel-Belted Radius Carrier can assign IPv4 addresses to users in several ways:

- Static assignment—The same IP address is assigned to a user each time the user connects. For example, if the user **Kevin** has a **Framed-IP-Address** attribute set to **123.11.245.123**, then the IP address 123.11.245.123 is assigned each time Kevin connects to the network.
- Assignment from a specific address pool—An address is assigned from a specific pool when the user connects. For example, if user **Kevin** has a **Framed-IP-Address** attribute set to the **Sales** IP address pool, the next available IP address from **Sales** is assigned when Kevin connects to the network.
- Assignment from the RADIUS client’s IP address pool (or set of IP address pools)—An address is assigned from one of the pools associated with the RADIUS client that makes the connection when a user connects. For example, assume that a RADIUS client called **RAS1** uses IP address pool A, and a RADIUS client called **RAS2** uses IP address pool B. A User entry called **Kevin** has a **Framed-IP-Address** attribute value of **pool associated with RADIUS Client**. When user **Kevin** gets a port on **RAS1**, an IP address from pool

A is assigned. On the next call, **Kevin** might connect to **RAS2**; in this case an address from pool B is assigned.

Alternatively, if a user has been associated with a particular NAS-specific IP address pool (and suffix), an IP address from that pool is assigned.

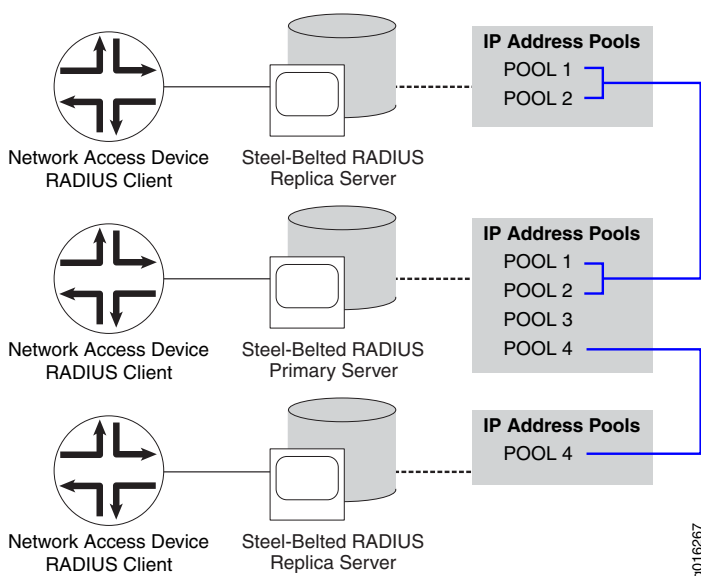
- Assignment from DHCP server—An address is assigned from a DHCP server for a user-configurable period of time (DHCP lease) when a user connects. The DHCP lease period is typically significant (for example, twenty-four hours).

Address Pools and Replication

Address pool applies to non-cluster installations only and the information is not distributed with other configuration information in a replicated environment. If you are using IPv4 address pools in a replication environment, you must configure address pools separately on each replica server, and then use the same names to configure a primary list of address pools on your primary server.

The primary list of address pools configured on the primary server must include the names of all the pools on all of the replica servers. For example, [Figure 18 on page 75](#) illustrates a simple environment that uses four address pools. POOL1 and POOL2 are configured on one replica server and POOL4 is configured on a different replica server. As a consequence, the IP address pool list on the primary server must include POOL1, POOL2, POOL3 (the pool used by the primary server), and POOL4.

Figure 18: IP Address Pools in a Replication Environment



The network administrator must configure RADIUS clients (including the address pool associated with a RADIUS client) on the primary server. This RADIUS client/address pool association (but not the address pool information itself) is stored as part of the replication package passed from the primary server to the replica servers.

Hints

Steel-Belted Radius Carrier can treat the attribute **Framed-IP-Address** as a *hint*. This means that if this attribute appears in the Access-Request and the user return list is configured to allocate **Framed-IP-Address** from a pool, the IP address in the Access-Request is returned instead of the newly-allocated IP address.

This functionality is defined in the [Configuration] section of **radius.ini**:

```
[Configuration]
Framed-IP-AddressHint = <yes/no>
```

When hints are enabled, Steel-Belted Radius Carrier uses a hint to determine the value of the **Framed-IP-Address** attribute in the access response. This means that **Framed-IP-Address** in the Access-Request is returned in the Access-Accept, regardless of the **Framed-IP-Address** value stored in the user's account.

The default value is no.

[Table 20 on page 76](#) details the effect of hints:

Table 20: Effect of Hints

Account Configuration	Framed-IP-Address Returned Without Hints	Framed-IP-Address Returned With Hints
No Framed-IP-Address	No value	Framed-IP-Address from Access-Request
Static Address	Static address	Static address
Address from Pool	Next address from pool	Framed-IP-Address from Access-Request



CAUTION: By using hints, you can assign the same IP address to multiple active accounts.

[Table 21 on page 77](#) details the effect of RFC 6911 hints.

Table 21: Effect of RFC 6911 Hints

Attribute	Present in Access-Request?	Present in Return List?	Response If Hints Set to No	Response If Hints Set to Yes
Framed-IPv6-Address	Yes	Yes	Framed-IPv6-Address from return list	Framed-IPv6-Address from Access-Request
	No	Yes	Framed-IPv6-Address from return list	Framed-IPv6-Address from return list
DNS-Server-IPv6-Address	Yes	Yes	DNS-Server-IPv6-Address from return list	DNS-Server-IPv6-Address from both Access-Request and return list
	No	Yes	DNS-Server-IPv6-Address from return list	DNS-Server-IPv6-Address from return list
Route-IPv6-Information	Yes	Yes	Route-IPv6-Information from return list	Route-IPv6-Information from both Access-Request and return list
	No	Yes	Route-IPv6-Information from return list	Route-IPv6-Information from return list
Delegated-IPv6-Prefix-Pool	Yes	Yes	Delegated-IPv6-Prefix-Pool from return list	Delegated-IPv6-Prefix-Pool from both Access-Request and return list
	No	Yes	Delegated-IPv6-Prefix-Pool from return list	Delegated-IPv6-Prefix-Pool from return list
Stateful-IPv6-Address-Pool	Yes	Yes	Stateful-IPv6-Address-Pool from return list	Stateful-IPv6-Address-Pool from both Access-Request and return list
	No	Yes	Stateful-IPv6-Address-Pool from return list	Stateful-IPv6-Address-Pool from return list

Resource Management

This section explains how Steel-Belted Radius Carrier manages limited resources, such as network addresses, user or tunnel connections, and UDP ports.

Network Address Assignment

The Steel-Belted Radius Carrier address pooling feature allows you to set up one or more pools out of which unique network addresses are assigned dynamically as users require them. Each pool consists of a list of one or more ranges of IP addresses (an IP pool).

By using this feature, you can avoid allocating specific fixed addresses to individual users. You can make fewer addresses go farther, and you can consolidate address assignment across all your network access servers.

How Address Assignment Works

Proper operation of address assignment from a pool depends crucially on both RADIUS authentication and RADIUS accounting transactions, as follows:

1. During the RADIUS authentication transaction, if the user's attribute settings specify address assignment from a pool, an address is allocated for that user from that pool.
2. The address is reserved for that user until a RADIUS accounting transaction indicates that the user has terminated the connection.

For this reason, the network access server must be configured for RADIUS accounting, and the same Steel-Belted Radius Carrier server must be specified for both authentication and accounting.

NOTE: If your network access server is not configured for accounting (or does not support accounting), you should not use the address pooling feature because addresses would be assigned but never released.

Setting Return List Attributes

The **Framed-IP-Address** return list attribute controls how the user's IP address is assigned. The **Framed-IP-Address** attribute can be set for each user in the Steel-Belted Radius Carrier database.

Handling Address Leaks

Under optimal conditions, Steel-Belted Radius Carrier assigns and releases addresses automatically. In some circumstances, you can get *address leakage*, where an address remains reserved for a user after the user has terminated the connection.

Address leakage occurs when the address has been assigned during the authentication transaction, but the accounting transaction that would have released the address is never received by Steel-Belted Radius Carrier. This can occur for several reasons:

- The Steel-Belted Radius Carrier server might have been taken down for a period of time during which accounting transactions occurred.
- The network access server might have failed or been taken down before the user terminated the connection. (In many cases, however, Steel-Belted Radius Carrier might be able to prevent address leakage by recovering the addresses when the NAS starts up again.)
- The network access server might have sent the authentication and accounting transactions to different RADIUS servers.
- Despite a successful authentication, the user's access with the NAS might have terminated unsuccessfully for a variety of reasons. In such a case, some network access servers might not initiate a subsequent accounting transaction.
- Routing problems might have prevented the accounting transaction from reaching Steel-Belted Radius Carrier.

An address that has *leaked* remains out of circulation until the session expires or you manually release it by displaying the current sessions list and deleting the corresponding session. See [“Deleting Sessions” on page 724](#).

NOTE: To prevent IP address leaks, when assigning IP addresses using tunneled authentication methods such as TTLS or PEAP, the "Transfer Inner Attribs To Accept" filter must be enabled, and the Framed-IP-Address attribute must be transferred by the filter. By default, the ttls_accept filter does this properly.

Address Leakage upon Stopping and Starting the Server

Steel-Belted Radius Carrier maintains all current address assignments in a persistent database on disk in non-cluster installations. For cluster, current address assignments are kept in the session database. If you shut down the server and then restart it, all the information about which address is assigned to which user is retained.

If you leave Steel-Belted Radius Carrier turned off for a substantial period of time after addresses have been assigned, address leakage may occur. After you restart the server, review the Current Sessions list (described in [“Current Sessions Overview” on page 718](#)) and delete entries you know are obsolete.

Overlapping Address Ranges

If you maintain multiple IP address pools, you can duplicate some of the addresses among the pools. The address tracking mechanism of Steel-Belted Radius Carrier, when it is enabled, ensures that, if an IP address

appears in more than one pool, after it is assigned out of any pool, it remains unavailable through any of the pools until it is released.

You must disable this type of address tracking if the server is assigning IP addresses from disjointed networks. In that configuration, two numerically identical IP addresses would signal a conflict, even though they actually belong to two different networks.

NOTE: Overlapping IP Address ranges are supported only in the standalone version of SBR Carrier.

Order of Address Assignment

IP addresses are assigned on a first-in-first-out basis; that is, the address that was first released is the first to be reassigned. This ensures that addresses are out of use for as long as possible before reuse.

Concurrent Network Connections

The Web GUI enables you to limit the number of active connections, on a per-user, or per-tunnel basis.

Concurrent User Connections

You can set a maximum limit on the number of concurrent connections that a user can have. Subsequently, when the user requests a new connection, Steel-Belted Radius Carrier compares the current number of connections to the maximum limit.

If a new connection would exceed the limit, Steel-Belted Radius Carrier can reject the additional connection or allow the connection, but log the event in the server log (described in [“Using the Server Log File” on page 850](#)).

NOTE: When counting connections, Steel-Belted Radius Carrier does not distinguish between multi-link connections and new user authentication attempts.

For concurrent connection limits to work, each NAS must be configured for RADIUS accounting and the same Steel-Belted Radius Carrier server must be responsible for both authentication and accounting. These conventions give the server full access to the data it needs to track connections accurately.

The maximum number of concurrent connections can be set individually for any type of user by selecting the **Maximum Concurrent Connections** check box and entering a number in the accompanying field in the appropriate Add User/Edit User dialog box. See [“Administering Users” on page 120](#).

NOTE: This concurrency is based on username only. The optional Session State Register module also supports attribute-based concurrency which is not limited to username only. For more information see [“Managing Concurrency with Attributes in Session State Register” on page 688](#).

Authentication methods that do not require user entries must provide alternate mechanisms for supporting concurrent connection limits. For example, if you are using external database authentication there is an alias mechanism you can use in the SQL or LDAP configuration file. Concurrent connection limits can be supported under proxy authentication only if the target server supports them.

Concurrent Tunnel Connections

Steel-Belted Radius Carrier uses its Current Sessions Table (CST) list to determine the number of active connections for each tunnel. The CST list summarizes all of the RADIUS accounting data currently available to the server. Tunnel connections appear in the CST list using a special display convention that distinguishes them from user connections.

You can set a maximum limit on the number of concurrent connections that can be open using a specific tunnel. Subsequently, when a user requests a new connection through that tunnel, Steel-Belted Radius Carrier compares the current number of connections to the maximum limit. If a new connection would exceed the limit, Steel-Belted Radius Carrier rejects the additional connection.

NOTE: Concurrent tunnel connections are supported locally only.

For concurrent connection limits to work, it is essential that each NAS that can open a tunnel be configured for RADIUS accounting and that the same Steel-Belted Radius Carrier server be specified for both authentication and accounting and this applies only to non-cluster. For cluster installations, authentication and accounting must be sent to servers within the same cluster. This permits the server's CST list to be kept up to date and available to every NAS that needs to authenticate a tunnel connection.

NOTE: Concurrent tunnel connections cannot be tracked across multiple Steel-Belted Radius Carrier servers without additional software extensions. Contact Juniper Networks for more information.

Attribute Value Pooling

Attribute value pooling lets you define pools of attribute sets that are assigned dynamically when an Authorization Request is processed. The attribute sets are distributed according to specified weights and the values are returned with the Access-Accept message.

Attribute value pooling allows for a dynamic allocation of attribute values sets, so that attributes needed to configure changeable and complex situations do not have to be assigned in static profiles. This functionality is supported by the use of a vendor-specific attribute called **Funk-Round-Robin-Group**. The value for this attribute is a string, and should be set to the name of an **.rr** suffix file that defines an attribute value pool.

An **.rr** file is defined as follows:

```
[Sets]
SetName1 = Weight1
SetName2 = Weight2
.
.
.

[ SetName1 ]
AttributeName1.1 = AttributeValue1.1
AttributeName1.2 = AttributeValue1.2
.
.
.
```

Steel-Belted Radius Carrier maintains *round-robin* statistics for each attribute value pool so that weight calculations can be performed properly. When a user who belongs to a profile that has been assigned to a particular attribute value pool logs in, the round-robin values are incremented to determine which Attribute Value set should be assigned to the user. This attribute set is added to the return list of the Access-Accept.

Attribute value pooling can be used in several ways. For example, the Acme Company wants off-site employees to be able to establish tunnels to the company network. The Acme Company maintains three tunnel connection endpoints to which end users can create VPNs into the corporate network, each of these with different capacities. The Acme Company would define an attribute value pool of three attribute sets, each describing how to establish a tunnel with one of these connection points. These attribute sets should be weighted according to the capacity of the three connection points.

The following is a sample **acme.rr** file.

```
;acme.rr
[Sets]
VPN1=20
VPN2=12
VPN3=7
```



```
[VPN1]
Tunnel-Server-Endpoint = 8.4.2.1
Tunnel-Password = GoodGuess

[VPN2]
Tunnel-Server-Endpoint = 8.4.2.2
Tunnel-Password = BestGuess

[VPN3]
Tunnel-Server-Endpoint = 8.4.2.4
Tunnel-Password = OurSecret
```

To make this attribute value pool visible, the Acme Company would define a **Funk-Round-Robin-Group** VSA and assign it to the users (or the profile assigned to these users) and make the value of the VSA point to the **acme.rr** file.

```
Funk-Round-Robin-Group = acme.rr
```

Attribute value pools can be combined with other features. For example, by specifying an IP Pool name for a **Framed-IP-Address** attribute, you could load-balance IP pools.

NOTE: Attribute merging rules do not apply to attributes in round-robin files. You must follow appropriate attribute usage (single-valued, multi-values, check list, and so on). No special checks are performed to ensure that the attributes and values specified in round-robin files are consistent with the rest of your system configuration. Check the dictionary file for information about attribute usage.

Attribute value pools can be reconfigured dynamically. When you issue the SIGHUP (1) signal to the Steel-Belted Radius Carrier process, the modified files are re-read and the pool configuration is reset appropriately.

NOTE: You can have multiple, active Round-Robin-Group attributes in use simultaneously.

Phantom Records

When Steel-Belted Radius Carrier allocates resources such as IP addresses, user connections, and tunnel connections, to its clients, it generates a *phantom* accounting record for its internal use. Phantom records are not written to the RADIUS accounting database, but they are displayed in the Current Sessions Table (CST) list (described on [“Current Sessions Overview” on page 718](#)). Phantom records resemble accounting start records, except that the session ID for phantom records is displayed as **N/A**.

After Steel-Belted Radius Carrier receives the corresponding accounting start request packet from the client, it discards the phantom record and replaces **N/A** with the actual **Session-ID** number returned by the client device in the Current Sessions list.

In some cases, Steel-Belted Radius Carrier can allocate a resource and create a phantom record, but then never receive a corresponding start packet from the client. To avoid committing the resource indefinitely, Steel-Belted Radius Carrier waits for a limited period for the start packet to confirm the transaction. By default, Steel-Belted Radius Carrier waits 180 seconds, though you can configure a different wait period by editing the **radius.ini** file (described in the *SBR Carrier Reference Guide*).

IPv6 Support

IPv6 is the next step in the evolution of the Internet Protocol, which provides many more unique addresses than the earlier IPv4, which is currently widely in use but has recently been exhausted. IPv6, which has been under development for more than 10 years, provides improvements over IPv4 in addressing, configuration, and security. Although IPv6 is still an evolving standard, many operating system vendors offer production-quality IPv6 implementations for customers interested in using IPv6 networks.

IPv6 and Steel-Belted Radius Carrier

With few exceptions, Steel-Belted Radius Carrier supports IPv6 addressing wherever IPv4 addressing is supported. You can perform configuration, authentication, and accounting of RADIUS IPv6 attributes per RFC 3162, *RADIUS and IPv6* and RFC 6911, *RADIUS Attributes for IPv6 Access Networks*. The Web GUI and the LDAP configuration interface (LCI) support the configuration of RADIUS IPv6 attributes.

NOTE: Because many third-party libraries and software development kits (SDKs) do not support IPv6, the Steel-Belted Radius Carrier server must support local IPv4 socket connections.

IPv6 Features

Significant changes from IPv4 to IPv6 include the following:

- Expanded routing and addressing capabilities—IPv6 increases the IP address size from 32 bits to 128 bits. As a consequence, IPv6 supports more levels of addressing hierarchy, provides a much greater number of available addresses, and simplifies auto-configuration of addresses. As a consequence, address conservation techniques such as network address translation, are not necessary.
- Improved multicast routing—Multicast routing, which existed in IPv4, has been redefined and improved in IPv6. Multicast addresses now include a Scope field that limits the scope of multicasts, improving scalability. A new Anycast address type allows you to send a message to the nearest single member of a multicast group.
- Header format simplification—The IPv6 packet header format has been designed to be efficient. The IPv6 header has a fixed length of 40 bytes, divided into eight fields.
- Improved support for extensions and options—IPv6 uses extension headers, which are inserted between the IPv6 header and the transport header and packet payload, to handle special packet processing requirements. Extension headers provide a flexible means to support authentication, encryption, fragmentation, source routing, network management, and other functions. An IPv6 packet can carry any number of extension headers.
- Improved datagram sizing and fragmentation—The maximum transmission unit (MTU) describes the maximum size of a datagram that can be transmitted over a network without fragmentation. IPv6 increases the minimum MTU from 576 bytes to 1280 bytes, which makes IPv6 packets more efficient and reduces the need for packet fragmentation. Path MTU discovery enables source routers to determine the appropriate packet size for a route.
- Quality-of-Service (QoS) functions—Packets belonging to a traffic flow that requires special handling, such as real-time video service, can be labeled by the sender. Because traffic in a particular flow can be identified in the IPv6 header, support for QoS can be implemented even when the payload of a packet is encrypted.
- Improved privacy and security—IPv6 supports extensions for authentication and data integrity to improve security and privacy of network traffic.

IPv6 Addressing

IPv6 addresses are 128 bits in length, which creates a much larger address space than 32-bit IPv4 addresses. IPv6 addresses identify individual interfaces and sets of interfaces. IPv6 addressing architecture is defined in RFC 2373, *IP Version 6 Addressing Architecture*.

Address Notation

In full form, IPv6 addresses are written as eight 16-bit hexadecimal blocks separated by colons:

```
FE80:0000:0000:0000:1232:E4BF:FE1A:8324
```

IPv6 addresses can be interpreted as having two variable-length fields: an IPv6 prefix and an IPv6 interface identifier.

- The IPv6 prefix varies from 0 to 128 bits in length and forms the routable network number portion of the IPv6 address. The trailing CIDR notation that may appear after human-readable IPv6 addresses (for example, /64) indicates the bit length of the IPv6 prefix.
- The IPv6 interface identifier consists of the non-prefix portion of the IPv6 address, if any, and identifies the host interface portion of the address, which identifies an IPv6 interface on the local network. The IPv6 interface identifier is typically generated automatically by the host as a function of a unique hardware identifier, such as an Ethernet MAC address. IPv6 hosts can automatically configure interface addresses by combining the IPv6 prefixes obtained from router advertisements with the IPv6 interface IDs that are determined locally.

For example:

```
IPv6 Prefix: FEC0:0000:0000:0000:0000:0000:0000:0000/64
IPv6 Interface ID: 0260:08FF:FEFF:FFFF
IPv6 Address: FEC0:0000:0000:0000:0260:08FF:FEFF:FFFF
```

To simplify address notation, IPv6 accepts abbreviations in address notation. For example, leading zeros in a 16-bit block may be omitted:

```
FE80:0:0:0:0232:E4BF:FE1A:8324
```

A double colon (::) can replace a series of consecutive zeros within an address:

```
FE80::232:E4BF:FE1A:8324
```

Only one double colon can be used to compress an IPv6 address. If more than one double colon was included in an address, networking devices would not know how many zeros to insert for each double colon when expanding a compressed address to its full 128-bit representation.

In networks that support IPv4 and IPv6 nodes, IPv4 addresses can be embedded in the last four bytes of the address. An IPv4 address of 192.168.1.12 can be represented in IPv6 format as ::192.168.1.12, where :: represents a string of zeroes to pad the address to 128 bits.

Address Prefixes

Like IPv4 addresses, IPv6 addresses are composed of a routable network number, known as the *IPv6 prefix*, and a host identifier, known as the *IPv6 interface ID*. IPv6 does not support address classes; IPv6 uses Classless Inter-Domain Routing with variable length network numbers, or prefixes, meaning that an IPv6 prefix is specified by supplying a bit length in conjunction with the address.

IPv6 prefixes are written as an IPv6 address, followed by a slash and the bit length of the prefix portion of the address. The prefix can be 0–128 bits in length, but the prefix is always written in terms of a 128-bit address. When writing prefixes, the trailing bits of the address comprising the interface ID are sometimes

dropped so that the prefix can be abbreviated. The following prefixes are equivalent, assuming that the interface ID portion of the address may be ignored:

Canonical form: **2001:1c44:820d:eea0:0260:08ff:feff:ffff/64**

Abbreviated form: **2001:1c44:820d:eea0::/64**

In many cases, the interface ID portion of the address contains relevant information. A hierarchy of prefixes may reflect the assignment and reassignment of blocks of addresses to progressively smaller organizations. For example, in a typical hierarchy, the largest service providers are assigned the largest blocks of addresses and hence the shortest prefixes, called *Top Level Aggregator Identifiers (TLA IDs)*. The large service providers reassign blocks of addresses to smaller service providers by adding a few more bits after the TLA ID; these added bits are known as *Next Level Aggregator IDs*. The smaller service providers again reassign smaller blocks of addresses to end user organizations by again adding a few more bits after the NLA ID. These added bits are known as *Site Level Aggregator IDs*. The hierarchy continues in this way until the prefix is exhausted, leaving only the trailing bits that correspond to the non-routable IPv6 interface ID.

Steel-Belted Radius Carrier accepts all equivalent forms of IPv6 prefixes and recognizes them as being equivalent. The following prefixes are related by hierarchy, but only the canonical and abbreviated forms are equivalent:

Canonical form: **2001:1c44:820d:eea0:0260:08ff:feff:ffff/64**

Abbreviated form: **2001:1c44:820d:eea0::/64**

Site level prefix: **2001:1c44:820d::/48**

Next level prefix: **2001:1c44:8200::/40**

Top level prefix: **2001:1c00::/24**

IPv6 prefixes should always be written out in full and unabbreviated form when wildcards are used, as the abbreviations become ambiguous in the presence of wildcards. The following prefixes are equivalent:

Canonical form: **2001:1c44:820d:eea0:0260:08ff:feff:ffff/64**

With wildcards: **2001:1c*:??0d:eea0*:*:*/64**

Address Interface IDs

Because the overall size of the IPv6 address is fixed, a longer address prefix means a shorter interface ID. For example, a 48-bit prefix implies an 80-bit interface ID:

Canonical prefix: **2001:1c44:820d:eea0:0260:08ff:0000:0000/48**

Abbreviated prefix: **2001:1c44:820d::/48**

48-bit interface ID: **eea0:0260:08ff:0000:0000**

Though the interface ID portion of an IPv6 address can be 0–128 bits in length, the RADIUS standard assumes 64-bit interface IDs. Steel-Belted Radius Carrier uses a convention that all interface IDs are written as a series of four unabbreviated hexadecimal fields regardless of how they are entered:

64-bit interface ID: **0260:08ff:0000:0000**

IPv6 interface IDs should always be written out in full and unabbreviated form when wildcards are used, as the abbreviations become ambiguous in the presence of wildcards. The following interface IDs are equivalent:

64-bit interface ID: **0260:08ff:0000:0000**

With wildcards: **02?:08ff:?:***

NOTE: The use of wildcards in IPv6 interface IDs is a Steel-Belted Radius Carrier feature. Wildcards in IPv6 interface IDs are not known to be documented or prohibited by any IPv6 standards at this time.

IPv6 Network Numbers

In very specific cases, such as check list processing, Steel-Belted Radius Carrier recognizes both IPv4 and IPv6 addresses as representing entire *ranges* of addresses. Steel-Belted Radius Carrier extends the concept of IPv4 network numbers to IPv6 as a means of representing a range of network addresses. Using this concept of network numbers means you cannot specify a valid network address that also happens to be a network number.

Before the adoption of Classless Inter-Domain Routing (CIDR), the IPv4 address space is divided into five address classes, as shown in [Table 22 on page 88](#).

Table 22: IPv4 Address Classes

Class	Address Range	Description
A	0.0.0.0 – 127.255.255.255	1-bit class, 7-bit network, 24-bit host
B	128.0.0.0 – 191.255.255.255	2-bit class, 14-bit network, 16-bit host
C	192.0.0.0 – 223.255.255.255	3-bit class, 21-bit network, 8-bit host
D	224.0.0.0 – 239.255.255.255	4-bit class, 28-bit multicast group
E	240.0.0.0 – 247.255.255.255	5-bit class, 27-bit reserved

Within an IPv4 address class, each network is identified by a network number that consists of the leading class bits and the network bits that follow. Network numbers are typically written as IP addresses with

trailing zero bits; for example, the network number corresponding to the class C address 199.100.10.24 would typically be written as 199.100.10.0.

Each network represents a potential physical interconnection of a maximum number of hosts determined by the number of host bits. Thus, the physical network identified by the network number 199.100.10.0 might connect up to 256 hosts identified by the addresses 199.100.10.0 through 199.100.10.255 inclusive. (In practice, host addresses such as 199.100.10.0 are avoided to prevent confusion between host addresses and network numbers.) Thus, it is reasonable to interpret a network number as the entire range of addresses sharing the same network portion of the address:

Network number: **199.100.10.0**

Start of address range: **199.100.10.0**

End of address range: **199.100.10.255**

As a wildcarded address: **199.100.10.***

To see how the concept of network numbers can be extended to IPv6 addresses, consider that IPv6 addresses can contain embedded IPv4 addresses. The IPv6 address **::ffff:199.100.10.0** can therefore be interpreted as the range **::ffff:199.100.10.0** through **::ffff:199.100.10.255** inclusive.

The IPv6 address space is not divided into classes, because IPv6 was designed with CIDR in mind. Constructing an arbitrary definition of IPv6 network numbers that both resembles IPv4 and scales well across all possible IPv6 addresses is difficult. However, since large portions of the IPv6 address space have not yet been defined and since the RADIUS specification concerns itself only with 64-bit interface IDs, we can consider arbitrarily assigning special meaning to all IPv6 addresses ending in 64 bits of zero. This represents a fraction ($1/2^{64}$) of the IPv6 address space, where the addresses have arbitrarily been assigned special meaning overriding their true meaning. The cases where this arbitrary definition would cause trouble are expected to be extremely rare, and it should be possible to avoid them.

Steel-Belted Radius Carrier artificially defines the concept of IPv6 network numbers as IPv6 addresses ending in 64 bits of zero, where the network number is interpreted as the entire range of IPv6 addresses sharing the same 64-bit prefix as the network number:

Network number: **2001:1c44:820d:eea0:0000:0000:0000:0000**

Start of address range: **2001:1c44:820d:eea0:0000:0000:0000:0000**

End of address range: **2001:1c44:820d:eea0:ffff:ffff:ffff:ffff**

As a wildcarded address: **2001:1c44:820d:eea0:*.*.*.***

IPv6 Support in Steel-Belted Radius Carrier

In general, IPv6 support in Steel-Belted Radius Carrier parallels IPv4 support, both in terms of IPv6 network transport and in terms of RADIUS IPv6 attributes. When IPv6 networking is not supported by an operating

system (either because the operating system cannot support IPv6 or because the IPv6 stack for the operating system has not been configured), Steel-Belted Radius Carrier recognizes IPv6 addresses and attributes, but does not use IPv6 transport mechanisms.

Socket interfaces in Steel-Belted Radius Carrier are both IPv4- and IPv6-capable. By default, IPv4 support is enabled and IPv6 support is disabled in Steel-Belted Radius Carrier. You must explicitly enable IPv6 support (by modifying settings in the [IPv6] section of the **radius.ini** file) before you can use IPv6 networking. Steel-Belted Radius Carrier recognizes IPv6 attributes whether or not IPv6 networking is enabled.

With few exceptions, IPv6 addresses may be configured wherever you can configure IPv4 addresses in configuration files and in the Web GUI.

Similarly, IPv6 RADIUS attributes can be configured wherever IPv4 RADIUS attributes can be configured. All IPv6 attributes are defined in the **radius.dct** file to allow inclusion in all standards-conforming dictionaries. IPv6 attributes are correctly interpreted and fully validated by the LDAP Configuration Interface (LCI) and Web GUI.

[Table 23 on page 90](#) presents a summary of IPv6 support in Steel-Belted Radius Carrier.

Table 23: IPv6 Feature Matrix

Feature	Supported?	Comments
Server networking	Yes	IPv6 networking must be explicitly enabled. IPv6 attributes can be processed even if IPv6 networking is not enabled. IPv6 network traffic is limited to the RADIUS authentication/accounting ports (typically 1645, 1646, 1812, and 1813). IPv6 addresses can be configured in the [Interfaces] section of the proxy.ini file. IPv6 link local and site local addressing is deprecated.
Server DNSv6	Yes	DNSv6 must be explicitly enabled. Both IPv6 and IPv4 network connections are supported with remote DNSv6 servers. Only IPv4 network connections are supported with remote DNS servers.
Server logs	Yes	The diagnostic logging and tracing of IPv6 network connections and IPv6 attributes are fully supported. Only RFC 3162 and RFC 6911 attributes have been tested.
LCI networking	Yes	If IPv6 networking is enabled, IPv6 addresses can be configured in the [LDAPAddresses] section of the radius.ini file.

Table 23: IPv6 Feature Matrix (continued)

Feature	Supported?	Comments
LCI inputs	Partial	In most cases, IPv6 values can be supplied wherever IPv4 inputs can be specified.
Attributes	Yes	Basic IPv6 attributes defined in RFC 3162, and RFC 6911 and listed in radius.dct are supported as native types or as regular text strings, as appropriate. Only RFC 3162 and RFC 6911 attributes have been tested.
Check lists	Partial	IPv6 attributes can appear in check lists, and IPv6 address values can contain network numbers similar to IPv4 address values. IPv6 prefix values and IPv6 interface values cannot be masked or wildcarded. Only RFC 3162 and RFC 6911 attributes have been tested.
Return lists	Yes	IPv6 attributes can appear within return lists, and IPv6 values can be assigned. Only RFC 3162 and RFC 6911 attributes have been tested.
Attribute value pools	Yes	IPv6 attributes can appear in attribute value pools, and IPv6 values can be assigned to implement round-robin return list processing. Only RFC 3162 attributes have been tested.
Attribute filtering	Yes	IPv6 attributes can appear in filter rules, and IPv6 values can be assigned. Only RFC 3162 attributes have been tested.
Service type mapping	Yes	IPv6 attributes can appear in a service type mapping, and IPv6 values can be wildcarded similar to IPv4 values. Reliable string comparison of regular expressions requires all values to be expressed in canonical form. Only RFC 3162 attributes have been tested.
Attribute mapping	Yes	IPv6 attributes can appear in an attribute mapping, and IPv6 values can be wildcarded similar to IPv4 values. Reliable string comparison of regular expressions requires all values to be expressed in canonical form. Only RFC 3162 attributes have been tested.

Table 23: IPv6 Feature Matrix (continued)

Feature	Supported?	Comments
Class attribute	Partial	The RADIUS Class attribute cannot contain any IPv6 attributes or structured attributes. You can configure IPv6 addresses in the [Hosts] section of the spl.ini file to process class attributes originating from IPv6 network connections.
DHCP	No	The use of IPv6 networking to communicate with any DHCP server is not supported. The allocation of IPv6 addresses obtained from any DHCP server is not supported.
IP address pools	No	<p>The allocation of IPv6 addresses from an SBR-managed IP address pool is supported through the Framed-IPv6-Prefix. For more information on using the managed IPv6 address pools, see <i>SBR Carrier Reference Guide</i>.</p> <p>However, RFC 3162 provides an attribute, Framed-IPv6-Pool, that allows the NAS to implement an IPv6 address pool.</p>
Networking for authentication	Yes	<p>IPv6 addresses can be configured in the [Addresses] section of the radius.ini file. IPv6 network traffic is limited to the RADIUS authentication/accounting ports (typically 1645, 1646, 1812, and 1813). Proxying to another RADIUS server is not supported. IPv6 link local and site local addressing is deprecated.</p>
Authentication logs	Yes	IPv6 attributes are fully supported. Only RFC 3162 and RFC 6911 attributes have been tested.
Local User authentication	Yes	IPv6 attributes are fully supported. Only RFC 3162 and RFC 6911 attributes have been tested.
Authenticate-Only requests	Yes	IPv6 attributes are fully supported. Only RFC 3162 and RFC 6911 attributes have been tested.

Table 23: IPv6 Feature Matrix (continued)

Feature	Supported?	Comments
Pass-through authentication	Partial	IPv6 attributes are fully supported. However, because many third-party libraries do not support IPv6, IPv6 networking is not necessarily supported with external services. Only RFC 3162 attributes have been tested. Third-party software may not support IPv6 networking or attributes.
External authentication (for example, LDAP or SQL)	Partial	IPv6 attributes are fully supported. However, because many third-party libraries do not support IPv6, IPv6 networking is not necessarily supported with external services such as LDAP and SQL. Only RFC 3162 and RFC 6911 attributes have been tested. Third-party software may not support IPv6 networking or attributes.
EAP-TTLS authentication	Yes	IPv6 attributes are fully supported. Only RFC 3162 and RFC 6911 attributes have been tested.
Directed authentication	Yes	IPv6 attributes are fully supported. Only RFC 3162 attributes have been tested.
Networking for accounting	Yes	IPv6 addresses can be configured in the [Addresses] section of the radius.ini file.
Accounting logs	Yes	IPv6 attributes are fully supported. Only RFC 3162 and RFC 6911 attributes have been tested.
External accounting (for example, SQL)	Partial	IPv6 attributes are fully supported. However, because many third-party libraries do not support IPv6, IPv6 networking is not necessarily supported with external services such as SQL. Only RFC 3162 and RFC 6911 attributes have been tested. Third-party software may not support IPv6 networking or attributes.
Directed accounting	Yes	IPv6 attributes are fully supported. Only RFC 3162 attributes have been tested.
Spooled accounting	Yes	IPv6 attributes are fully supported. Only RFC 3162 attributes have been tested.

Table 23: IPv6 Feature Matrix (continued)

Feature	Supported?	Comments
Networking for proxy	Yes	IPv6 addresses can be configured in the [Interfaces] section of the proxy.ini file. NOTE: This feature cannot be configured using the GUI.
Proxy authentication	Yes	IPv6 attributes are fully supported.
Proxy accounting	Yes	IPv6 attributes are fully supported.
Master SNMP agent	Yes	The use of IPv6 networking to communicate with an IPv6 capable SNMP management station or SNMP subagent is supported.
SNMP subagent	Yes	IPv6 networking, IPv6 trap variables, and IPv6 MIB objects are supported. IPv6 addresses are reported as IPv4 MIB objects possessing the value 255.255.255.255.
Networking for plug-ins		Steel-Belted Radius Carrier does not control the networking of back-end servers with its plug-ins. IPv6 networking is generally a hidden detail of third-party back-end server configuration.
Plug-in APIs	Yes	IPv6 features and parameters are exposed in the new plug-in APIs. The older plug-in APIs are deprecated but still functional. You should upgrade to the new plug-in APIs to gain access to IPv6 features. IPv6 APIs can be invoked even if IPv6 networking is not enabled.
Plug-in attributes	Yes	Basic IPv6 attributes defined in RFC-3162, RFC 6911 and listed in radius.dct are supported as native types or as regular text strings, as appropriate.
Oracle plug-ins	Yes	IPv6 attributes are fully supported, but the required third-party software may not support IPv6 connectivity.

Table 23: IPv6 Feature Matrix (continued)

Feature	Supported?	Comments
LDAP plug-in	Yes	IPv6 attributes are fully supported, but the required third-party software may not support IPv6 connectivity.
PEAP plug-in	Yes	IPv6 attributes are fully supported, but the required third party software may not support IPv6 connectivity.
TLS plug-in	Yes	IPv6 attributes are fully supported, but the required third-party software may not support IPv6 connectivity.
TTLS plug-in	Yes	IPv6 attributes are fully supported, but the required third-party software may not support IPv6 connectivity.
JDBC plug-in	Partial	(Linux only) IPv6 attributes are fully supported, but the required third-party software may not support IPv6 connectivity.

RADIUS IPv6 Attributes

All RADIUS attributes defined in RFC 3162, *RADIUS and IPv6* and RFC 6911, *RADIUS Attributes for IPv6 Access Networks*, are supported in Steel-Belted Radius Carrier as native types or as regular text strings. All forms of attribute processing, such check list processing, return list processing, attribute echoing/deleting/merging, are supported. However, IPv6 prefix values and IPv6 interface values cannot be masked or wildcarded in check list processing.

Third-party plug-ins that have not been upgraded to support IPv6 should be able to process IPv6 attributes as opaque hexadecimal strings.

[Table 24 on page 95](#) lists the attribute number, and the number of times an attribute can appear in an Access-Request, Access-Accept, Access-Reject, Access-Challenge, and Accounting-Request packets for each type of IPv6 RADIUS attribute.

Table 24: IPv6-Specific RADIUS Attributes

Attribute	Attr Num	Acc-Req	Acc-Accept	Acc-Rej	Acc-Chall	Acct-Req
NAS-IPv6-Address	95	0-1	0	0	0	0-1

Table 24: IPv6-Specific RADIUS Attributes (*continued*)

Attribute	Attr Num	Acc-Req	Acc-Accept	Acc-Rej	Acc-Chall	Acct-Req
Framed-Interface-Id	96	0-1	0-1	0	0	0-1
Framed-IPv6-Prefix	97	0+	0+	0	0	0+
Login-IPv6-Host	98	0+	0+	0	0	0+
Framed-IPv6-Route	99	0	0+	0	0	0+
Framed-IPv6-Pool	100	0	0-1	0	0	0-1
Framed-IPv6-Address	168	0+	0+	0	0	0+
DNS-Server-IPv6-Address	169	0+	0+	0	0	0+
Route-IPv6-Information	170	0+	0+	0	0	0+
Delegated-IPv6-Prefix-Pool	171	0+	0+	0	0	0+
Stateful-IPv6-Address-Pool	172	0+	0+	0	0	0+

NAS-IPv6-Address

The NAS-IPv6-Address attribute is similar in function to the NAS-IP-Address attribute. Either attribute is sufficient to identify the IP address of the requesting NAS to the RADIUS server. If both attributes appear in the same RADIUS Access-Request packet, Steel-Belted Radius Carrier processes the NAS-IPv6-Address attribute for the purpose of identifying the NAS.

The NAS-IPv6-Address attribute may be specified by the NAS in access and accounting request packets. Zero or one NAS-IPv6-Address attributes may be specified. If present, the fixed length NAS-IPv6-Address attribute contains the complete 128-bit IPv6 address of the requesting NAS.

Steel-Belted Radius Carrier allows zero or one 128-bit IPv6 address to be specified for each NAS. The server authentication logic validates these addresses on extraction from the database and compares them with NAS-IPv6-Address attributes when they are received in access and accounting request packets. The server accounting logic writes these addresses to the accounting logs in a human readable format.

Example

Human readable:fe80::260:8ff:feff:ffff

RADIUS attribute:5f 12 fe 80 00 00 00 00 00 02 60 08 ff fe ff ff ff

Framed-Interface-Id

The Framed-Interface-Id attribute specifies the IPv6 interface ID to be assigned to a user. When combined with a Framed-IPv6-Prefix attribute, a single Framed-Interface-Id attribute forms one or more complete IPv6 addresses to be assigned to the user.

In general, the user is assigned the number of addresses equal to the number of Framed-IPv6-Prefix attributes, where the addresses have the same Framed-Interface-Id value and different Framed-IPv6-Prefix values.

It is possible to assign complete IPv6 addresses using only Framed-IPv6-Prefix attributes (for example, without specifying any Framed-Interface-Id attribute). For example, in the case of PPP, it can be quite difficult to automatically generate a unique IPv6 interface ID for a given network segment, so we recommend that the RADIUS server honor the hint if this attribute is suggested by the NAS. This is typically accomplished with echo attributes.

The Framed-Interface-Id attribute may be specified by the NAS in Access- and Accounting-Request packets, and by the RADIUS server in Access-Accept packets. Zero or one Framed-Interface-Id attributes may be specified. If present, the fixed length Framed-Interface-Id attribute contains the 64-bit interface ID to be assigned to the user.

Steel-Belted Radius Carrier allows zero or one 64-bit interface ID to be specified for each local user. The server authentication logic validates this interface ID on extraction from the database and includes the Framed-Interface-Id attribute in Access-Accept packets if none is received in the Access-Request packets.

Example

```
Human readable:260:8ff:feff:ffff
RADIUS attribute:60 0a 02 60 08 ff fe ff ff ff
```

Framed-IPv6-Prefix

The Framed-IPv6-Prefix attribute specifies the IPv6 networks to be assigned to a user. When combined with a Framed-Interface-ID attribute, a single Framed-IPv6-Prefix attribute forms one or more complete IPv6 addresses to be assigned to the user.

In general, the user is assigned the number of addresses equal to the number of Framed-IPv6-Prefix attributes, where the addresses have the same Framed-Interface-Id value, but different Framed-IPv6-Prefix values.

It is possible to assign complete IPv6 addresses using only Framed-IPv6 Prefix attributes (that is, without specifying any Framed-Interface-Id attribute). For example, the Framed-IPv6-Prefix attributes may be suggested by the NAS and overridden by the RADIUS server. In any case, the NAS is expected to be able to plumb the routes implied by the Framed-IPv6-Prefix attributes and these routes need not be repeated in separate Framed-IPv6-Route attributes.

The Framed-IPv6-Prefix attribute may be specified by the NAS in Access- and Accounting-Request packets, and by the RADIUS server in Access-Accept packets. Zero or more Framed-IPv6-Prefix attributes may be

specified. If present, the variable-length Framed-IPv6-Prefix attribute contains an IPv6 prefix from 0 to 128 bits in length. Extra bits beyond the actual prefix length must be set to 0.

Steel-Belted Radius Carrier allows zero or more variable-length IPv6 prefixes to be specified for each local user or attribute profile. The server authentication logic validates these prefixes on extraction from the database and includes the proper number of Framed-IPv6-Prefix attributes in Access-Accept packets if none are received in the access request packets. The server accounting logic writes these prefixes to the accounting logs in a human readable format.

Example

```
Human readable:fe80::260:8ff:feff:ffff/10
RADIUS attribute:61 14 00 0a fe 80 00 00 00 00 00 00 00 00 00 00 00 00
(8-bit type) (8-bit length)
(8-bit zero) (8-bit prefix length) (128-bit IPv6 prefix)
```

Login-IPv6-Host

The Login-IPv6-Host attributes specify the IPv6 addresses of the systems with which the user is connected when the Login-Service attribute is also included. The Login-IPv6-Host attribute may be suggested by the NAS and overridden by the RADIUS server.

The Login-IPv6-Host attribute may be specified by the NAS in access and accounting request packets, and by the RADIUS server in Access-Accept packets. Zero or more Login-IPv6-Host attributes may be specified. If present, the fixed length Login-IPv6-Host attribute contains the complete 128-bit IPv6 address of the login host, or a special value:

- 128-bits set to 0 indicates that the NAS should select the login host for the user.
- 128-bits set to 1 indicates that the NAS should allow the user to select the login host.
- Other values indicate the actual 128-bit IPv6 address of the login host.

Steel-Belted Radius Carrier allows zero or more 128-bit IPv6 addresses (including special values) to be specified for each local user or attribute profile. The server authentication logic validates these addresses (including special values) on extraction from the database and includes the proper number of Login-IPv6-Host attributes in Access-Accept packets if none are received in the access request packets. The server accounting logic writes these addresses to the accounting logs in a human readable format.

Example

```
Human readable:fe80::260:8ff:feff:ffff
RADIUS attribute:62 12 fe 80 00 00 00 00 00 00 02 60 08 ff fe ff ff ff
```

Framed-IPv6-Pool

The Framed-IPv6-Pool attribute specifies the name of a NAS managed pool (as opposed to a RADIUS server managed pool) from which the NAS should assign an IPv6 prefix to the user. The Framed-IPv6-Pool attribute may not be suggested by the NAS and is always determined by the RADIUS server.

The Framed-IPv6-Pool attribute may be specified by the NAS in Accounting-Request packets, and by the RADIUS server in Access-Accept packets. Zero or one Framed-IPv6-Pool attributes may be specified. If present, the variable-length Framed-IPv6-Pool attribute contains the name of a NAS managed pool in human readable text. The text is not NULL terminated.

Steel-Belted Radius Carrier allows zero or one variable length pool name to be specified for each local user. The server authentication logic validates the pool name on extraction from the database and includes the proper number of Framed-IPv6-Pool attributes in Access-Accept packets. The server accounting logic writes these pool names to the accounting logs in a human readable format.

Example

```
Human readable:ipv6-pool
RADIUS attribute:64 0b 69 70 76 36 2d 70 6f 6f 6c
```

Framed-IPv6-Route

The Framed-IPv6-Route attribute specifies the IPv6 routing information to be configured for the user on the NAS. The NAS is expected to be able to plumb the routes specified by the Framed-IPv6-Route attributes in addition to those that may already be implied by separate Framed-IPv6-Prefix attributes. The Framed-IPv6-Route attribute may not be suggested by the NAS and is always determined by the RADIUS server.

The Framed-IPv6-Route attribute may be specified by the NAS in accounting request packets, and by the RADIUS server in Access-Accept packets. Zero or more Framed-IPv6-Route attributes may be specified. If present, the variable-length Framed-IPv6-Route attribute contains IPv6 routing information in human readable text. The text is not NULL terminated. The format of the text (destination prefix, gateway address, metrics) is described in RFC-3162.

Steel-Belted Radius Carrier allows zero or more variable-length IPv6 routes to be specified for each local user or attribute profile. The server authentication logic validates these routes on extraction from the database and includes the proper number of Framed-IPv6-Route attributes in Access-Accept packets. The server accounting logic writes these routes to the accounting logs in a human readable format.

Example

```
Human readable:2000:0:0:106::/64 2000::106:a00:20ff:fe99:a998 1
RADIUS attribute:63 32 32 30 30 30 3a 30 3a 30 ... 39 3a 61 39 39 38 20 31
```

Framed-IPv6-Address

The Framed-IPv6-Address attribute is a single-value attribute indicating the IPv6 address of the user. The Framed-IPv6-Address attribute may be specified by the NAS in Access-Request and Accounting-Request packets, and by the RADIUS server in Access-Accept packets. This attribute may be used in an Access-Request packet as a hint by the NAS to the RADIUS server that it would prefer this IPv6 address, but the RADIUS server is not required to honor the hint.

SBR Carrier allows zero or one 128-bit IPv6 address to be specified for each user on the NAS. The server authentication logic validates these addresses on extraction from the database and compares them with the Framed-IPv6-Address attributes when they are received in access and accounting request packets. The server accounting logic writes these addresses to the accounting logs in a human readable format. The sessions are created with a single instance of the Framed-IPv6-Address.

The length of the Framed-IPv6-Address attribute must be 18 bytes.

NOTE: You cannot define multiple instances of Framed-IPv6-Address attributes in a return list. The Framed-IPv6-Address attribute can appear only once in a return list.

Example

Human readable: fe80::260:8ff:feff:ffff
 RADIUS attribute: 5f 12 fe 80 00 00 00 00 00 02 60 08 ff fe ff ff ff
 (8-bit type) (8-bit length) (128-bit IPv6 address)

DNS-Server-IPv6-Address

The DNS-Server-IPv6-Address attribute contains the IPv6 address of a DNS server. This attribute may be included multiple times in Access-Accept packets through the return list when the intention is for a NAS to announce more than one DNS server address to an RG/host. The DNS-Server-IPv6-Address attribute may be used in an Access-Request packet as a hint by the NAS to the RADIUS server regarding the DNS IPv6 address, but the RADIUS server is not required to honor the hint. The format of the Address field is the same as that of the corresponding field in the NAS-IPv6-Address attribute.

The length of the DNS-Server-IPv6-Address attribute must be 18 bytes.

Example

Human readable: 2007::1
 RADIUS attribute: 20 07 00 00 00 00 00 00 00 00 00 00 00 00 00 01
 (8-bit type) (8-bit length) (128-bit IPv6 address)

Route-IPv6-Information

The Route-IPv6-Information attribute specifies a prefix for the user on the NAS, which is announced using the Route Information option. This attribute is used in the Access-Accept packet and can appear multiple times. The Route-IPv6-Information attribute may be used in an Access-Request packet as a hint by the NAS to the RADIUS server, but the RADIUS server is not required to honor the hint.

The length of the Route-IPv6-Information attribute ranges from 4 bytes through 20 bytes.

Example

Human readable: 2006::1/64
 RADIUS attribute: 00 40 20 06 00 00 00 00 00 00 00 00 00 00 00 01

Delegated-IPv6-Prefix-Pool

The Delegated-IPv6-Prefix-Pool attribute contains the name of an assigned pool that should be used to select an IPv6 delegated prefix for the user on the NAS. If a NAS does not support prefix pools, the NAS must ignore this attribute. This attribute may be used in an Access-Request packet as a hint by the NAS to the RADIUS server regarding the pool, but the RADIUS server is not required to honor the hint.

The length of the Delegated-IPv6-Prefix-Pool attribute ranges from 3 bytes through 255 bytes.

Example

Human readable: IPv6_Prefix_POOL_2
 RADIUS attribute: 49 50 76 36 5f 50 72 65 66 69 78 5f 50 4f 4f 4c 5f 32

Stateful-IPv6-Address-Pool

The Stateful-IPv6-Address-Pool attribute contains the name of an assigned pool that should be used to select an IPv6 address for the user on the NAS. If a NAS does not support address pools, the NAS must ignore this attribute. The Stateful-IPv6-Address-Pool attribute may be used in an Access-Request packet as a hint by the NAS to the RADIUS server regarding the pool, but the RADIUS server is not required to honor the hint.

The length of the Stateful-IPv6-Address-Pool attribute ranges from 3 bytes through 255 bytes.

Example

Human readable: IPv6_POOL_1
 RADIUS attribute: 49 50 76 36 5f 50 4f 4f 4c 5f 31

Enabling IPv6 Networking

To enable IPv6 networking in Steel-Belted Radius Carrier, you must modify the **radius.ini** file and then restart your Steel-Belted Radius Carrier server. For information about the settings in the **radius.ini** file, refer to the *SBR Carrier Reference Guide*.

Steel-Belted Radius Carrier can process IPv6 attributes even if IPv6 networking is not enabled, provided that the IPv6 attributes are described in the RADIUS dictionary files.

Configuring IPv6 Scope IDs

Some types of IPv6 addresses require an IPv6 scope ID to avoid address ambiguity. In some cases, the Steel-Belted Radius Carrier server can select a scope ID automatically. You can specify the scope ID explicitly for IPv6 link local and site local addresses.

The [IPv6] section of the **radius.ini** file can specify how scope IDs are selected for each IPv6 address type that is recognized by the server. If the parameter value is 0, the Steel-Belted Radius Carrier server selects a scope ID automatically. If the parameter value is not 0, then the Steel-Belted Radius Carrier server uses that value as the scope ID when establishing network connections involving that IPv6 address type.

NOTE: You can use the output of the **ifconfig -a** shell command on Solaris platforms to determine the proper host specific scope ID for an address type. The scope ID is identical to the interface index on which the address type is supported and on which the desired destinations are reachable. On Solaris platforms, the server accepts traditional interface names, such as **hme0**, instead of numeric scope IDs.

Configuring IPv6 Addresses for RADIUS Client Connections

You can configure the [Addresses] section of the **radius.ini** file if you want to specify the local address or addresses on which Steel-Belted Radius Carrier listens for RADIUS client connections. By default, Steel-Belted Radius Carrier automatically discovers and configures all available IPv4 interfaces on the local host. If IPv6 is enabled, Steel-Belted Radius Carrier discovers and configures both IPv4 and IPv6 interfaces.

You can configure Steel-Belted Radius Carrier to configure IPv4 automatically by entering **AutoConfigureIPv4** in the [Addresses] section. Similarly, you can configure Steel-Belted Radius Carrier to configure IPv6 automatically by entering **AutoConfigureIPv6** in the [Addresses] section. If you configure specific IPv4 or IPv6 addresses, Steel-Belted Radius Carrier listens for RADIUS traffic on only those interfaces.

The IPv6 unspecified address **::** allows connections on any IPv6 address or IPv4 address, with IPv4 connections represented as IPv6 IPv4 mapped addresses.

Configuring DNSv6 Support

Enabling Domain Name Service Version 6 (DNSv6) support lets Steel-Belted Radius Carrier communicate with a DNSv6 server to resolve hostnames. By default, Steel-Belted Radius Carrier uses DNS unless IPv6 is enabled and DNSv6 support is configured by means of the **DynamicNameResolution** parameter in the [IPv6] section of the **radius.ini** file.

- If `DynamicNameResolution` is set to 0, Steel-Belted Radius Carrier uses IPv4 DNS, which means it does not query DNSv6 services.
- If `DynamicNameResolution` is set to 1, Steel-Belted Radius Carrier uses IPv6 DNS (DNSv6), which means it does not query IPv4 DNS services and ignores IPv4-specific information returned by DNSv6 services.
- If `DynamicNameResolution` is set to 2, Steel-Belted Radius Carrier queries DNSv6 services for IPv6-specific information, and then queries IPv4 DNS services for IPv4 specific information if DNSv6 fails to resolve a hostname.

Administering RADIUS Clients and Client Groups

IN THIS CHAPTER

- Overview | 104
- Adding a RADIUS Client or Client Group | 104
- Editing a RADIUS Client or Client Group | 112
- Deleting a RADIUS Client or Client Group | 113

This chapter describes how to set up RADIUS clients and client groups. This chapter contains these topics:

Overview

A RADIUS client is a network device or software application that contacts SBR Carrier when it needs to authenticate a user or to record accounting information about a network connection.

A RADIUS client group is a collection of network devices or software applications that contacts SBR Carrier to authenticate a user or to record accounting information about a network connection. Members of a RADIUS client group use a contiguous range of IP addresses and use identical settings, such as a shared secret or an IP address pool.

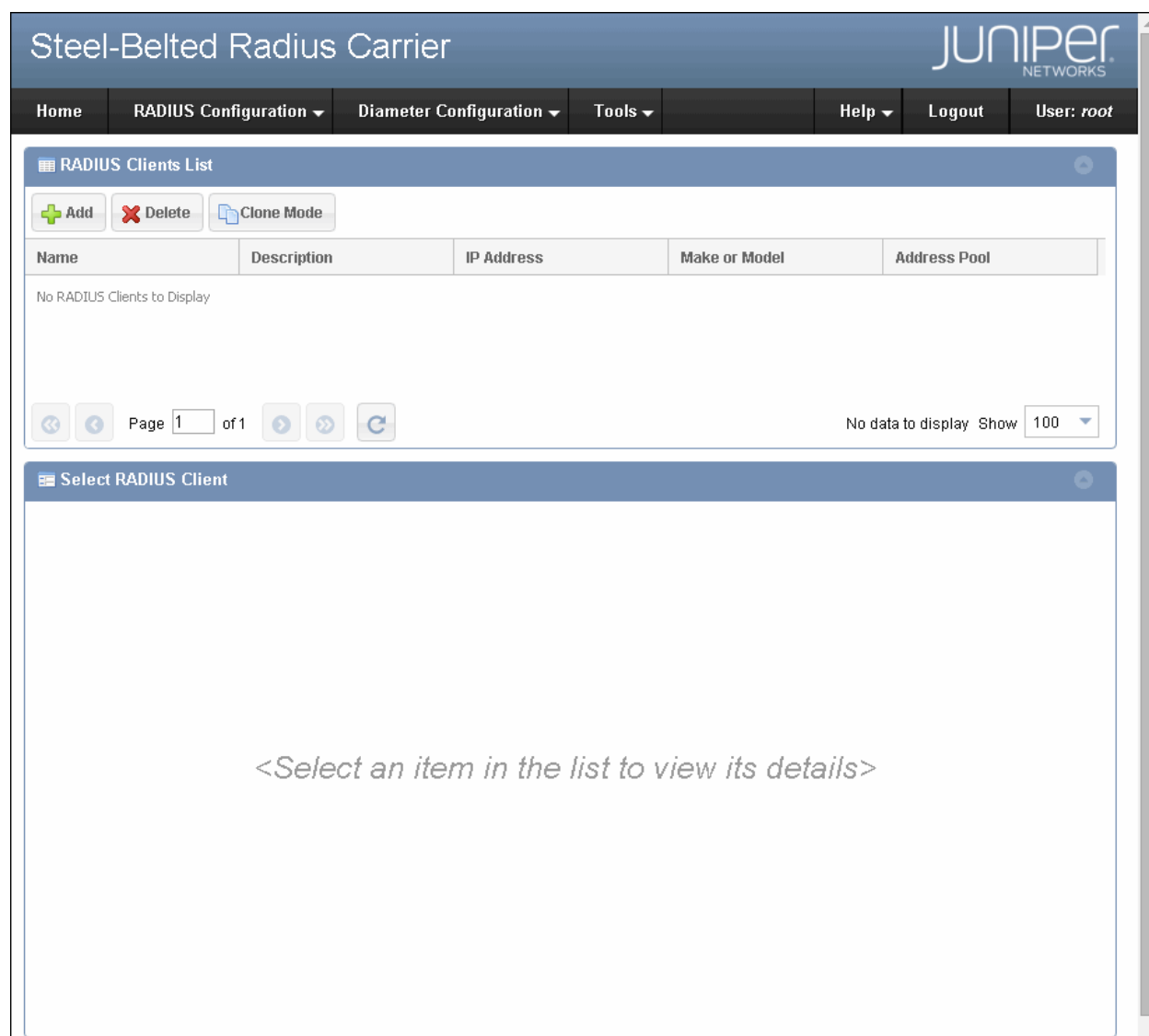
Adding a RADIUS Client or Client Group

To add a RADIUS client or client group using the Web GUI:

1. Select **RADIUS Configuration > RADIUS Clients**.

The **RADIUS Clients List** page (Figure 19 on page 105) appears.

Figure 19: RADIUS Clients List Page



2. Click **Add**.

The **Create RADIUS Client** pane (Figure 20 on page 106) appears with the **Basic Configuration** tab selected.

Figure 20: Create RADIUS Client Pane—Basic Configuration

The screenshot shows the 'Steel-Belted Radius Carrier' web interface. At the top is a navigation bar with links: Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. Below this is a 'RADIUS Clients List' section. The main area is titled 'Create RADIUS Client' and contains four tabs: Basic Configuration, Profiles, Diameter Configuration, and Advanced Configuration. The 'Basic Configuration' tab is active and contains the following fields and controls:

- Name:** A text input field.
- Description:** A text input field.
- IP Address:** A text input field.
- Range:** A checkbox and a numeric input field (currently showing '1').
- Shared Secret:** A text input field with a 'Show' button next to it.
- Make or Model:** A dropdown menu currently set to '- Standard Radius -'.
- Address Pool:** A checkbox and a dropdown menu.
- Location Group:** A checkbox and a dropdown menu.
- Any RADIUS Client:** A checkbox located to the right of the Name field.
- Use IPv6:** A checkbox located to the right of the IP Address field.

At the bottom of the pane are three buttons: Save, Clear, and Cancel.

3. Enter the name for the RADIUS client or client group in the **Name** field.

Although you can assign any name to a RADIUS client entry, you should use the device's IP address or DNS hostname to avoid confusion.

You can create a special RADIUS client entry called **<ANY>** by selecting the **Any RADIUS Client** check box. The **<ANY>** RADIUS client allows SBR Carrier to accept requests from any NAD or proxy RADIUS server, as long as the shared secret is correct.

The **IP Address** field for the **<ANY>** RADIUS client cannot be edited. **<ANY>** implies that the server accepts requests from any IP address, provided that the shared secret is correct.

NOTE: COA or DM messages do not work for the **<ANY>** RADIUS client. For COA or DM to work, the RADIUS client must have a unique name.

4. Optionally, enter a description for the RADIUS client in the **Description** field.

The description you associate with a RADIUS client is not used during processing.

5. Enter the IP address for the RADIUS client in the **IP Address** field.
6. Optionally, select the **Use IPv6** check box to use IPv6 addressing.

7. If you want the RADIUS client to use an IP address range, enter the starting address for the range in the **IP Address** field, select the **Range** check box, and enter the number of addresses in the **Range** field. You can create an address range supporting a maximum of 500 addresses.

For more information about IPv4 address ranges for RADIUS clients, see [“Radius Client Groups” on page 73](#).

8. Enter the authentication shared secret for the RADIUS client in the **Shared Secret** field.

For privacy, characters are masked. You can click **Show** to display the characters in the shared secret. After viewing the characters, you can click **Hide** to hide the characters.

After you configure the authentication shared secret on the server side, you must enter the same authentication shared secret when you configure the NAD.

9. Use the **Make or Model** list to select the make and model of your RADIUS client device.

The make or model selection determines which attribute dictionary SBR Carrier uses when communicating with this client. If you are not sure which make and model you are using or if your device is not in the list, leave the default - **Standard Radius** - selection in the **Make or Model** list.

10. If you want the RADIUS client to obtain IPv4 and IPv6 addresses from an address pool, select the **Address Pool** check box and use the **Address Pool** list to specify which address pool to use when returning an address in an access-accept to this RADIUS client.

NOTE: You must configure IP address pools before you set up RADIUS clients if you want the clients to use address pools. For more information, see [“Administering Address Pools” on page 182](#).

For more information about the usage of Framed-IPv6-Prefix for assigning addresses from pools, see the *SBR Carrier Reference Guide*.

11. If you want to associate the RADIUS client with a location group, select the **Location Group** check box and use the **Location Group** list to specify the location group to which the RADIUS client belongs.

NOTE: You must configure RADIUS location groups before you set up RADIUS clients if you want the clients to use location group profiles. For more information, see [“Administering RADIUS Location Groups” on page 114](#).

12. If you want to associate a profile with the RADIUS client, click the **Profiles** tab ([Figure 21 on page 108](#)), select the **Use Profile** check box, and use the drop-down list to select the profile you want the RADIUS client to use.

Figure 21: Create RADIUS Client Pane—Profiles

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and a User: root session. Below this, there are two tabs: 'RADIUS Clients List' and 'Create RADIUS Client'. The 'Create RADIUS Client' pane is active and contains four sub-tabs: 'Basic Configuration', 'Profiles', 'Diameter Configuration', and 'Advanced Configuration'. The 'Profiles' tab is selected, displaying a 'Use Profile' checkbox, a dropdown menu, and two sections: 'Attribute Combination' with 'Merge' (selected) and 'Override' radio buttons, and 'Merge Precedence' with 'User' (selected) and 'RADIUS Client' radio buttons. At the bottom of the pane are 'Save', 'Clear', and 'Cancel' buttons.

NOTE: The **Profiles** tab is disabled if you have selected the **Location Group** check box.

13. Specify how you want the profile to interact with the user settings:

- If you want attributes in the profile to override identically-named attributes configured for the user, select the **Override** option button.
- If you want attributes in the profile to be merged with identically-named attributes configured for the user, select the **Merge** option button and then select either **User** or **RADIUS Client** option button to take precedence if the attributes in the profile specify different values for the same single-value or ordered-multiple-value attribute.

14. Optionally, click the **Diameter Configuration** tab ([Figure 22 on page 109](#)) to specify whether you want to enable RADIUS to Diameter translation, which converts RADIUS authentication or authorization requests to Diameter authentication or authorization requests.

NOTE: The **Diameter Configuration** tab is available only if you have installed a valid Diameter license.

Figure 22: Create RADIUS Client Pane—Diameter Configuration

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes the title 'Steel-Belted Radius Carrier' and the Juniper Networks logo. Below the navigation bar, there are tabs for 'Home', 'RADIUS Configuration', 'Diameter Configuration', 'Tools', 'Help', 'Logout', and 'User: root'. The main content area is titled 'Create RADIUS Client' and contains four sub-tabs: 'Basic Configuration', 'Profiles', 'Diameter Configuration', and 'Advanced Configuration'. The 'Diameter Configuration' tab is currently selected. It features a checkbox labeled 'Enable Diameter Conversion'. Below this checkbox is a 'Policy' section with two radio button options: 'Use Diameter Conversion Always for this NAS' (which is selected) and 'Use Diameter Conversion for the User with Specific Profile'. A dropdown menu is visible below the second radio button option. At the bottom of the pane are three buttons: 'Save', 'Clear', and 'Cancel'.

Select the **Enable Diameter Conversion** check box to enable the fields in the **Policy** area.

- If you want to always enable RADIUS to Diameter translation for all the RADIUS requests that come to the RADIUS client, select the **Use Diameter Conversion Always for this NAS** check box.
- If you want to enable RADIUS to Diameter translation only for the user with a specific profile, select the **Use Diameter Conversion for the User with Specific Profile** check box and use the drop-down list to specify the profile.

15. Optionally, click the **Advanced Configuration** tab ([Figure 23 on page 110](#)) to configure advanced settings for the RADIUS client.

Figure 23: Create RADIUS Client Pane—Advanced Configuration

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and a User: root session. Below this, there are two tabs: 'RADIUS Clients List' and 'Create RADIUS Client'. The 'Create RADIUS Client' pane is active, and within it, the 'Advanced Configuration' tab is selected. This tab contains several configuration fields: a checkbox for 'Use different Shared Secret for Accounting' with a 'Show' button; a dropdown for 'RFC 3576 CoA/DM Port'; a text field for 'RFC 3576 CoA/DM Shared Secret' with a 'Show' button; a dropdown for 'POD Port'; and a text field for 'POD Shared Secret' with a 'Show' button. At the bottom of the pane are 'Save', 'Clear', and 'Cancel' buttons.

16. Optionally, specify an accounting secret for the RADIUS client. By default, SBR Carrier uses the same shared secret for authentication and accounting.

If you want the RADIUS client to use different shared secrets for authentication and accounting, select the **Use different Shared Secret for Accounting** check box and enter the shared secret you want the RADIUS client to use for accounting in the **Use different Shared Secret for Accounting** field.

For privacy, characters are masked. You can select the **Show** check box to display the characters in the shared secret. After viewing the characters, you can click **Hide** to hide the characters.

NOTE: You must enter the same accounting shared secret when you configure the RADIUS client.

17. Optionally, if you have purchased the Session Control module license, specify the number of the UDP port you want the SBR Carrier to use for COA and DM messages in the **RFC 3576 CoA/DM Port** field. The default UDP port is 3799.

NOTE: COA or DM and Packet of Disconnect (POD) messages do not work for the <ANY> RADIUS client.

18. Specify the shared secret used to authenticate COA and DM messages in the **RFC 3576 CoA/DM Shared Secret** field.

For privacy, characters are masked. You can click **Show** to display the characters in the shared secret. After viewing the characters, you can click **Hide** to hide the characters.

NOTE: After you configure the COA or DM shared secret on the server side, you must enter the same COA or DM shared secret when you configure the NAD.

NOTE: If a NAD client is configured without saving the shared secret, you are prompted to enter the shared secret when the client is subsequently viewed. If unexpected results such as invalid signatures occur, ensure that the shared secret is set correctly.

19. Specify the number of ports you want the SBR Carrier server to use for POD messages in the **POD Port** field.

20. Specify the shared secret used to authenticate POD messages in the **POD Shared Secret** field.

For privacy, characters are masked. You can click **Show** to display the characters in the shared secret. After viewing the characters, you can click **Hide** to hide the characters.

NOTE: After you configure the POD shared secret on the server side, you must enter the same POD shared secret when you configure the NAD.

NOTE: If a NAD client is configured without saving the shared secret, you are prompted to enter the shared secret when the client is subsequently viewed. If unexpected results such as invalid signatures occur, ensure that the shared secret is set correctly.

21. Click **Save** to save the RADIUS client or client group configuration.

The **RADIUS Clients List** page ([Figure 19 on page 105](#)) displays an updated list of RADIUS client or client group entries.

Editing a RADIUS Client or Client Group

To edit a RADIUS client or client group using the Web GUI:

1. Select **RADIUS Configuration > RADIUS Clients**.

The **RADIUS Clients List** page (Figure 19 on page 105) appears.

2. Select the RADIUS client entry that you want to edit.

The **Selected RADIUS Client** pane (Figure 24 on page 112) displays the settings configured for the RADIUS client.

Figure 24: Selected RADIUS Client Pane

The screenshot displays the Steel-Belted Radius Carrier Web GUI. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user 'root'. The main content area is divided into two sections. The top section, 'RADIUS Clients List', shows a table with one client entry: '81MACHINE' with IP address '10.13.20.81' and Make or Model '- Standard Radius -'. Below the table are pagination controls showing 'Page 1 of 1' and 'Displaying 1 - 1 of 1'. The bottom section, 'Selected RADIUS Client: 81MACHINE', contains a configuration form with tabs for Basic Configuration, Profiles, Diameter Configuration, and Advanced Configuration. The 'Basic Configuration' tab is active, showing fields for Name (81MACHINE), Description, IP Address (10.13.20.81), Range (1), Shared Secret (masked), Make or Model (- Standard Radius -), Address Pool, and Location Group. There are 'Show' and 'Validate' buttons next to the Shared Secret field. At the bottom of the form are 'Save', 'Reset', and 'Cancel' buttons.

Name	Description	IP Address	Make or Model	Address Pool
81MACHINE		10.13.20.81	- Standard Radius -	

Page 1 of 1 Displaying 1 - 1 of 1 Show 100

Selected RADIUS Client: 81MACHINE

Basic Configuration Profiles Diameter Configuration Advanced Configuration

Name: 81MACHINE

Description:

IP Address: 10.13.20.81 ☐ Use IPv6

Range: ☐ 1

Shared Secret: ***** Show Validate

Make or Model: - Standard Radius -

Address Pool: ☐

Location Group: ☐

Save Reset Cancel

3. Edit the settings for the RADIUS client entry as appropriate.

For information about the fields in the **Selected RADIUS Client** pane, see [“Adding a RADIUS Client or Client Group” on page 104](#). Additionally, the **Selected RADIUS Client** pane displays a **Validate** button in the right-hand side of the **Shared Secret**, **Use different Shared Secret for Accounting**, **RFC 3576 CoA/DM Shared Secret**, and **POD Shared Secret** fields. You can click **Validate** to verify the entered shared secret. The Web GUI displays a message indicating whether the entered shared secret matches the shared secret previously stored in the database.

NOTE: You cannot edit the name of the RADIUS client.

4. Click **Save** to save the changes.

The **RADIUS Clients List** page ([Figure 19 on page 105](#)) displays an updated list of RADIUS client entries.

Deleting a RADIUS Client or Client Group

To delete a RADIUS client or client group using the Web GUI:

1. Select **RADIUS Configuration > RADIUS Clients**.

The **RADIUS Clients List** page ([Figure 19 on page 105](#)) appears.

2. Select the RADIUS client entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to confirm the delete request.

The **RADIUS Clients List** page ([Figure 19 on page 105](#)) displays an updated list of RADIUS client entries.

Administering RADIUS Location Groups

IN THIS CHAPTER

- About RADIUS Location Groups | 114
- Adding a Location Group | 115
- Editing a Location Group | 117
- Deleting a Location Group | 119

This chapter describes how to set up RADIUS location groups. This chapter contains these topics:

About RADIUS Location Groups

RADIUS location groups allow you to assign an attribute profile to a user based on the network access server through which the user is connecting to your network. You can specify that users must use only the attributes specified in the profile associated with the location group, or you can specify that attributes from the NAD profile are merged with attributes from the user's profile.

To simplify administration of RADIUS client profiles, you can associate profiles with location groups and then associate location groups with RADIUS clients.

If you set the **AddFunkLocationGroupIdToRequest** parameter in the **[Configuration]** section of the **radius.ini** file to 1 (**AddFunkLocationGroupIdToRequest** = 1), if an inbound RADIUS authentication or accounting request is matched to a location group, then a **Funk-Location-Group-Id** attribute with a value set to the name of the location group is added to the RADIUS request. You can then use the location group name associated with the RADIUS client for SQL, LDAP, and check list attribute processing.

NOTE: The location group name is case-sensitive. If you create a location group called LOC1 and then assign a user a profile with a check list attribute called Loc1, the user is rejected.

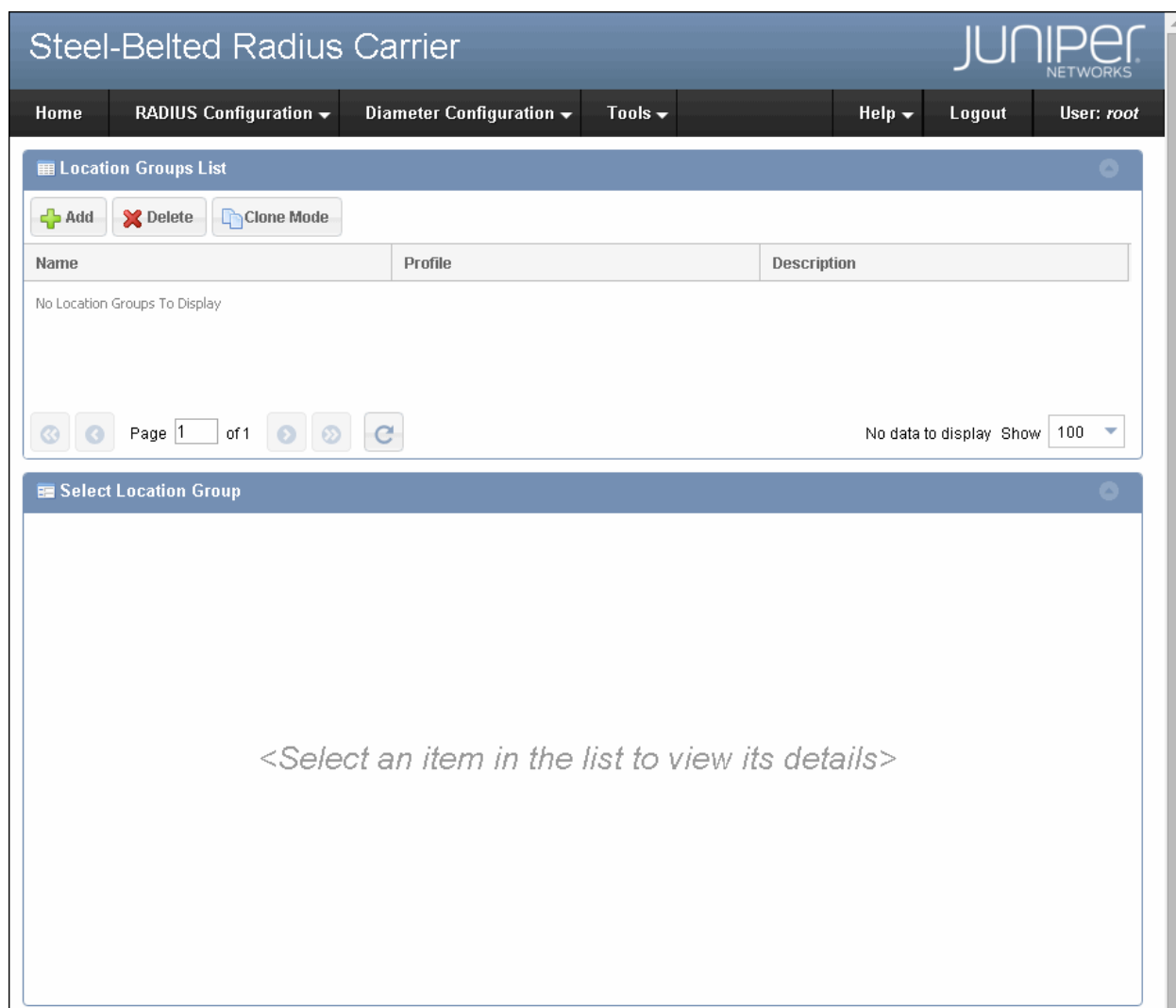
Adding a Location Group

To add a RADIUS location group using the Web GUI:

1. Select **RADIUS Configuration > Location Groups**.

The **Location Groups List** page ([Figure 25 on page 115](#)) appears.

Figure 25: Location Groups List Page



2. Click **Add**.

The **Create Location Group** pane ([Figure 26 on page 116](#)) appears with the **Basic Configuration** tab selected.

Figure 26: Create Location Group Pane—Basic Configuration

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and a user profile for 'root'. Below the navigation bar is a 'Location Groups List' section. The main area is titled 'Create Location Group' and contains two tabs: 'Basic Configuration' (selected) and 'RADIUS Clients'. The 'Basic Configuration' tab has the following fields and options:

- Name:** A text input field.
- Description:** A text input field.
- Profiles:** A section containing:
 - ☐ Use Profile: A checkbox followed by a dropdown menu.
 - Attribute Combination:** A section with two radio buttons: ☒ Merge and ☐ Override.
 - Merge Precedence:** A section with two radio buttons: ☒ User and ☐ RADIUS Client.

At the bottom of the pane are three buttons: Save, Clear, and Cancel.

3. Enter the name of the RADIUS location group in the **Name** field.
4. Optionally, enter a description of the RADIUS location group in the **Description** field. The description is not used during processing.
5. To associate a profile with the RADIUS location group, select the **Use Profile** check box and select an existing profile from the drop-down list. (To create a new profile, refer to [“Administering Profiles” on page 144](#)).
6. Specify how you want the profile to interact with the user settings:
 - If you want attributes in the specified profile to override identically named attributes configured for the user, select the **Override** option button.
 - If you want attributes in the profile to be merged with identically named attributes configured for the user, select the **Merge** option button and then select either **User** or **RADIUS Client** option button to take precedence if the attributes in the profile specify different values for the same single-value or ordered-multiple-value attribute.
7. Click the **RADIUS Clients** tab ([Figure 27 on page 117](#)) and select one or more client entries to identify the RADIUS clients that belong to the location group.

You can remove a RADIUS client from the location group by clearing the appropriate entry.

Figure 27: Create Location Group Pane—RADIUS Clients

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and a User: root session. Below the navigation bar, there are two tabs: 'Location Groups List' and 'Create Location Group'. The 'Create Location Group' tab is active, and within it, the 'RADIUS Clients' sub-tab is selected. This sub-tab contains a table with columns: Name, Description, IP Address, Make Or Model, and Address Pool. The table lists two clients: 'CLIENT 2' with IP 10.212.98.6 and 'V1 CLIENT' with IP 10.212.10.6. Below the table are 'Save', 'Clear', and 'Cancel' buttons.

<input type="checkbox"/>	Name	Description	IP Address	Make Or Model	Address Pool
<input type="checkbox"/>	CLIENT 2	RADIUS client 2	10.212.98.6	- Standard Radius -	
<input type="checkbox"/>	V1 CLIENT	RADIUS client 1	10.212.10.6	- Standard Radius -	

Save Clear Cancel

- Click **Save** to save the location group configuration.

The **Location Groups List** page (Figure 25 on page 115) displays an updated list of location group entries.

Editing a Location Group

To edit a RADIUS location group using the Web GUI:

- Select **RADIUS Configuration > Location Groups**.

The **Location Groups List** page (Figure 25 on page 115) appears.

- Select the RADIUS location group entry that you want to edit.

The **Selected Location Group** pane (Figure 28 on page 118) displays the settings configured for the location group.

Figure 28: Selected Location Group Pane

The screenshot displays the Juniper Networks Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user 'root'. The main content area is divided into two sections. The top section, 'Location Groups List', shows a table with one entry: 'LOCATION GROUP 1' with profile 'PROFILE1' and description 'Example'. Below the table are pagination controls showing 'Page 1 of 1' and 'Displaying 1 - 1 of 1'. The bottom section, 'Selected Location Group: LOCATION GROUP 1', contains configuration tabs for 'Basic Configuration' and 'RADIUS Clients'. The 'Basic Configuration' tab is active, showing fields for Name (LOCATION GROUP 1) and Description (Example). Under the 'Profiles' section, there is a checked 'Use Profile' checkbox with a dropdown menu set to 'PROFILE1'. Below this, the 'Attribute Combination' section has radio buttons for 'Merge' (selected) and 'Override'. The 'Merge Precedence' section has radio buttons for 'User' (selected) and 'RADIUS Client'. At the bottom of the configuration pane are 'Save', 'Reset', and 'Cancel' buttons.

3. Edit the settings for the location group entry as appropriate.

For information about the fields in the **Selected Location Group** pane, see [“Adding a Location Group” on page 115](#).

NOTE: You cannot edit the name of the location group.

4. Click **Save** to save the changes.

The **Location Groups List** page ([Figure 25 on page 115](#)) displays an updated list of location group entries.

Deleting a Location Group

To delete a RADIUS location group using the Web GUI:

1. Select **RADIUS Configuration > Location Groups**.

The **Location Groups List** page ([Figure 25 on page 115](#)) appears.

2. Select the RADIUS location group entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to confirm the delete request.

The **Location Groups List** page ([Figure 25 on page 115](#)) displays an updated list of location group entries.

Administering Users

IN THIS CHAPTER

- Users Overview | 120
- User Files | 121
- Setting Up Native Users | 122
- Setting Up UNIX Users or Groups | 135

This chapter describes how to add users to the SBR Carrier database. This chapter contains these topics:

Users Overview

Each user entry in the SBR Carrier database identifies one method by which the server can authenticate a specific user. Native user entries require you to enter the username and password into the SBR Carrier database. For all other types of user entry, the server relies on another database to confirm the user password.

You can add a UNIX user entry to provide authentication for a specific user defined on the Solaris or Linux server. For more flexibility, you can add a UNIX group to provide authentication for all users that belong to a specific group defined on the server.

NOTE: You can populate the user database for SBR Carrier by entering information using the Web GUI or by importing data from other servers. For more information about importing user information, see [“Importing and Exporting Data” on page 981](#).

Allowed Access Hours

The user’s allowed access hours can be specified by adding the **Funk-Allowed-Access-Hours** attribute to the user’s check list.

Funk-Allowed-Access-Hours is a variable-length string that identifies time periods in a 7-day week of 24-hour days. This string consists of one or more day specifiers (each of which can list one or more days or ranges of days) followed by one or more ranges of 24-hour times, in minutes.

The following is a sample **Funk-Allowed-Access-Hours** attribute value:

```
Funk-Allowed-Access-Hours M-W 0100-1400 2300-2400 M, Tu, Th, F 0530-1500, Sa-Su 0000-2400
```

The syntax rules for **Funk-Allowed-Access-Hours** are as follows:

- Time ranges can be inclusive (1000-1100 allows access only between 10 a.m. and 11 a.m.) or exclusive (1100-1000 allows access any time except between 10 a.m. and 11 a.m.).
- Day specifiers, and ranges of days and times can be separated by commas or spaces; ranges of days or times are indicated by hyphens (m-w or 0239-1459).
- Days can be specified by the minimum number of case-insensitive letters necessary to distinguish them (Su, M, Tu, W, Th, F, Sa) and can wrap around the end of the week (Sa-Su).
- At least one time period is required for each day; that is, each day, list, or range of days must be followed by one or more ranges of times.
- Times are specified using four digits, with leading zeroes where needed (0001 for 12:01 a.m., 0630 for 6:30 a.m., and so forth).

When assigned to a user's check list, the **Funk-Allowed-Access-Hours** value allows the user access during the following time periods:

- 1 a.m. to 2 p.m. and 11:00 p.m. to midnight, Monday through Wednesday
- 5:30 a.m. to 3:00 p.m. Monday, Tuesday, Thursday, and Friday
- Any time Saturday or Sunday

The total access times Monday are 1 a.m. to 3:00 p.m. and 11:00 p.m. to midnight.

If the user attempts access on Sunday at 11:30 p.m., access would be allowed and a Session-Timeout attribute specifying a value of 1800 seconds (30 minutes, until midnight Sunday, when the access period ends) would be returned. However, if the user's return list includes a Session-Timeout with a value less than 1800, this lesser value would be returned.

User Files

[Table 25 on page 122](#) describes the configuration files you use to establish settings for users. For more information about these files, refer to the *SBR Carrier Reference Guide*.

Table 25: User Account Files

File Name	Function
lockout.ini	Configures settings for when you want SBR Carrier to lock user accounts after repeated failed login attempts.
redirect.ini	Configures settings for when you want SBR Carrier to redirect users after repeated failed login attempts.
radius.ini	Specifies (among other things) the settings relating to a third party authentication support in SBR Carrier.

Setting Up Native Users

This section describes how to add, edit, or delete a native user.

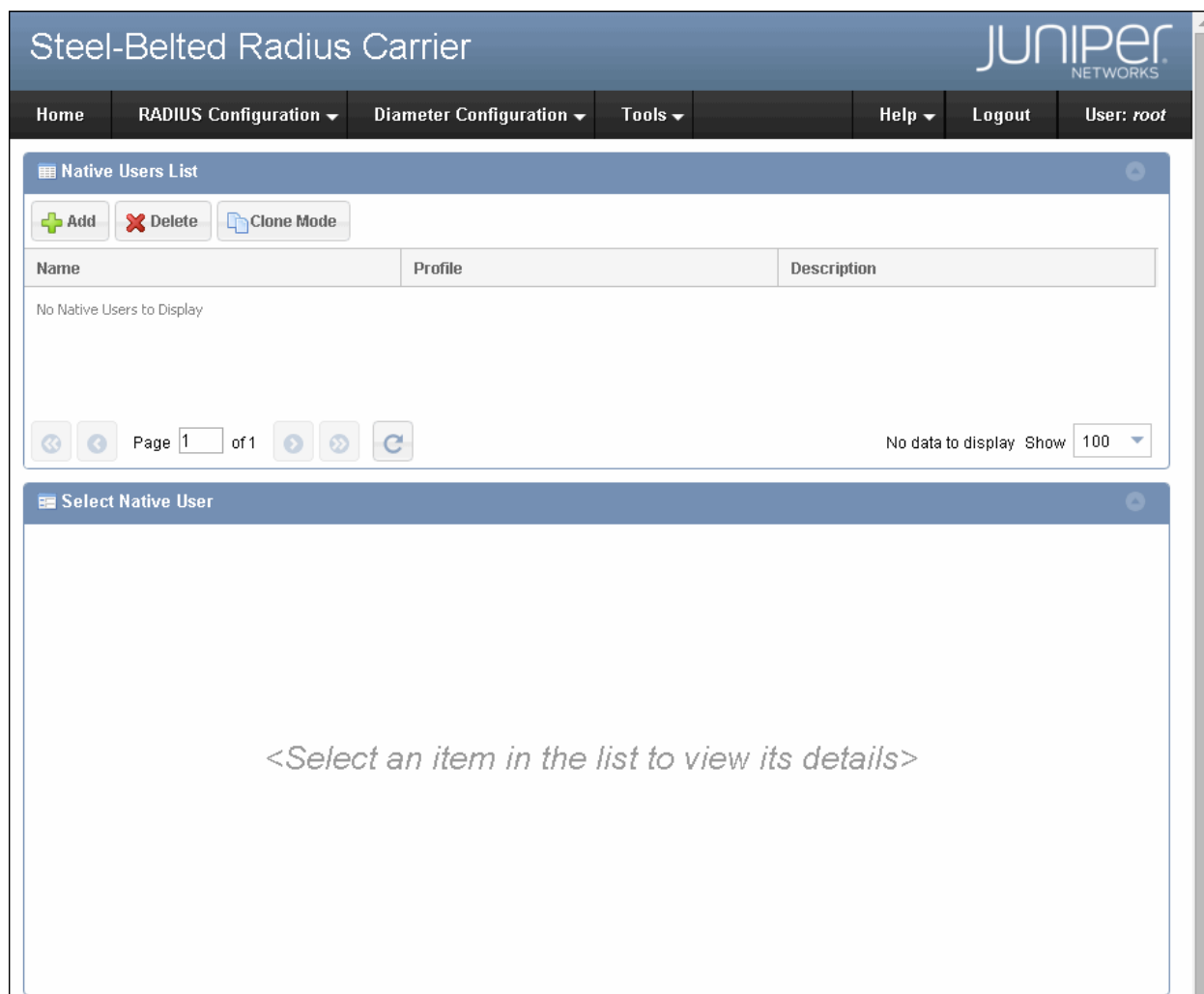
Adding a Native User

To add a native user to the SBR Carrier database using the Web GUI:

1. Select **RADIUS Configuration > Users > Native Users**.

The **Native Users List** page (Figure 29 on page 123) appears.

Figure 29: Native Users List Page



2. Click **Add**.

The **Create Native User** pane (Figure 30 on page 124) appears with the **Basic Configuration** tab selected.

Figure 30: Create Native User Pane—Basic Configuration

The screenshot shows the 'Steel-Belted Radius Carrier' web interface. The top navigation bar includes 'Home', 'RADIUS Configuration', 'Diameter Configuration', 'Tools', 'Help', 'Logout', and 'User: root'. Below this is a 'Native Users List' button. The main content area is titled 'Create Native User' and has two tabs: 'Basic Configuration' (selected) and 'Attributes'. The 'Basic Configuration' tab contains the following fields and controls:

- Name:** A text input field.
- Password:** A text input field with a 'Show' button to its right.
- Store Hash of Password:** A checkbox.
- Description:** A text input field.
- Profile:** A dropdown menu with the text 'Select a User Profile..'.
- Concurrency:** A checkbox.
- Max Connections:** A numeric input field with the value '1' and up/down arrows.

At the bottom of the pane are three buttons: 'Save', 'Clear', and 'Cancel'.

3. Enter a login name for the native user in the **Name** field.

Native user entries in the SBR Carrier database have all uppercase names. No matter how the native username is typed when the account is created, it is converted to all uppercase letters. For example, a native username entered as **realLife1** is stored as **REALLIFE1** in the SBR Carrier database.

NOTE: The entered native username is not converted to uppercase letters during authentication request processing.

4. Enter a login password in the **Password** field.

Passwords are case-sensitive. If you want the characters in the password to display as you type, click **Show**. After viewing the characters, you can click **Hide** to hide the characters.

5. Specify whether you want the login password to be encrypted before it is stored.

- If the native user requires PAP authentication and you want to store the hash of the password in the SBR Carrier database, select the **Store Hash of Password** check box. This option allows the native user to authenticate using only PAP.
- If the native user requires CHAP authentication, clear the **Store Hash of Password** check box.

6. Optionally, enter a description for the native user in the **Description** field.

The description you associate with a native user is not used during processing.

7. If you want to use a profile to assign check list and return list attributes to the native user, use the **Profile** list to select the profile.

For more information about profiles, see [“Administering Profiles” on page 144](#).

NOTE: Attributes inherited from a profile are overridden by attributes assigned to a specific user.

8. If you want to specify the maximum number of concurrent connections the native user can maintain, select the **Concurrency** check box and enter a number in the **Max Connections** field.

When the user requests access, the user can be authenticated using the given authentication method only if fewer than the number of connections are currently open for the user.

9. To add check list or return list attributes for the native user, click the **Attributes** tab ([Figure 31 on page 125](#)).

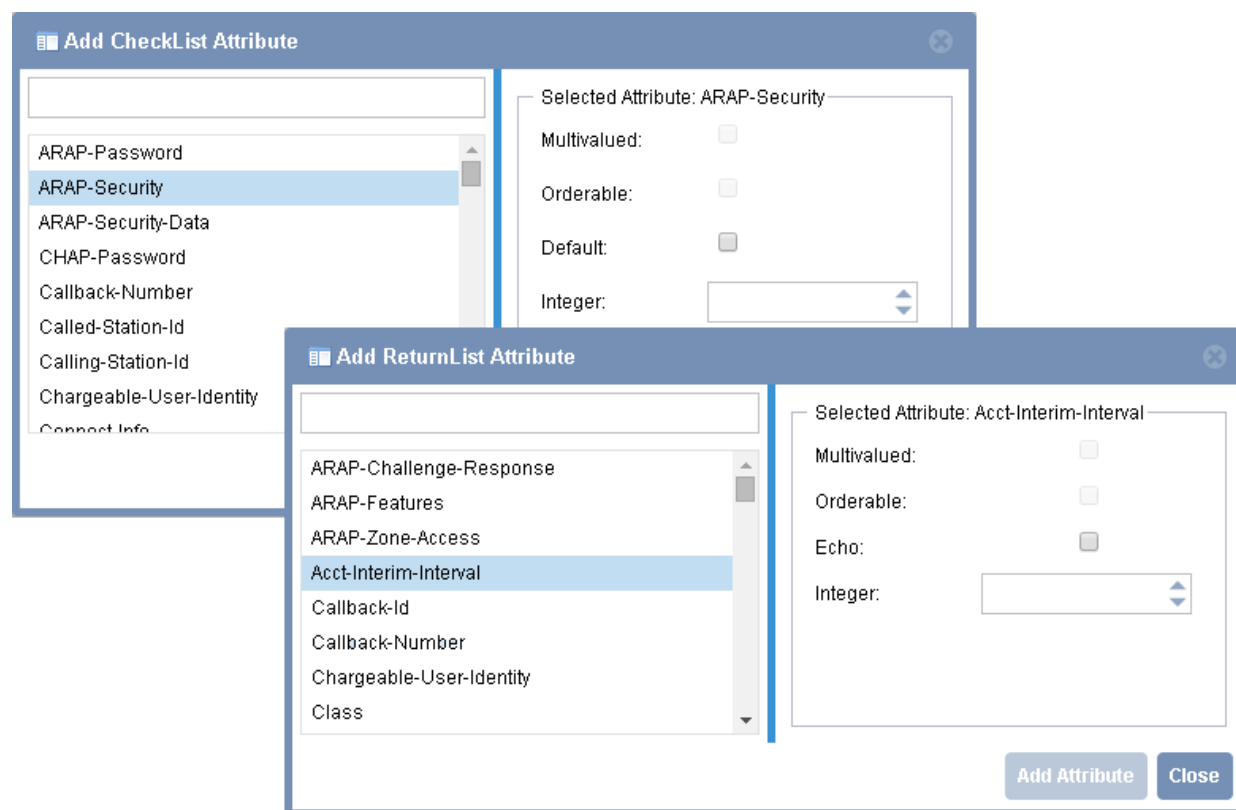
Figure 31: Create Native User Pane—Attributes

The screenshot shows the 'Steel-Belted Radius Carrier' web interface. The top navigation bar includes 'Home', 'RADIUS Configuration', 'Diameter Configuration', 'Tools', 'Help', 'Logout', and 'User: root'. The main content area is titled 'Native Users List' and 'Create Native User'. The 'Attributes' tab is selected, showing two sections: 'CheckList' and 'ReturnList'. Each section has a table with columns for attributes and values. The 'CheckList' table has columns 'Attribute', 'Value', and 'Default'. The 'ReturnList' table has columns 'Attribute', 'Value', and 'Echo'. Below the 'ReturnList' table, there is a section for 'When Both Framed-IP-Address and Framed-IPv6-Prefix Attributes Configured' with radio buttons for 'Return Only Framed-IPv6-Prefix Attribute' and 'Return Both Attributes'. At the bottom of the pane are buttons for 'Save', 'Clear', and 'Cancel'.

10. Click **Add** under the **CheckList** area or the **ReturnList** area.

The **Add CheckList Attribute** or **Add ReturnList Attribute** dialog box [Figure 32 on page 126](#) appears.

Figure 32: Add CheckList Attribute and Add ReturnList Attribute Dialogs



11. Select the attribute you want to add to the check list or return list from the attributes list.

You can search the attributes by entering the attribute name in the text box.

12. Select or enter a value for the selected attribute.

The dialog changes according to the attribute you choose. Some attributes require that you enter a value, string, or IP address. Other attributes require that you choose from a fixed list of values.

The **Multivalued** check box always appears dimmed, so you cannot select or clear this check box. If the **Multivalued** check box appears cleared, an attribute can have only one value. If the **Multivalued** check box appears selected, you can add multiple values for the attribute.

The **Orderable** check box always appears dimmed, so you cannot select or clear this check box. If the **Orderable** check box appears selected, you can define the order of the multi-valued attributes. If the **Orderable** check box appears cleared, the attribute is neither a multi-valued attribute nor an orderable attribute.

(Check list attributes only) To set the attributes value to the default value (which is useful in situations where the attribute is not included in the RADIUS request), select the **Default** check box.

(Return list single-valued attributes only) If you do not want to specify a particular value, but want to make sure that whatever value of the attribute appears in the RADIUS request is echoed to the client in the RADIUS response, select the **Echo** check box.

NOTE: The echo property is disabled for multi-valued return list attributes. The echo property is also disabled for the Framed-IPv6-Address attribute regardless of its multi-value setting.

NOTE: You cannot define multiple instances of Framed-IPv6-Address attributes in a return list or check list. The Framed-IPv6-Address attribute can appear only once in a return list or check list.

13. Click **Add Attribute** to add this AVP to the list.

14. Repeat steps 11 through 13 to add more return list or check list attributes for the user.

15. When you are finished adding AVPs, click **Close**.

The **CheckList** area or the **ReturnList** area in the **Create Native User** pane (Figure 31 on page 125) displays the updated list of selected attributes.

You can modify the return or check list by using the **Edit** and **Delete** buttons. You can reorder the attributes by selecting each attribute and using the **Up** or **Down** arrow.

NOTE: The **Up** arrow is disabled, if the selected attribute is not orderable or if the selected attribute is already the first value. The **Down** arrow is disabled, if the selected attribute is not orderable or if the selected attribute is already the last value.

16. Optionally, if you have added a structured or parent attribute to the native user, add its subattributes to the structured or parent attribute. For more information about how to add subattributes to a structured or parent attribute, see [“Adding Subattributes to a Structured Attribute” on page 128](#).

17. If you have added both Framed-IP-Address and Framed-IPv6-Prefix attributes to the return list, specify whether to return only the Framed-IPv6-Prefix attribute or both Framed-IP-Address and Framed-IPv6-Prefix attributes in the RADIUS response. In the **When Both Framed-IP-Address and Framed-IPv6-Prefix Attributes Configured** area, you can:

- Select the **Return Only Framed-IPv6-Prefix Attribute** option to return only the Framed-IPv6-Prefix attribute in the RADIUS response.

- Select the **Return Both Attributes** option to return both Framed-IP-Address and Framed-IPv6-Prefix attributes in the RADIUS response.

NOTE: The **When Both Framed-IP-Address and Framed-IPv6-Prefix Attributes Configured** area is enabled only if you have added both Framed-IP-Address and Framed-IPv6-Prefix attributes to the return list.

18. Click **Save** to save the native user configuration.

The **Native Users List** page ([Figure 29 on page 123](#)) displays an updated list of native user entries.

Adding Subattributes to a Structured Attribute

The following terminologies are used in this section:

- **Attribute**—used to represent a standard RADIUS attribute in the packet.
- **Parent or structured Attribute**—used to describe an attribute that contains subattributes, rather than a conventional simple data type such as an integer. This may be a parent attribute, or it may itself be a subattribute.
- **Subattribute**—refers to the data items within a structured or parent attribute. While the subattributes are frequently in TLV format, occasionally they are missing Type, Length, or both.

NOTE: Structured attributes (VSAs with subattributes) defined in return lists are added to the reply message as a whole unit, rather than their subattributes being added individually to any existing response VSAs. In this way they are treated just as unstructured VSAs.

For example:

- Attribute "ParentAttr" is defined as being a multivalue return list attribute, with possible subattributes "ChildAttrA" and "ChildAttrB".
- A response already has a copy of "ParentAttr" with subattribute "ChildAttrA", for example from an authentication process.
- A profile specifies that "ParentAttr" must be added with subattribute "ChildAttrB".

The result is a response with two ParentAttr structured attributes:

ParentAttr

ChildAttrA

ParentAttr

ChildAttrB

The result is *not* a response with a single ParentAttr:

ParentAttr

ChildAttrA

ChildAttrB

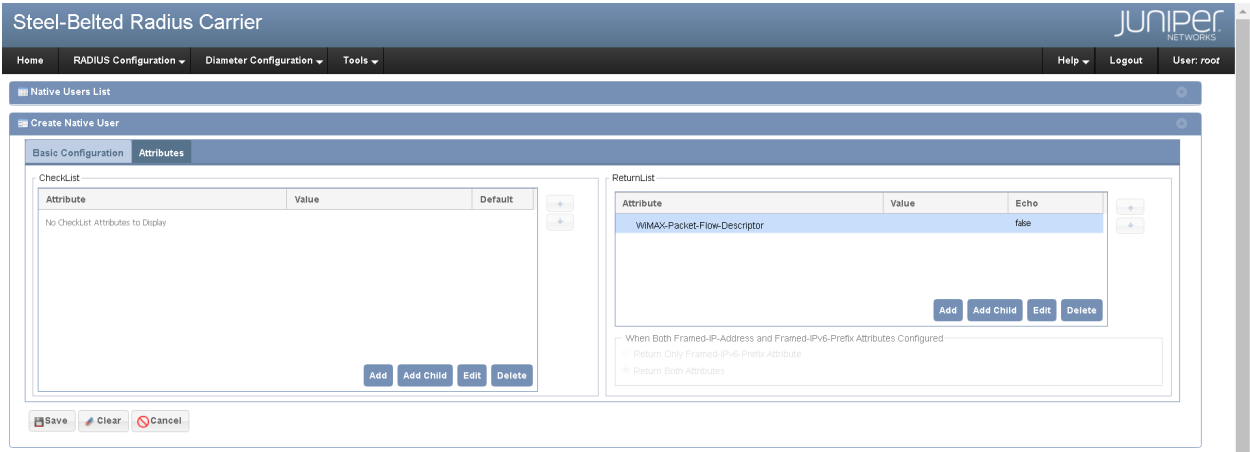
To add subattributes to a structured attribute using the Web GUI:

1. Select the parent attribute to which you want to add subattributes.

In the example shown in [Figure 33 on page 130](#), **WiMAX-Packet-Flow-Descriptor** is the structured attribute to which you add subattributes.

NOTE: If you enable the **Echo** check box at the parent attribute level, you cannot add subattributes.

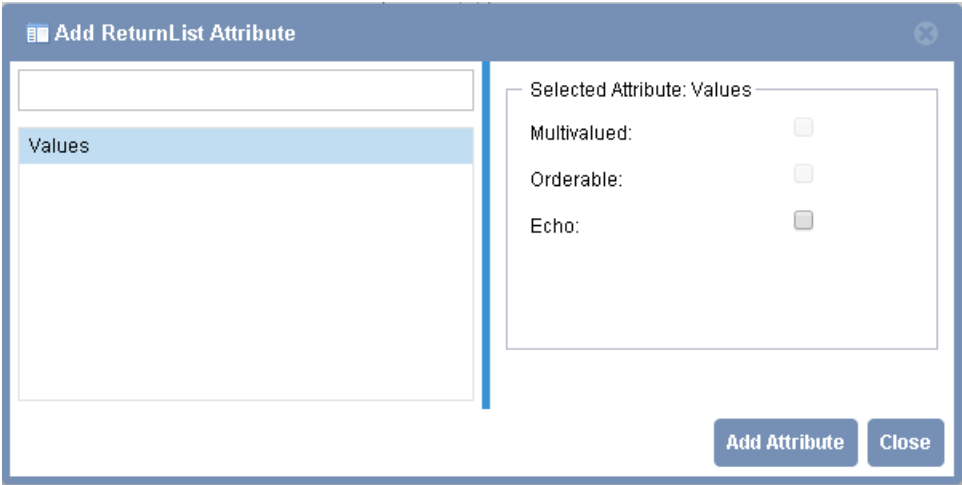
Figure 33: Adding Subattributes



2. Click **Add Child**.

The **Add CheckList Attribute** or **Add ReturnList Attribute** dialog box appears. As an example, [Figure 34 on page 131](#) shows the **Add ReturnList Attribute** dialog box.

Figure 34: Add ReturnList Attribute Dialog



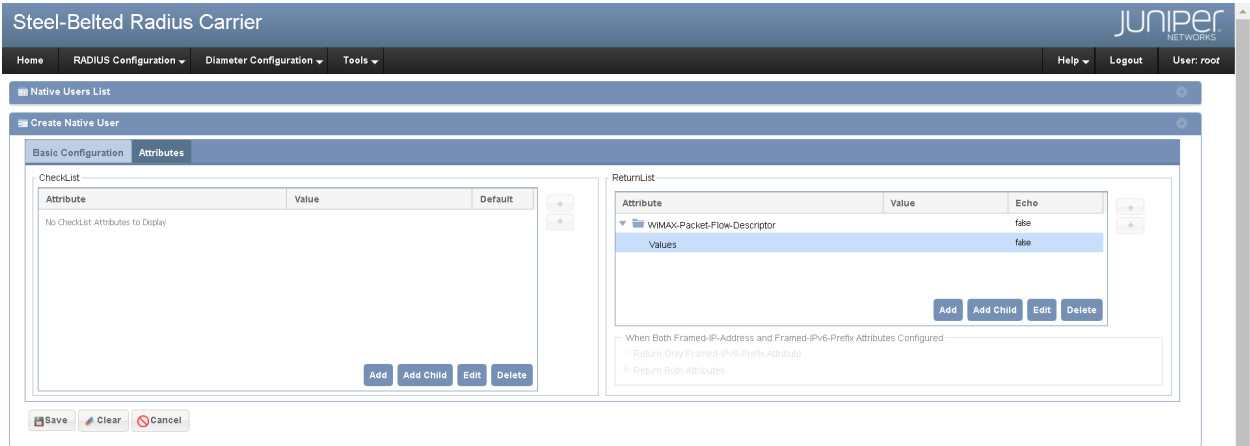
3. Select **Values** and click **Add Attribute**.

NOTE: In most cases, the attributes list displays the subattributes appropriate for the selected parent attribute, and you simply select the desired subattribute and click **Add Attribute**. However, because WiMAX structured (parent) attributes are quite long, a continuation flag attribute is required. This is denoted by the **Values** attribute.

4. Click **Close**.

Values now appears under the parent attribute, indicating that you can add a subattribute ([Figure 35 on page 131](#)).

Figure 35: Parent Attribute



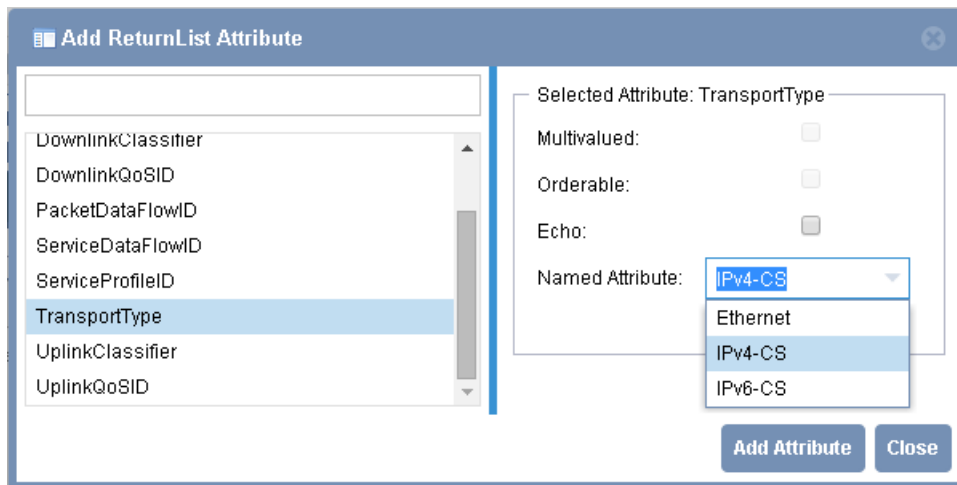
5. Select **Values** from the list and click **Add Child**.

6. Select or enter a value for the subattribute.

The dialog changes according to the subattribute. Some subattributes require that you enter a value, string, or IP address. Other subattributes require that you choose from a fixed list of values.

[Figure 36 on page 132](#) shows an example in which **TransportType** is the subattribute and **IPv4-CS** has been selected as the value.

Figure 36: Example Subattribute



7. Click **Add Attribute** to add this subattribute to the list.

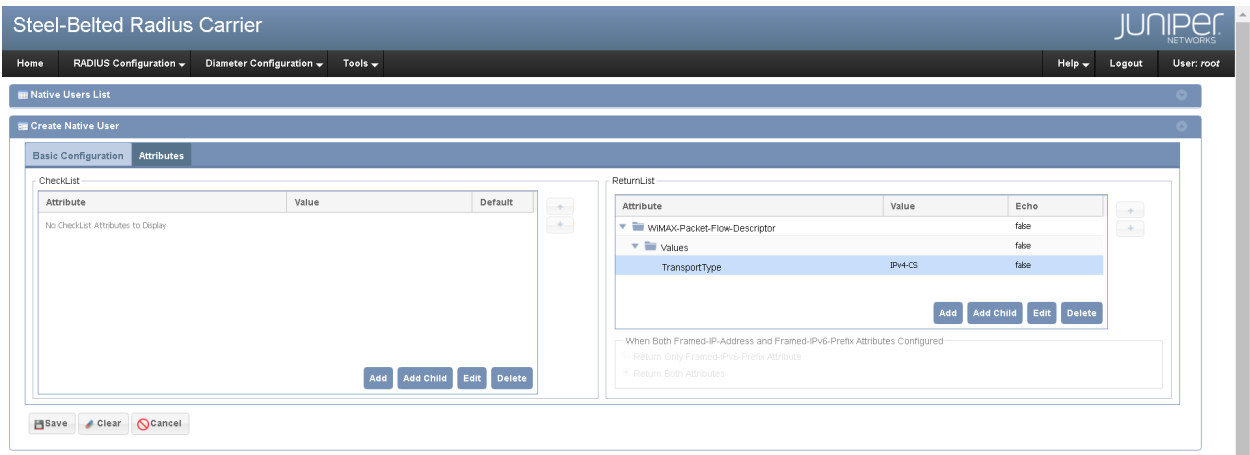
8. Repeat steps 6 and 7 to add more subattributes to the parent attribute.

9. When you are finished adding subattributes, click **Close**.

The **CheckList** area or the **ReturnList** area ([Figure 37 on page 133](#)) displays the updated list of subattributes.

You can modify the subattributes by using the **Edit** and **Delete** buttons.

Figure 37: Structured Attributes Added



Editing a Native User

To edit a native user entry in the SBR Carrier database using the Web GUI:

1. Select **RADIUS Configuration > Users > Native Users**.

The **Native Users List** page (Figure 29 on page 123) appears.

2. Select the native user entry that you want to edit.

The **Selected Native User** pane (Figure 38 on page 134) displays the settings configured for the native user entry.

Figure 38: Selected Native User Pane

The screenshot displays the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user 'root'. The main content area is divided into two panes. The top pane, 'Native Users List', shows a table with two entries: 'NATIVE1' and 'USER'. The 'NATIVE1' entry is selected. Below the table are pagination controls showing 'Page 1 of 1' and 'Displaying 1 - 2 of 2'. The bottom pane, 'Selected Native User: NATIVE1', contains a 'Basic Configuration' tab. This tab includes fields for Name (NATIVE1), Password (masked with dots), Description, Profile (PROFILE1), and Concurrency (unchecked). There are 'Show' and 'Validate' buttons next to the password field, and a 'Store Hash of Password' checkbox. At the bottom of the pane are 'Save', 'Reset', and 'Cancel' buttons.

Name	Profile	Description
NATIVE1	PROFILE1	
USER		

Page 1 of 1 Displaying 1 - 2 of 2 Show 100

Selected Native User: NATIVE1

Basic Configuration Attributes

Name: NATIVE1

Password: Show Validate

☐ Store Hash of Password

Description:

Profile: PROFILE1

Concurrency: ☐ Max Connections:

Save Reset Cancel

3. Edit the settings for the native user entry as appropriate.

For information about the fields in the **Selected User** pane, see [“Adding a Native User” on page 123](#).

NOTE: You cannot edit the name of the native user.

4. Click **Save** to save the changes.

The **Native Users List** page (Figure 29 on page 123) displays an updated list of native user entries.

Deleting a Native User

To delete a native user using the Web GUI:

1. Select **RADIUS Configuration > Users > Native Users**.

The **Native Users List** page ([Figure 29 on page 123](#)) appears.

2. Select the native user entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to confirm the delete request.

The **Native Users List** page ([Figure 29 on page 123](#)) displays an updated list of native user entries.

Setting Up UNIX Users or Groups

This section describes how to add, edit, or delete a UNIX user or group.

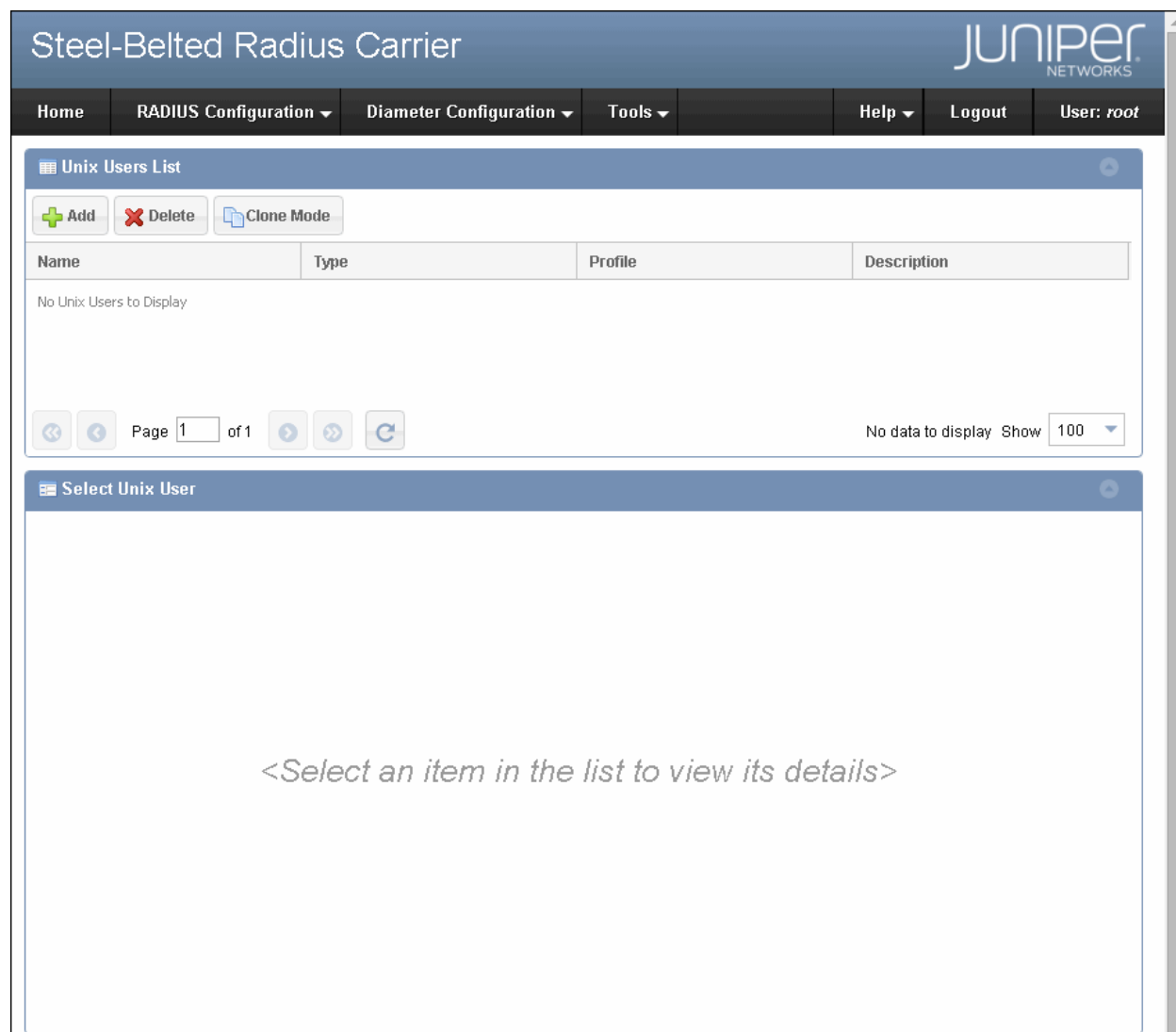
Adding a UNIX User or Group

To add a UNIX user or group using the Web GUI:

1. Select **RADIUS Configuration > Users > Unix Users**.

The **Unix Users List** page ([Figure 39 on page 136](#)) appears.

Figure 39: Unix Users List Page



2. Click **Add**.

The **Create Unix User** pane ([Figure 40 on page 137](#)) appears with the **Basic Configuration** tab selected.

Figure 40: Create Unix User Pane—Basic Configuration

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user (root). Below the navigation bar, there are two tabs: 'Unix Users List' and 'Create Unix User'. The 'Create Unix User' tab is active, and within it, the 'Basic Configuration' sub-tab is selected. The form contains the following fields and controls:

- Name:** A text input field.
- Description:** A text input field.
- Use Profile:** A checkbox and a dropdown menu.
- Concurrency:** A checkbox.
- Max Connections:** A numeric input field with up/down arrows.
- User/Group Selection:** Two radio buttons labeled 'User' and 'Group'. The 'User' radio button is selected. Next to the 'User' radio button is a dropdown menu labeled 'Select an User'. Next to the 'Group' radio button is a dropdown menu labeled 'Select a Group'.

At the bottom of the form, there are three buttons: 'Save', 'Clear', and 'Cancel'.

3. Search for the users or groups that you want to add in the **Name** field.
 - If you want to add a user, select the **User** option button and select the name of a user from the drop-down list.
 - If you want to add a group, select the **Group** option button and select the name of a user group from the drop-down list.
4. Optionally, enter a description for the UNIX user or group in the **Description** field.
5. If you want to use a profile to assign check list and return list attributes to the user, select the **Use Profile** check box and use the drop-down list to select the profile.

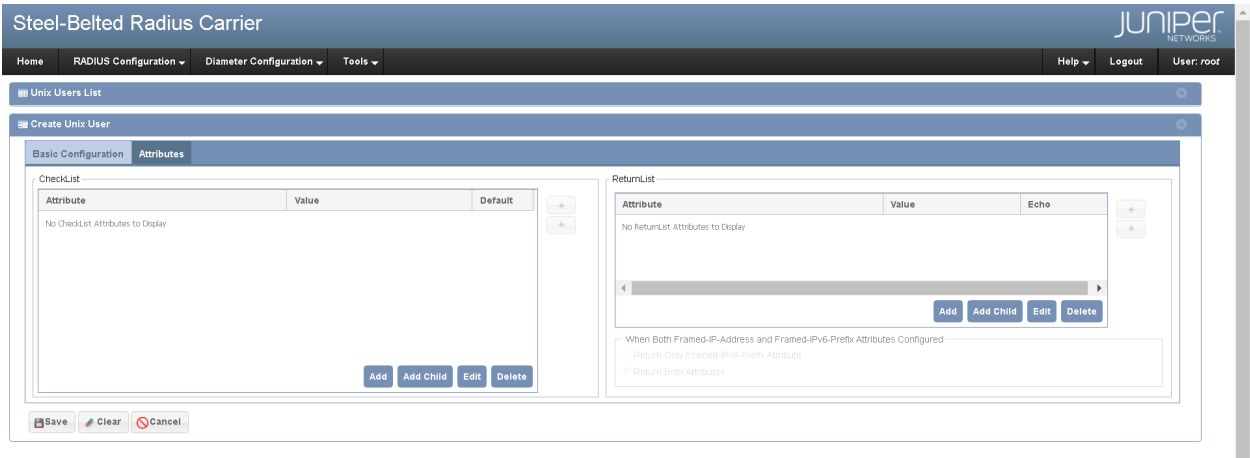
NOTE: Attributes inherited from a profile are overridden by attributes assigned to a specific user.

6. If you want to specify the maximum number of concurrent connections the UNIX user can maintain, select the **Concurrency** check box and enter a number in the **Max Connections** field.

When the user requests access, the user can be authenticated using the given authentication method only if fewer than the number of connections are currently open for the user.

7. To add check list or return list attributes for the UNIX user or group, click the **Attributes** tab (Figure 41 on page 138).

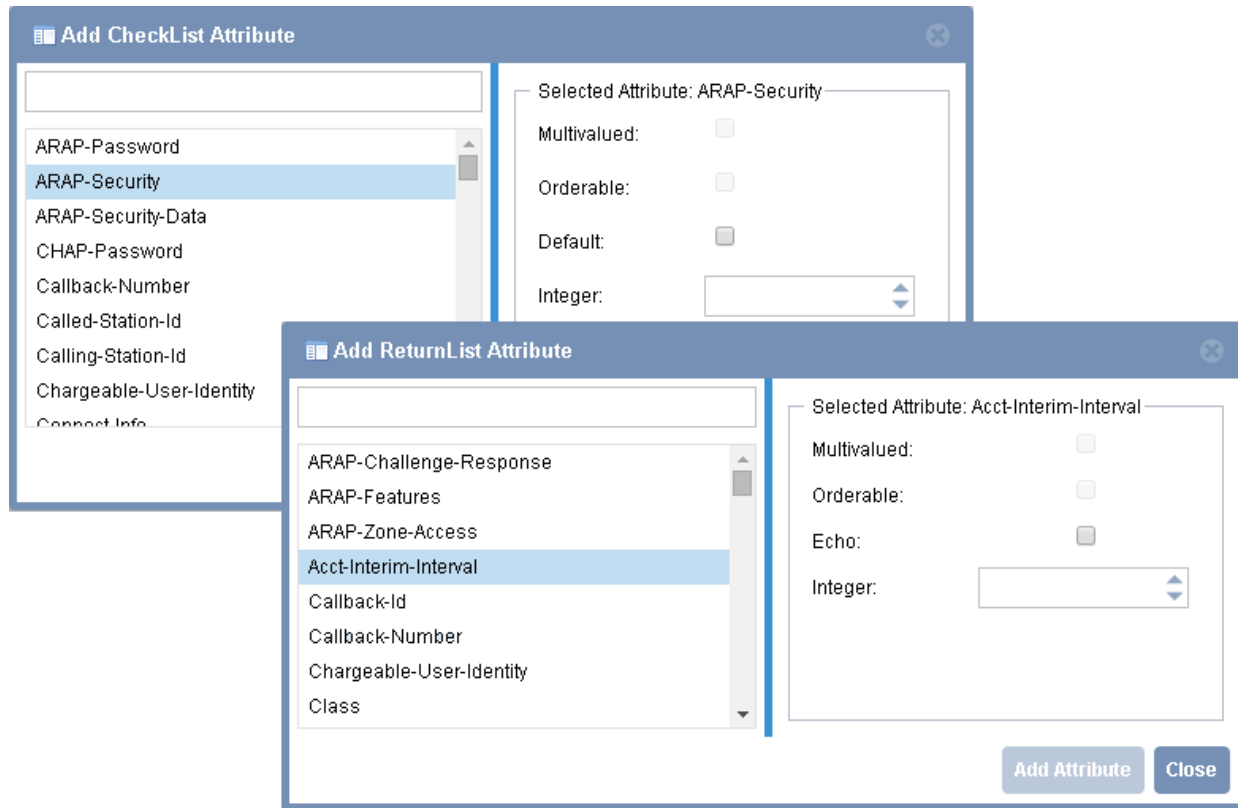
Figure 41: Create Unix User Pane—Attributes



8. Click **Add** under the **CheckList** area or the **ReturnList** area.

The **Add CheckList Attribute** or **Add ReturnList Attribute** dialog box (Figure 42 on page 139) appears.

Figure 42: Add CheckList Attribute and Add ReturnList Attribute Dialogs



9. Select the attribute you want to add to the check list or return list from the attributes list.

You can search the attributes by entering the attribute name in the text box.

10. Select or enter a value for the selected attribute.

The dialog changes according to the attribute you choose. Some attributes require that you enter a value, string, or IP address. Other attributes require that you choose from a fixed list of values.

The **Multivalued** check box always appears dimmed, so you cannot select or clear this check box. If the **Multivalued** check box appears cleared, an attribute can have only one value. If the **Multivalued** check box appears selected, you can add multiple values for the attribute.

The **Orderable** check box always appears dimmed, so you cannot select or clear this check box. If the **Orderable** check box appears selected, you can define the order of the multi-valued attributes. If the **Orderable** check box appears cleared, the attribute is neither a multi-valued attribute nor an orderable attribute.

(Check list attributes only) To set the attributes value to the default value (which is useful in situations where the attribute is not included in the RADIUS request), select the **Default** check box.

(Return list single-valued attributes only) If you do not want to specify a particular value, but want to make sure that whatever value of the attribute appears in the RADIUS request is echoed to the client in the RADIUS response, select the **Echo** check box.

NOTE: The echo property is disabled for multi-valued return list attributes. The echo property is also disabled for the Framed-IPv6-Address attribute regardless of its multi-value setting.

NOTE: You cannot define multiple instances of Framed-IPv6-Address attributes in a return list or check list. The Framed-IPv6-Address attribute can appear only once in a return list or check list.

11. Click **Add Attribute** to add this AVP to the list.

12. Repeat steps 9 through 11 to add more return list or check list attributes for the user.

13. When you are finished adding AVPs, click **Close**.

The **CheckList** area or the **ReturnList** area in the **Create Unix User** pane (Figure 41 on page 138) displays the updated list of selected attributes.

You can modify the return or check list by using the **Edit** and **Delete** buttons. You can reorder the attributes by selecting each attribute and using the **Up** or **Down** arrow.

NOTE: The **Up** arrow is disabled, if the selected attribute is not orderable or if the selected attribute is already the first value. The **Down** arrow is disabled, if the selected attribute is not orderable or if the selected attribute is already the last value.

14. Optionally, if you have added a structured or parent attribute to the UNIX user or group, add its subattributes to the structured or parent attribute. For more information about how to add subattributes to a structured or parent attribute, see [“Adding Subattributes to a Structured Attribute” on page 128](#).

15. If you have added both Framed-IP-Address and Framed-IPv6-Prefix attributes to the return list, specify whether to return only the Framed-IPv6-Prefix attribute or both Framed-IP-Address and Framed-IPv6-Prefix attributes in the RADIUS response. In the **When Both Framed-IP-Address and Framed-IPv6-Prefix Attributes Configured** area, you can:

- Select the **Return Only Framed-IPv6-Prefix Attribute** option to return only the Framed-IPv6-Prefix attribute in the RADIUS response.

- Select the **Return Both Attributes** option to return both Framed-IP-Address and Framed-IPv6-Prefix attributes in the RADIUS response.

NOTE: The **When Both Framed-IP-Address and Framed-IPv6-Prefix Attributes Configured** area is enabled only if you have added both Framed-IP-Address and Framed-IPv6-Prefix attributes to the return list.

16. Click **Save** to save the UNIX user or group configuration.

The **Unix Users List** page ([Figure 39 on page 136](#)) displays an updated list of UNIX user or group entries.

Editing a UNIX User or Group

To edit a UNIX user or group entry using the Web GUI:

1. Select **RADIUS Configuration > Users > Unix Users**.

The **Unix Users List** page ([Figure 39 on page 136](#)) appears.

2. Select the UNIX user or group entry that you want to edit.

The **Selected Unix User** pane ([Figure 43 on page 142](#)) displays the settings configured for the UNIX user or group entry.

Figure 43: Selected Unix User Pane

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user (User: root). The main content area is divided into two panes. The top pane, titled 'Unix Users List', contains buttons for Add, Delete, and Clone Mode, and a table with one entry: 'UNIX user' of type 'user' with profile 'PROFILE11' and description 'Example configuration'. The bottom pane, titled 'Selected Unix User: UNIX user', has two tabs: 'Basic Configuration' and 'Attributes'. The 'Basic Configuration' tab is active, showing fields for Name (UNIX user), Description (Example configuration), Use Profile (checked, PROFILE11), and Concurrency (unchecked). At the bottom of this pane are Save, Reset, and Cancel buttons.

Name	Type	Profile	Description
UNIX user	user	PROFILE11	Example configuration

Page 1 of 1
Displaying 1 - 1 of 1 Show 100

Selected Unix User: UNIX user

Basic Configuration | Attributes

Name: UNIX user ☒ User ☐ Group

Description: Example configuration

☒ Use Profile PROFILE11

Concurrency: ☐ Max Connections:

Save Reset Cancel

3. Edit the settings for the UNIX user or group entry as appropriate.

For information about the fields in the **Selected User** pane, see [“Adding a UNIX User or Group” on page 136](#).

NOTE: You cannot edit the name of the UNIX user or group.

4. Click **Save** to save the changes.

The **Unix Users List** page (Figure 39 on page 136) displays an updated list of UNIX user or group entries.

Deleting a UNIX User or Group

To delete a UNIX user or group entry using the Web GUI:

1. Select **RADIUS Configuration > Users > Unix Users**.

The **Unix Users List** page ([Figure 39 on page 136](#)) appears.

2. Select the UNIX user or group entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to confirm the delete request.

The **Unix Users List** page ([Figure 39 on page 136](#)) displays an updated list of UNIX user or group entries.

Administering Profiles

IN THIS CHAPTER

- [About Profiles | 144](#)
- [Adding a Profile | 146](#)
- [Editing a Profile | 150](#)
- [Removing a Profile | 151](#)

This chapter describes how to set up and administer user profiles. This chapter contains these topics:

About Profiles

SBR Carrier enables you to define default templates of check list and return list pairs called *profiles*. A profile provides specific attributes for one or both lists. You can define as many profiles as you require. Profiles provide a powerful means of managing and configuring accounts.

When you edit a user account, you can assign a profile to the user so the attributes of the profile become the default settings for that user account. After you assign a profile to a user account, you can modify the new entries on the user's check list and return list. Changes you make apply only to the specific user entry; they do not affect the profile itself. Assigning a profile and then overriding individual attributes is a convenient way to leverage SBR Carrier features to your advantage.

To change attribute settings across many users immediately, edit the profile that you have assigned to these users. The changes you make to a profile are automatically reflected in each user's check list and return list.

Adding a Check List or Return List Attribute to a Profile

A *check list attribute* is an item of information that must accompany a request for connection before the connection can be authenticated. A *return list attribute* is an item of information that SBR Carrier includes in the Access-Accept message when a connection request is approved.

Resolving Profile and User Attributes

If user-specific attributes are stored in an external database, SBR Carrier determines the final set of attributes for a user by merging the attributes stored in the native database with those retrieved from the external database. This calculation is performed as follows:

1. The attributes from the profile (or alias user) assigned to the user are first retrieved.
2. These attributes are then merged with the user-specific modifications to the attributes in the following manner:
 - If the attribute is multi-valued, then the attribute(s) retrieved from the external database is added to the overall list of attributes.
 - If the attribute is single-valued, then the attribute(s) retrieved from the external database replaces any attribute of the same name in the profile or associated with the alias.
 - If the attribute is orderable, then the attribute(s) retrieved from the external database replaces any orderable attribute of the same name in the profile or associated with the alias.

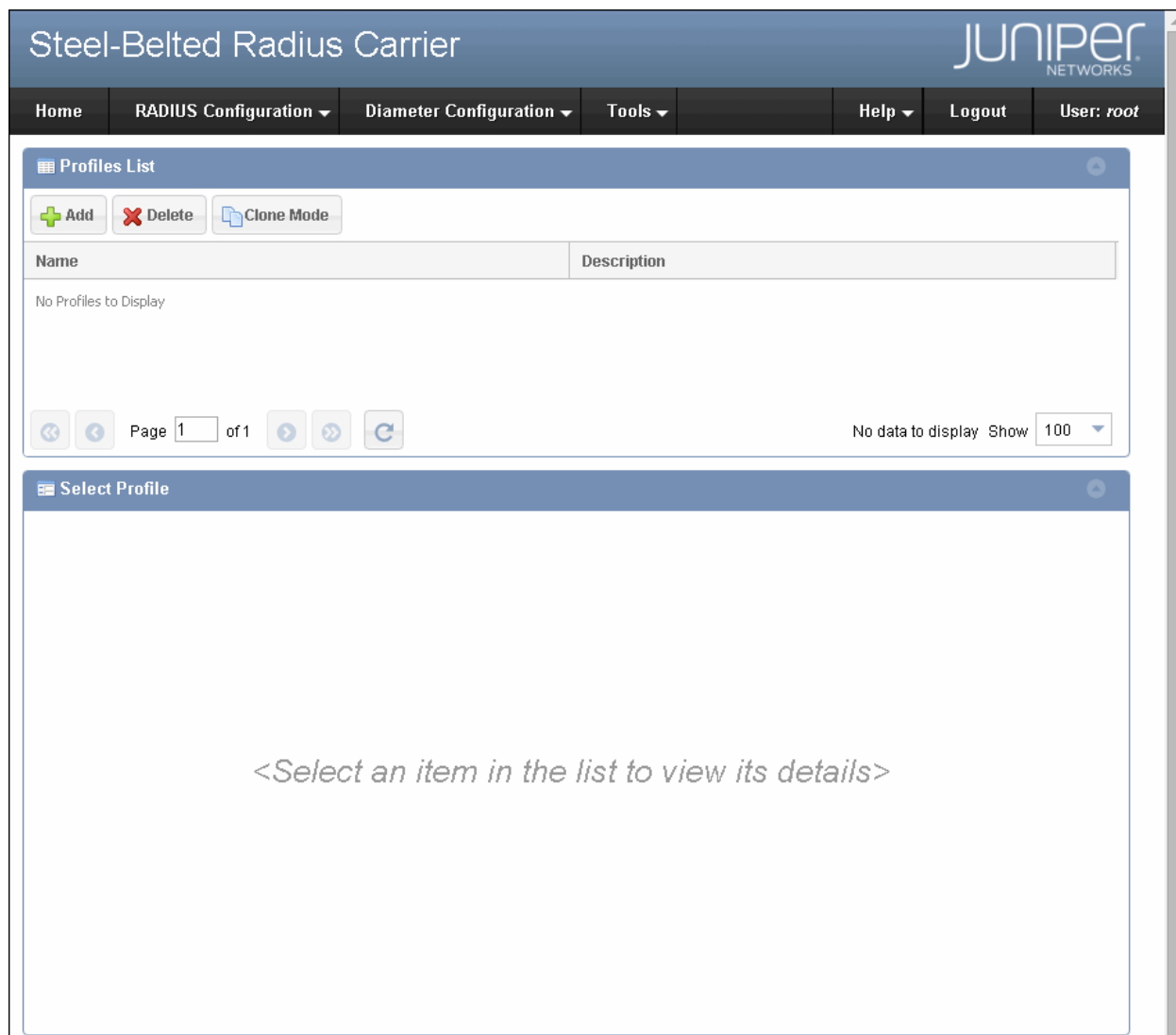
Adding a Profile

To add a profile using the Web GUI:

1. Select **RADIUS Configuration > Profiles**.

The **Profiles List** page ([Figure 44 on page 146](#)) appears.

Figure 44: Profiles List Page



2. Click **Add**.

The **Create Profile** pane ([Figure 45 on page 147](#)) appears.

Figure 45: Create Profile Pane

Steel-Belted Radius Carrier

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

Profiles List

Create Profile

Name:

Description:

Attributes

CheckList

Attribute	Value	Default
No CheckList Attributes to Display		

Add Add Child Edit Delete

ReturnList

Attribute	Value	Echo
No ReturnList Attributes to Display		

Add Add Child Edit Delete

When Both Framed-IP-Address and Framed-IPv6-Prefix Attributes Configured

Return Only Framed-IPv6-Prefix Attribute

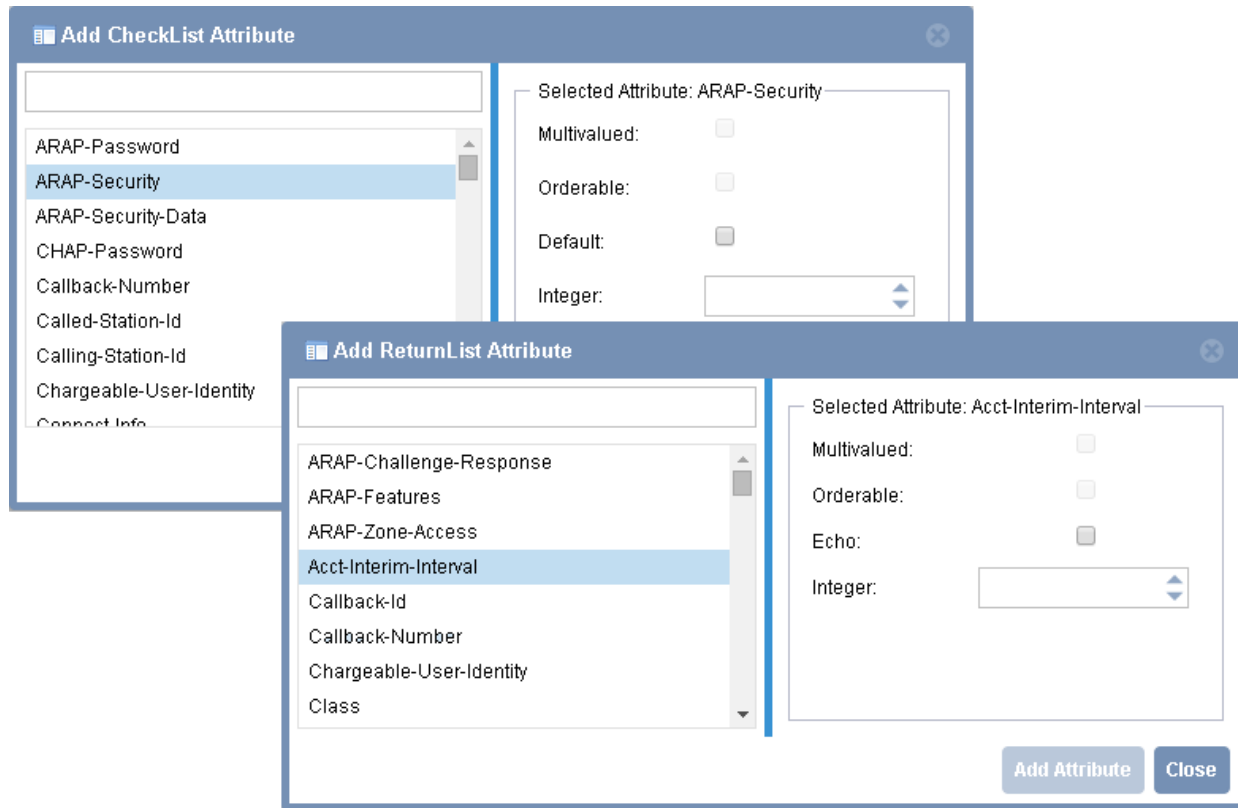
Return Both Attributes

Save Clear Cancel

3. Enter a name for the new profile in the **Name** field.
4. Optionally, enter a description for the profile in the **Description** field.
5. To add standard RADIUS attributes and parent or structured attributes to a check list or return list of the profile, click **Add** under the **CheckList** area or the **ReturnList** area.

The **Add CheckList Attribute** or **Add ReturnList Attribute** dialog box [Figure 46 on page 148](#) appears.

Figure 46: Add CheckList Attribute and Add ReturnList Attribute Dialogs



6. Select the attribute you want to add to the check list or return list from the attributes list.

You can search the attributes by entering the attribute name in the text box.

7. Select or enter a value for the selected attribute.

The dialog changes according to the attribute you choose. Some attributes require that you enter a value, string, or IP address. Other attributes require that you choose from a fixed list of values.

The **Multivalued** check box always appears dimmed, so you cannot select or clear this check box. If the **Multivalued** check box appears cleared, an attribute can have only one value. If the **Multivalued** check box appears selected, you can add multiple values for the attribute.

The **Orderable** check box always appears dimmed, so you cannot select or clear this check box. If the **Orderable** check box appears selected, you can define the order of the multi-valued attributes. If the **Orderable** check box appears cleared, the attribute is neither a multi-valued attribute nor an orderable attribute.

(Check list attributes only) To set the attributes value to the default value (which is useful in situations where the attribute is not included in the RADIUS request), select the **Default** check box.

(Return list single-valued attributes only) If you do not want to specify a particular value, but want to make sure that whatever value of the attribute appears in the RADIUS request is echoed to the client in the RADIUS response, select the **Echo** check box.

NOTE: The echo property is disabled for multi-valued return list attributes. The echo property is also disabled for the Framed-IPv6-Address attribute regardless of its multi-value setting.

NOTE: You cannot define multiple instances of Framed-IPv6-Address attributes in a return list or check list. The Framed-IPv6-Address attribute can appear only once in a return list or check list.

8. Click **Add Attribute** to add this AVP to the list.
9. Repeat steps 6 through 8 to add more attributes to the return list or check list.
10. When you are finished adding AVPs, click **Close**.

The **CheckList** area or the **ReturnList** area in the **Create Profile** pane (Figure 45 on page 147) displays the updated list of selected attributes.

You can modify the return or check list by using the **Edit** and **Delete** buttons. You can reorder the attributes by selecting each attribute and using the **Up** or **Down** arrow.

NOTE: The **Up** arrow is disabled, if the selected attribute is not orderable or if the selected attribute is already the first value. The **Down** arrow is disabled, if the selected attribute is not orderable or if the selected attribute is already the last value.

11. Optionally, if you have added a structured or parent attribute to the profile, add its subattributes to the structured or parent attribute. For more information about how to add subattributes to a structured or parent attribute, see [“Adding Subattributes to a Structured Attribute” on page 128](#).
12. If you have added both Framed-IP-Address and Framed-IPv6-Prefix attributes to the return list, specify whether to return only the Framed-IPv6-Prefix attribute or both Framed-IP-Address and Framed-IPv6-Prefix attributes in the RADIUS response. In the **When Both Framed-IP-Address and Framed-IPv6-Prefix Attributes Configured** area, you can:
 - Select the **Return Only Framed-IPv6-Prefix Attribute** option to return only the Framed-IPv6-Prefix attribute in the RADIUS response.

- Select the **Return Both Attributes** option to return both Framed-IP-Address and Framed-IPv6-Prefix attributes in the RADIUS response.

NOTE: The **When Both Framed-IP-Address and Framed-IPv6-Prefix Attributes Configured** area is enabled only if you have added both Framed-IP-Address and Framed-IPv6-Prefix attributes to the return list.

13. Click **Save** to save the profile configuration.

The **Profiles List** page ([Figure 44 on page 146](#)) displays an updated list of profile entries.

Editing a Profile

To edit a profile using the Web GUI:

1. Select **RADIUS Configuration > Profiles**.

The **Profiles List** page ([Figure 44 on page 146](#)) appears.

2. Select the profile entry that you want to edit.

The **Selected Profile** pane ([Figure 47 on page 151](#)) displays the settings configured for the profile.

Figure 47: Selected Profile Pane

The screenshot shows the Juniper Steel-Belted Radius Carrier web interface. The top navigation bar includes 'Home', 'RADIUS Configuration', 'Diameter Configuration', and 'Tools'. The 'Profiles List' pane is open, showing a table with two entries: PROFILE1 (Example profile) and PROFILE2 (Example configuration). Below the table, the 'Selected Profile: PROFILE1' pane is displayed. It contains fields for 'Name' (PROFILE1) and 'Description' (Example profile). Under the 'Attributes' section, there is a 'CheckList' table with columns 'Attribute', 'Value', and 'Default'. Below this table are buttons: 'Add', 'Add Child', 'Edit', and 'Delete'. To the right, the 'ReturnList' section has a table with columns 'Attribute', 'Value', and 'Echo', and buttons: 'Add', 'Add Child', 'Edit', and 'Delete'. At the bottom of the 'ReturnList' section, there is a checkbox labeled 'When Both Framed-IP-Address and Framed-IPv6-Prefix Attributes Configured' with options 'Return Only Framed-IPv6-Prefix Attribute' and 'Return Both Attributes'. At the bottom of the 'Selected Profile' pane are buttons: 'Save', 'Reset', and 'Cancel'.

3. Edit the settings for the profile entry as appropriate.

For information about the fields in the **Selected Profile** pane, see [“Adding a Profile” on page 146](#).

NOTE: You cannot edit the name of the profile.

4. Click **Save** to save the changes.

The **Profiles List** page ([Figure 44 on page 146](#)) displays an updated list of profile entries.

Removing a Profile

To delete a profile using the Web GUI:

1. Select **RADIUS Configuration > Profiles**.

The **Profiles List** page ([Figure 44 on page 146](#)) appears.

2. Select the profile entry that you want to delete.
3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to confirm the delete request.

The **Profiles List** page ([Figure 44 on page 146](#)) displays an updated list of profile entries.

NOTE: Do not delete a profile that is assigned to a user. If you delete an active profile, the attributes defined in the profile are removed from all user's settings, which is likely to result in authentication failures.

Administering Proxy RADIUS

IN THIS CHAPTER

- Proxy RADIUS Overview | 153
- Adding a Proxy Target | 160
- Editing a Proxy Target | 164
- Deleting a Proxy Target | 166
- Steel-Belted Radius Carrier as a Target | 166

This chapter presents an overview of proxy RADIUS and describes how to set up proxy targets. This chapter contains these topics:

Proxy RADIUS Overview

Steel-Belted Radius Carrier can forward a RADIUS request to another server for processing and relay the other server's result back to its client. Steel-Belted Radius Carrier is acting as a *proxy* for the *target server*, and that Steel-Belted Radius Carrier is *proxy-forwarding* the request to the target server. The IP address of the client originating the RADIUS packet can be determined by one or more attributes: NAS-IPv6-Address, NAS-IP-Address, or NAS-Identifier. For NAS-Identifier, the address is determined from the setting in the configuration database. If none of the addresses match the source address of the UDP packet, it is assumed that the RADIUS request has been proxied.

Any Steel-Belted Radius Carrier server can act as proxy or target for authentication or accounting messages (or both).

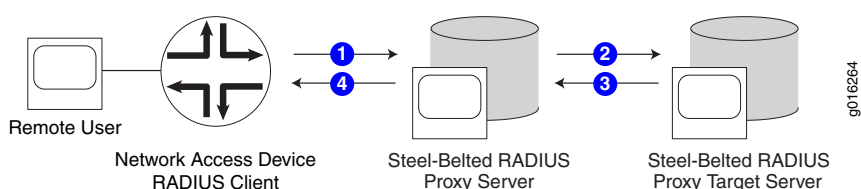
NOTE: The change of authorization/disconnect message (CoA/DM) feature does not support proxy RADIUS.

Proxy RADIUS Authentication

Figure 48 on page 154 illustrates how RADIUS authentication messages are forwarded by proxy:

1. A network access server (RADIUS client) sends an authentication request to a RADIUS proxy server.
2. The proxy RADIUS server forwards the message to a RADIUS target server.
3. The target RADIUS server performs the authentication services indicated by the message, then returns a response message to the proxy RADIUS server.
4. The proxy RADIUS server relays the response message to the RADIUS client.

Figure 48: RADIUS Proxy Forwarding



Proxy RADIUS Accounting

RADIUS accounting messages are forwarded by proxy as follows:

1. A RADIUS server receives an accounting request.
2. Depending on its configuration, the RADIUS server forwards the accounting message to a target server, records accounting attributes locally on the proxy server, or records the information in both places.
3. If the proxy server does not receive an acknowledgement of the forwarded packet, it periodically re-sends the packet according to its retry policy.
4. When the target server acknowledges the request, the proxy server forwards an acknowledgement to the RADIUS client. SBR can also be configured to respond immediately to the NAS.

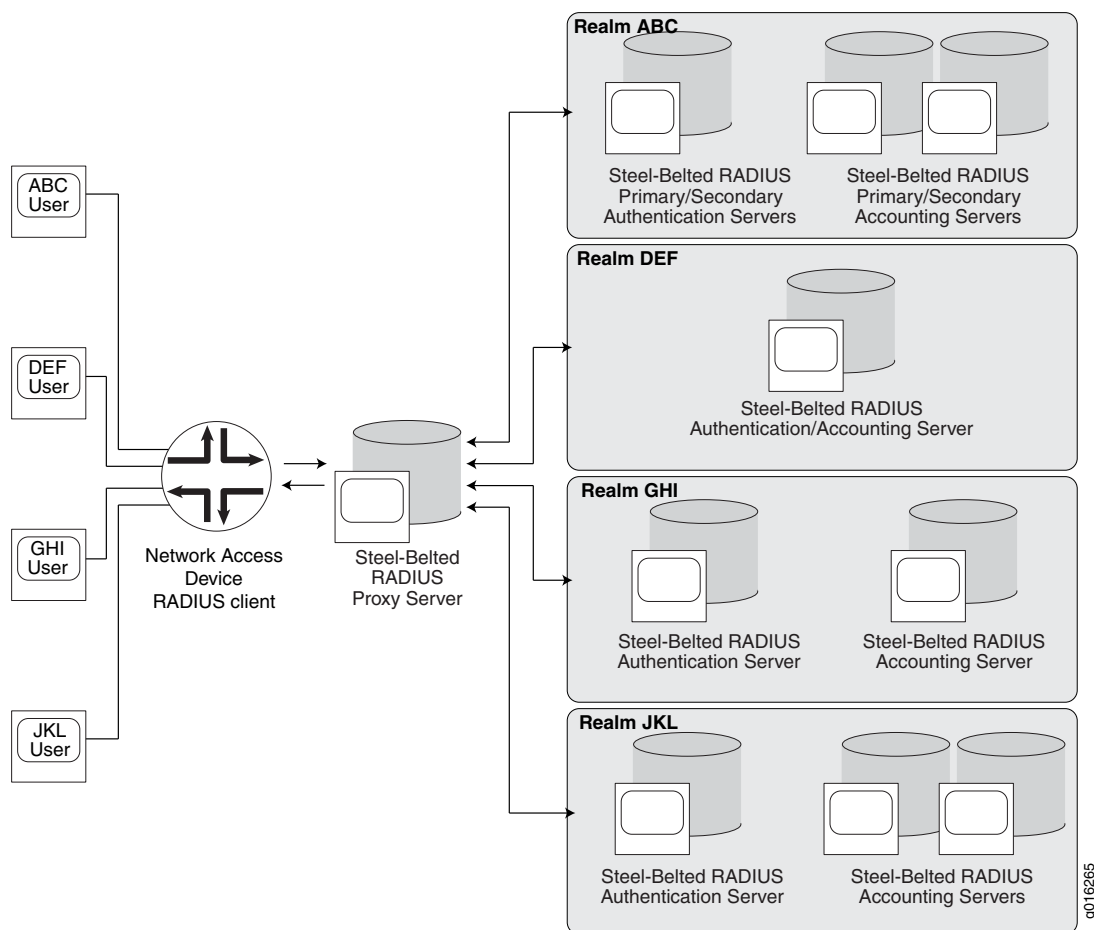
Proxy RADIUS Realms

Proxy RADIUS realms are pools of RADIUS servers to which Steel-Belted Radius Carrier can forward RADIUS requests. Proxy RADIUS realms can be configured to support workload distribution, redundancy, fault tolerance, retry policies, primary-secondary server roles, and separation of authentication and accounting responsibilities by server. A carrier with Steel-Belted Radius Carrier installed on its LAN might create a realm that consists of all the RADIUS servers owned by a particular service provider. In this case, the RADIUS servers are already owned and maintained by the service provider; the realm simply organizes the routing of RADIUS packets from Steel-Belted Radius Carrier to these servers.

The carrier might define several such realms, one for each service provider that employs its services. If a service provider's network is extremely large, a carrier might decide to use several realms to represent a single service provider. For each of these realms, it is possible to define an independent set of conventions for storing, forwarding, routing, and filtering the RADIUS requests that enter the Steel-Belted Radius Carrier server.

For more information, see [Figure 49 on page 155](#) and [“Configuring a Proxy RADIUS Realm” on page 201](#).

Figure 49: RADIUS Server and Proxy Realms



Target Selection within a Realm

For proxy RADIUS realms, after the destination realm is identified, Steel-Belted Radius Carrier must select a target within the realm. Target selection depends upon a number of factors, all of which you can set up in advance by editing the realm configuration files on the Steel-Belted Radius Carrier server: **proxy.ini**, **radius.ini**, **filter.ini**, and one **RealmName.pro** file per realm.

After the target is selected, Steel-Belted Radius Carrier matches the target name with a proxy entry in its database. Using the data in this entry (IP address, UDP port, shared secret) Steel-Belted Radius Carrier

establishes a connection between itself and the target, and proxy-forwards the RADIUS request. You can configure the realm so that all realm routing information and delimiters are stripped from the Username before forwarding.

The target processes the request as it normally would for RADIUS authentication or accounting. In the case of authentication, Steel-Belted Radius Carrier waits for a response from the target, then relays this response to its RADIUS client.

Message-Authenticator Support

The Message-Authenticator attribute enables Steel-Belted Radius Carrier to determine whether the packet received is from an actual proxy server. It might also sign the forward request.

Steel-Belted Radius Carrier can be configured to use the Message-Authenticator attribute when forwarding packets using proxy RADIUS. It can also be configured to validate or ignore the Message-Authenticator if included in the packets received.

Proxy Fast-Fail

During proxy forwarding, Steel-Belted Radius Carrier acts as the RADIUS client of another RADIUS server. Since RADIUS clients take responsibility for delivering RADIUS packets, all of them have a *retry policy* that determines how often and for how long they continue to try to deliver a packet until they receive the response that they expect from the RADIUS server. This includes the Steel-Belted Radius Carrier server when it acts as the RADIUS client of a proxy RADIUS target server.

Steel-Belted Radius Carrier provides a *fast-fail* option for proxy RADIUS realms. This fast-fail feature saves Steel-Belted Radius Carrier from continuing to send packets to a target server that appears to be down temporarily. For example, if Steel-Belted Radius Carrier is sending a packet to a target and it is not getting the timely response it expects, it periodically tries to send the packet until it reaches the number of tries in its retry policy. If it still has not received a response from the target at that point, Steel-Belted Radius Carrier removes the target from the active list and places it on the fast-fail list.

Once a target is presumed down (on the fast-fail list), Steel-Belted Radius Carrier directs proxy requests to another target in the same realm, if available. It does not wait for responses from the failed target. However, Steel-Belted Radius Carrier can be configured to send strobe requests periodically to the failed target server to detect when that server comes back up. Once a response is received to one of these strobe requests, the target server is removed from the fast-fail list. When the fast-fail timer expires for a target, it is placed back on the active list. Strobe requests are sent to the failed target server only if there are proxy requests addressed to its realm, the `StrobeEnable` parameter is set to 1, and the `RetryInterval` has been met or exceeded.

We strongly recommend that you specify a `[FastFail]` section in each proxy RADIUS realm configuration (`.pro`) file. The `[FastFail]` section permits you to fine tune retry policies for individual realms, or for specific

targets within realms. Any [FastFail] settings that you supply in a **.pro** file override the current **ProxyFastFail** setting.

The **radius.ini** file offers a **ProxyFastFail** setting for single-target proxy entries that are not a member of any realm. **ProxyFastFail** has an integer value, usually 1800. If a target remains on the fast-fail list longer than this number of seconds, it is automatically removed from the fast-fail list. If conditions warrant, a target may be returned to the fast-fail list at any time.

For information about configuring the **radius.ini** file to support the fast-fail feature, see the *SBR Carrier Reference Guide*.

Static Proxy Accounting

Static proxy accounting allows you to send copies of certain types of accounting messages to proxy RADIUS realms, as well as to the normal routing of the original accounting message. The number of copies is not limited.

Static proxy accounting does not prevent the request from being dynamically routed for RADIUS accounting services based on Username decoration, DNIS number, or attribute mapping, nor does it prevent local logging or other accounting methods from occurring. If static proxy-forwarding fails (due to a lack of response from the target), this does not prevent the original RADIUS accounting request from being acknowledged.

An important function of static proxy accounting is to ensure that **Accounting-On** and **Accounting-Off** messages can be routed to realms. A NAS (RADIUS client) normally issues these accounting messages to its RADIUS server when it goes online (**Accounting-On**) and offline (**Accounting-Off**). In such cases, all connections previously made by this NAS are considered invalid, and the RADIUS server can free resources that it allocated to those sessions.

Static proxy accounting is necessary to deliver **Accounting-On** and **Accounting-Off** messages to realms, because these messages do not contain the **User-Name** or **Called-Station-Id** attributes that Steel-Belted Radius Carrier would normally use to route packets to realms.

For example, assume the original Access-Request, an authentication message, was used to determine the realm destination for both authentication and accounting for a particular session. The attribute used to route the Access-Request may have been the **User-Name**, the **Called-Station-Id**, or any other RADIUS attribute in the Access-Request, depending on how you have configured request routing for authentication messages.

Accounting packets for this same session can be matched with the realm destination only if the server knows which session is involved (as it does in **Start**, **Stop**, and **Interim** messages). The **Accounting-On** and **Accounting-Off** messages are independent of specific sessions; therefore it is impossible to route them to realms without additional information.

By setting up static proxy accounting, and listing all realms as targets for **Accounting-On** and **Accounting-Off** messages, you can ensure that network information (such as NAS status) is sent to everyone who might require it.

Proxy AutoStop Feature

A user session can be removed from the Current Sessions table in ways other than the usual Accounting-Stop message from the NAS:

- An **Accounting-On** or **Accounting-Off** message received from the NAS causes all sessions originating from the NAS to be purged, as these messages signal that the NAS has been restarted or is going down.
- The administrator can remove users by means of the LDAP configuration interface (LCI).
- The administrator can remove users by means of the Web GUI.

Termination information must be passed on if the users exist as proxied sessions on downstream RADIUS servers because these servers must free the resources previously allocated to the sessions, which have now been terminated.

The Proxy AutoStop feature handles such cases. In addition, if you use the LCI command or dynamic authorization to free resources from the central administrative server, the appropriate messages are propagated so that the resources associated with the user in each of the downstream servers are automatically freed.

Routed Proxy Authentication

Routed proxy authentication enables you to select an authentication realm based on information in an external SQL or LDAP database and determine the routing of the authentication request. You can use this information to preauthenticate the user, select a target realm for a subsequent proxy, modify the User-Name in the proxy request, and insert attributes into the response. You can configure a proxy target statically (see [“Proxy RADIUS Authentication and Accounting” on page 485](#)), or have it determined based on information in the packet, such as NAI (decorated user-name), DNIS, or other computations (attribute mapping).

You can use this feature to:

- Allow the User-Name to determine the realm without requiring decoration.
- Centralize the mapping of attributes to the realm (the attribute mapping feature, but off-loaded to LDAP or SQL).
- Allow the User-Name to be decorated only with a final realm, mapping it to the Next-Hop realm through LDAP. This allows an enterprise to change their ISP without requiring reconfiguration of user clients. (Also, by storing shared-secret information for the Next-Hop realm in LDAP, Secure Peer Discovery functionality is available.)

Operation

Routed proxy authentication occurs when certain information is returned from an external SQL or LDAP database. The operation is determined by two variables:

- **%ProxyRealm**
- **%ProxyUser-Name**

These variables are accessible by authentication methods through the SQL and LDAP authentication plug-ins:

- If **%ProxyRealm** is not set, routed proxy authentication does not occur.
- If **%ProxyRealm** is set to a directed realm, then handle it as a directed realm request.
- If **%ProxyRealm** is set to a proxy realm, then send the request off to that proxy realm.
- If **%ProxyUserName** is set to the **User-Name** attribute, which must be sent in the proxy request. If **%ProxyUserName** is not set, the User-Name from the original request packet is used.

For more information about the SQL authentication plug-in, see [“Configuring SQL Authentication” on page 442](#). For more information about the LDAP authentication plug-in, see [“Configuring LDAP Authentication” on page 471](#).

Routed proxy authentication supports RADIUS authentication, including the RADIUS challenge process and RADIUS accounting. Password authentication may happen when the external database is accessed, or later by the proxy target itself, depending on whether the values of the **%Password** and **%ProxyRealm** variables returned from the database are blank or non-blank (see [Table 26 on page 159](#)).

Table 26: Four Scenarios for Routed Proxy Authentication

%Password value returned from database	%ProxyRealm value returned from database	Action
blank	blank	Authentication fails.
blank	non-blank	Authenticate at proxy target.
non-blank	blank	Authenticate password now; no proxy.
non-blank	non-blank	Authenticate password now; direct to appropriate realm if successful.

NOTE: Only one routed proxy is allowed per transaction; transactions cannot be nested.

Adding a Proxy Target

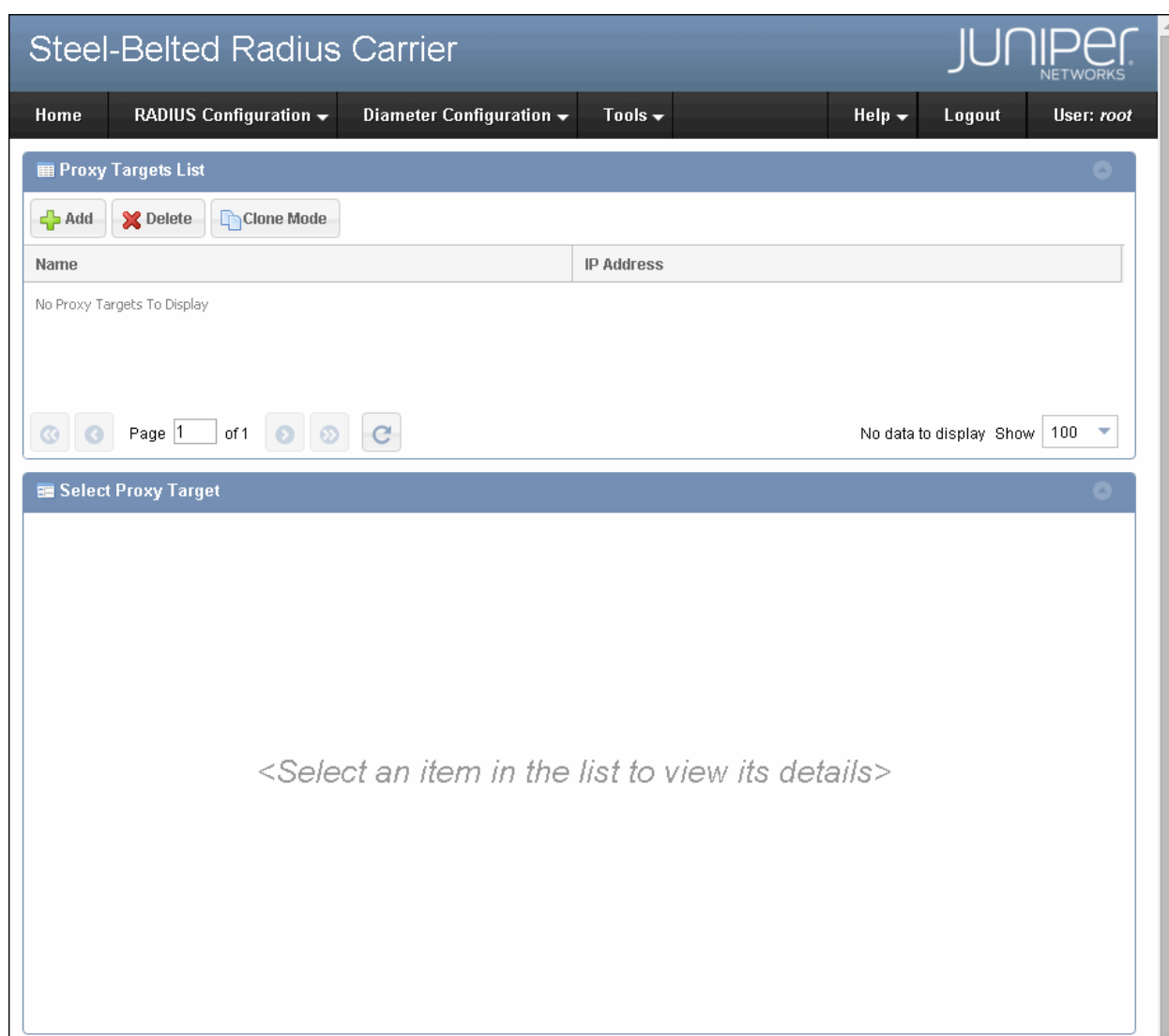
This section explains how to set up proxy forwarding from the SBR Carrier server (the proxy) to another RADIUS server (the target).

To add a proxy target using the Web GUI:

1. Select **RADIUS Configuration > Proxy Targets**.

The **Proxy Targets List** page (Figure 50 on page 160) appears.

Figure 50: Proxy Targets List Page



2. Click **Add**.

The **Create Proxy Target** pane (Figure 51 on page 161) appears with the **Basic Configuration** tab selected.

Figure 51: Create Proxy Target Pane—Basic Configuration

The screenshot shows the Steel-Belted Radius Carrier web interface. At the top is a navigation bar with the title "Steel-Belted Radius Carrier" and the Juniper Networks logo. Below the navigation bar are tabs for "Home", "RADIUS Configuration", "Diameter Configuration", "Tools", "Help", "Logout", and "User: root". The main content area has a "Proxy Targets List" section and a "Create Proxy Target" pane. The "Create Proxy Target" pane has two tabs: "Basic Configuration" (selected) and "Advanced Configuration". The "Basic Configuration" tab contains the following fields: "Name:" (text input), "Description:" (text input), "IP Address:" (text input), and "SharedSecret:" (text input). There is a "Use IPv6" checkbox next to the "IP Address" field and a "show" button next to the "SharedSecret" field. At the bottom of the pane are "Save", "Clear", and "Cancel" buttons.

3. Enter a name for the proxy target in the **Name** field.

The target name must not duplicate any other target name, realm name, or tunnel name in your SBR Carrier configuration. The entered name for a proxy target is not used in processing; SBR Carrier uses the IP address of the proxy target to route RADIUS packets.

4. Enter a description for the proxy target in the **Description** field.

5. Enter the IP address of the proxy target in the **IP Address** field.

6. Optionally, select the **Use IPv6** check box to use IPv6 addressing.

7. Enter the shared secret for the proxy target in the **SharedSecret** field.

Shared secrets are case-sensitive. If you want the characters in the shared secret to appear as you type, click **Show**. After viewing the characters, you can click **Hide** to hide the characters.

The shared secret configured for the proxy target in SBR Carrier must match the shared secret configured on the proxy target.

8. Optionally, click the **Advanced Configuration** tab (Figure 52 on page 162) to configure advanced settings for the proxy target.

Figure 52: Create Proxy Target Pane—Advanced Configuration

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and a user profile for 'root'. Below the navigation bar is a 'Proxy Targets List' section. The main area is titled 'Create Proxy Target' and contains two tabs: 'Basic Configuration' and 'Advanced Configuration'. The 'Advanced Configuration' tab is active, showing several configuration fields:

- Retry Policy Set:**
 - Number of Retries: 3 (spin box)
 - Milliseconds b/w Retries: 5000 (spin box)
- Non-Default Ports:**
 - Authentication: ☐ (spin box)
 - Accounting: ☐ (spin box)
- Accounting Requests Methods:**
 - ☒ Forward
 - ☐ Record Locally
- Use Different Shared Secret for Accounting:** ☐ (spin box) **Show**
- Use Different Shared Secret for CoA/DM:** ☐ (spin box) **Show**
- ☐ Make Available as an Authentication Method

At the bottom of the pane are three buttons: **Save**, **Clear**, and **Cancel**.

9. Enter the number of times SBR should try to reach the proxy target in the **Number of Retries** field. A request is retransmitted for the specified number of times if an acknowledgment from the target is not received. If the number of retries is exhausted, then the original request is rejected. By default, SBR Carrier retries three times before giving up.

When SBR Carrier acts as a proxy, it emulates the characteristics of a NAD. This includes the ability to retransmit a request if the first attempt does not get a timely response from the proxy target.

10. Enter the time interval between each retry in milliseconds in the **Milliseconds b/w Retries** field. By default, SBR Carrier waits 5000 milliseconds (5 seconds) between retries.
11. The port numbers configured for the proxy target in SBR Carrier must match the port numbers configured on the proxy target. By default, SBR Carrier uses port 1645 for authentication and port 1646 for

accounting. If the proxy target uses ports different from the default values for authentication or accounting, select the **Authentication** or **Accounting** check box and enter the port number you want SBR Carrier to use when exchanging RADIUS authentication or accounting information with the proxy target in the **Authentication** or **Accounting** field.

12. Specify whether you want accounting requests to be forwarded or recorded locally.

- If you select the **Forward** check box, SBR Carrier forwards the accounting transaction to the same proxy target that received the authentication transaction.
- If you select the **Record Locally** check box, SBR Carrier logs the accounting transaction locally (regardless of whether an authentication request was forwarded to the proxy target).

You can select both check boxes if you want accounting requests to be forwarded and logged locally.

13. If you want SBR Carrier to use a different shared secret for accounting when communicating with the proxy target, select the **Use Different Shared Secret for Accounting** check box and enter an accounting shared secret in the **Use Different Shared Secret for Accounting** field.

For privacy, characters are masked. You can click **Show** to display the characters in the shared secret. After viewing the characters, you can click **Hide** to hide the characters.

14. If you want SBR Carrier to use a different shared secret for authenticating COA and DM messages, select the **Use Different Shared Secret for CoA/DM** check box and enter an accounting shared secret in the **Use Different Shared Secret for CoA/DM** field.

For privacy, characters are masked. You can click **Show** to display the characters in the shared secret. After viewing the characters, you can click **Hide** to hide the characters.

15. If you want to use a proxy target as an authentication method, select the **Make Available as an Authentication Method** check box.

If you enable this option, the name of the proxy target appears in the **Authentication Methods** page ([Figure 78 on page 234](#)) as **proxy:name**. This is useful if you have user records defined on an older RADIUS server and you want to provide a seamless migration to SBR Carrier. Using the older server as a proxy RADIUS target means that RADIUS requests that arrive addressed to this target are handled by SBR Carrier automatically, without requiring end users to change their addressing conventions.

16. Click **Save** to save the proxy target configuration.

The **Proxy Targets List** page ([Figure 50 on page 160](#)) displays an updated list of proxy target entries.

Ask the administrator at the target site to log in to the target server's RADIUS configuration program and add Steel-Belted Radius Carrier as a RADIUS client of the target server. Provide this administrator with the IP address of the Steel-Belted Radius Carrier server.

Editing a Proxy Target

To edit a proxy target using the Web GUI:

1. Select **RADIUS Configuration > Proxy Targets**.

The **Proxy Targets List** page ([Figure 50 on page 160](#)) appears.

2. Select the proxy target entry that you want to edit.

The **Selected Target** pane ([Figure 53 on page 165](#)) displays the settings configured for the proxy target. Additionally, the **Selected Target** pane displays a **Validate** button in the right-hand side of the **Shared Secret**, **Use Different Shared Secret for Accounting**, and **Use Different Shared Secret for CoA/DM** fields. You can click **Validate** to verify the entered shared secret. The Web GUI displays a message indicating whether the entered shared secret matches the shared secret previously stored in the database.

Figure 53: Selected Target Pane

The screenshot displays the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user (root). The main content area is divided into two panes. The top pane, titled "Proxy Targets List", shows a table with one entry: PROXY TARGET 1 with IP Address 10.212.10.6. Below the table are pagination controls showing "Page 1 of 1" and "Displaying 1 - 1 of 1". The bottom pane, titled "Selected Target: PROXY TARGET 1", contains two tabs: "Basic Configuration" (active) and "Advanced Configuration". The Basic Configuration tab shows fields for Name (PROXY TARGET 1), Description (Example configuration), IP Address (10.212.10.6), and SharedSecret (masked with dots). There is a checkbox for "Use IPv6" and buttons for "show" and "Validate". At the bottom of the pane are "Save", "Reset", and "Cancel" buttons.

Name	IP Address
PROXY TARGET 1	10.212.10.6

Page 1 of 1 Displaying 1 - 1 of 1 Show 100

Selected Target: PROXY TARGET 1

Basic Configuration Advanced Configuration

Name: PROXY TARGET 1

Description: Example configuration

IP Address: 10.212.10.6 ☐ Use IPv6

SharedSecret: show Validate

Save Reset Cancel

3. Edit the settings for the proxy target entry as appropriate.

For information about the fields in the **Selected Target** pane, see [“Adding a Proxy Target” on page 160](#).

NOTE: You cannot edit the name of the proxy target.

4. Click **Save** to save the changes.

The **Proxy Targets List** page (Figure 50 on page 160) displays an updated list of proxy target entries.

Deleting a Proxy Target

To delete a target server from the proxy target list using the Web GUI:

1. Select **RADIUS Configuration > Proxy Targets**.

The **Proxy Targets List** page (Figure 50 on page 160) appears.

2. Select the proxy target entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to confirm the delete request.

The **Proxy Targets List** page (Figure 50 on page 160) displays an updated list of proxy target entries.

Steel-Belted Radius Carrier as a Target

This section describes how to set up proxy forwarding from some other RADIUS server (the proxy) to the Steel-Belted Radius Carrier server (the target):

1. Set up the proxy as a RADIUS client of Steel-Belted Radius Carrier.

Add the entry using the **RADIUS Clients List** page. Specify the proxy's name, its IP address, and the shared secret that you want to use for encryption between the proxy and Steel-Belted Radius Carrier.

2. Ask the administrator at the target site to log in to the proxy's RADIUS configuration program and set up Steel-Belted Radius Carrier as a proxy RADIUS target. Provide this administrator with the IP address of the Steel-Belted Radius Carrier server.

NOTE: Make sure that the same UDP port and shared secret are entered on both proxy and target sides.

Dictionaries When Steel-Belted Radius Carrier is the Target

When Steel-Belted Radius Carrier receives a packet forwarded by proxy, it consults its RADIUS client entry for that proxy server. The **Make/model** field of this entry determines which attribute dictionary Steel-Belted Radius Carrier uses.

At various different times, Steel-Belted Radius Carrier can receive requests from the same proxy server that have originated from different network access servers, possibly of different types. The single **Make/model** field that was entered for the proxy might not be adequate to handle the variety of NASs on the other side of the transaction.

One way to handle this problem is to add the originating network access servers to Steel-Belted Radius Carrier's list of RADIUS clients. Steel-Belted Radius Carrier can be configured to examine each packet forwarded by proxy for clues as to the make and model of the originating device. If clues are found, Steel-Belted Radius Carrier does everything it can to map this information to a vendor-specific dictionary, and uses this dictionary in preference to the one for the proxy.

Accepting Packets from Any Proxy

If you want Steel-Belted Radius Carrier to be able to accept proxy requests from any IP address, use the **RADIUS Clients List** page to add a special entry called **<ANY>**, and specify a shared secret. The **<ANY>** entry permits forwarded requests from any proxy to be accepted, provided the shared secret is correct.

NOTE: This feature requires that proxies are configured to use the shared secret you provide in the **<ANY>** entry.

Administering RADIUS Tunnels

IN THIS CHAPTER

- [About RADIUS Tunnels | 168](#)
- [Configuring RADIUS Tunnels | 171](#)
- [Configuring Tunnel Name Parsing | 179](#)

This chapter describes how to set up and administer RADIUS tunnels. This chapter contains these topics:

About RADIUS Tunnels

A *tunnel* is a uniquely secure type of remote connection. A tunnel passes data between a remote site and an enterprise site, providing an additional layer of encrypted protocol *wrapper* around the data. A tunnel offers authentication and encryption features that help secure the connection against network vandals and eavesdroppers. In addition, a tunnel can provide quality of service features such as guaranteed bandwidth.

NOTE: SBR Carrier does not add tunnel functionality to your network. SBR Carrier is able to support the authentication and accounting needs of any tunnels that you have already set up.

Administration and configuration of the tunnel happens at the remote site, since this is the side of the connection that requests remote access and opens the tunnel. An administrator at the remote site must configure the tunnel with various attributes: its destination IP address, what security protocols it supports, its password, and so on. These attributes are stored in a database to be retrieved when needed to set up a connection.

Storing tunnel attributes on a RADIUS server simplifies tunnel connections. At connection time, the tunnel is established by a network access server at the remote site. The NAD retrieves the tunnel configuration attributes from the RADIUS server and uses them to open the tunnel into the carrier's network. After the tunnel is open, the user can be authenticated at the carrier's network.

A RADIUS server is said to support tunnels if it has the ability to store and retrieve the configuration data that a NAD needs to open a tunnel. SBR Carrier fully supports tunnels:

- SBR Carrier can determine from the attributes in the incoming Access-Request whether the connection request involves a tunnel, and if so, which tunnel.
- SBR Carrier can store and retrieve tunnel configuration data.
- SBR Carrier can track the number of tunnels currently in use, compare to a maximum number, and refuse the connection if the number is exceeded.

Tunnel Authentication Sequence

1. SBR Carrier receives an Access-Request message.
2. SBR Carrier checks whether the Access-Request contains a **Called-Station-Id** attribute. If it does, SBR Carrier searches its database for a tunnel entry that contains the indicated telephone number in its called station ID list.

If a match between the **Called-Station-Id** and a tunnel entry can be found, SBR Carrier constructs an Access-Accept message using the Attributes list in the matching tunnel entry. It then returns the Access-Accept to the client NAD. If a match exists, then skip to step 4; if no match exists, continue with step 3.

NOTE: If realms are in use, SBR Carrier also searches for the called station ID number in its realm configuration files. If a match is found, the Access-Request is routed to the realm, and the quest for a tunnel is abandoned. For this reason, make sure that DNIS numbers are unique across all tunnel entries and across all realm configuration files.

3. If no match was found in step 2, then SBR Carrier checks whether the Access-Request contains a username in the form **User<Delimiter>TunnelName** or **TunnelName<Delimiter>User**. **<Delimiter>** is a single character that must match the server's tunnel delimiter character. The order of the realm name relative to the username must match the server's tunnel naming convention (**prefix** or **suffix**). Both of these values are determined per server (that is, all tunnels that use this server must follow the same conventions) by entering them in the **Name Parsing** page ([Figure 61 on page 180](#)). If a match exists, continue with step 4; if no match exists, then skip to step 6.
4. SBR Carrier searches its database for a tunnel entry whose name matches the incoming **TunnelName**. If a match can be found, SBR Carrier constructs an Access-Accept message using the Attributes list in the matching tunnel entry. It then returns the Access-Accept to the client NAD.
5. If SBR Carrier was able to match the Access-Request with a tunnel entry, the NAD uses the attributes returned in the Access-Accept message to open a tunnel into the enterprise site. Authentication of the

User-Name is attempted, usually at the carrier's site. If user authentication succeeds, the connection is complete. Otherwise, the user's connection request is denied.

6. If no matching tunnel entry was found in steps 2 or 3, SBR Carrier concludes that a tunnel is not involved in making this connection. It then continues with its User-Name parsing sequence determine a destination for the authentication request.

Configuring Tunnel Support

You can configure SBR Carrier to support a tunnel using the Web GUI. A tunnel entry allows you to specify a list of connection Attributes such as the tunnel password, the IP address of the NAD at the enterprise site, encryption conventions to use, and so on. You can also enter the maximum number of tunnels that can be open at one time. You need to coordinate with the administrator at the enterprise site to get some of this information.

Called Station ID

DNIS (Diald Number Information Services) refers to a capability that many network access servers have to determine and use the telephone number that was dialed to make a connection request. The RADIUS standard supports DNIS by specifying the following attributes:

- **Calling-Station-Id** is the number from which the user originated the request.
- **Called-Station-Id** is the telephone number that was dialed to make the network connection.

When setting up a tunnel entry for the SBR Carrier database, you can enter a telephone number or list of numbers. This list identifies **Called-Station-Id** attribute values that the server should expect to find in tunnel connection requests.

Dictionaries for Tunnel Support

You can use the Web GUI to create the attributes list. The available selections include attributes from all standard and vendor-specific RADIUS dictionaries installed on the SBR Carrier server.

When the server can accept a tunnel connection request, it consults the corresponding tunnel entry for the list of Attributes to return in the Access-Accept packet. SBR Carrier always returns any standard RADIUS attributes that appear in the Attributes list. It also returns any vendor-specific attributes that are appropriate for the NAD that requested the tunnel connection. Vendor-specific attributes in the Attributes list that do not apply to the requesting NAD are ignored.

Concurrent Tunnel Connections

SBR Carrier tracks the number of active connections for each tunnel. You can limit the number of concurrent connections that can be open through a specific tunnel. When a user requests a new connection through a tunnel, SBR Carrier compares the number of active connections in a tunnel to the maximum number of connections: if a new connection would exceed the limit, SBR Carrier rejects the additional connection.

For concurrent connection limits to work, each NAD that can open a tunnel must be configured for RADIUS accounting and the same SBR Carrier server must be specified for both authentication and accounting.

NOTE: If you are using a clustered solution with SSR, concurrent tunnel connections are not tracked. They are tracked individually using an in memory store on each SBR Carrier server. So if a tunnel exceeds its maximum concurrent connections on one SBR Carrier server it may be able to authenticate on another SBR Carrier server.

Configuring RADIUS Tunnels

This section describes how to enable SBR Carrier to support tunnels. When you add a tunnel entry, you are not creating a tunnel; you are enabling SBR Carrier to support an existing tunnel's authentication and accounting needs.

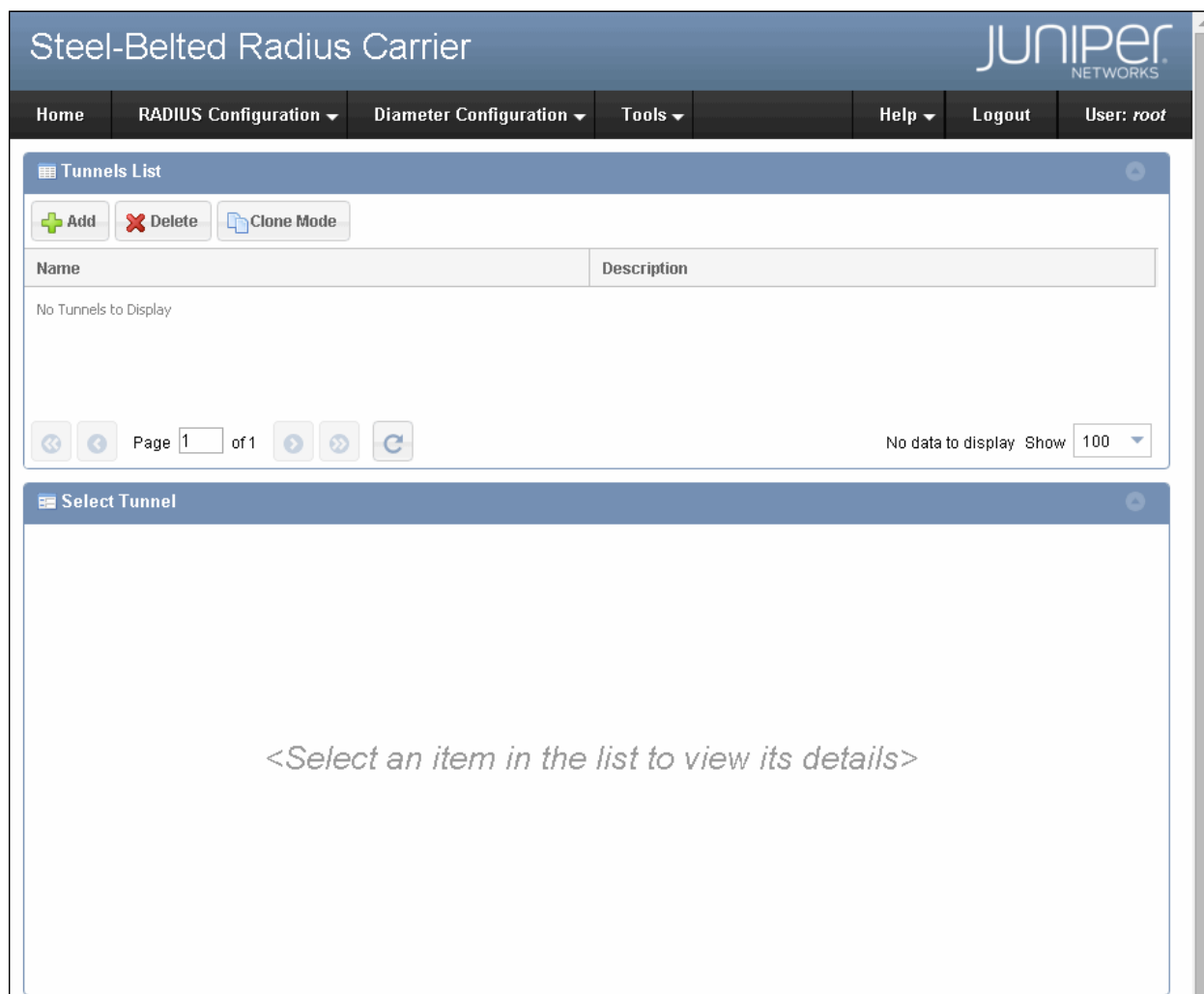
Adding a Tunnel

To add a tunnel entry using the Web GUI:

1. Select **RADIUS Configuration > Tunnels > Tunnels**.

The **Tunnels List** page (Figure 54 on page 172) appears.

Figure 54: Tunnels List Page



2. Click **Add**.

The **Create Tunnel** pane (Figure 55 on page 173) appears with the **Basic Configuration** tab selected.

Figure 55: Create Tunnel Pane—Basic Configuration

The screenshot shows the Steel-Belted Radius Carrier web interface. At the top, there is a header bar with the title "Steel-Belted Radius Carrier" and the Juniper Networks logo. Below the header is a navigation bar with links: Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. The main content area is divided into two panes. The top pane is titled "Tunnels List" and contains a table with one row showing a tunnel named "Tunnel1". The bottom pane is titled "Create Tunnel" and contains three tabs: "Basic Configuration", "Attributes", and "Called Station IDs". The "Basic Configuration" tab is selected and shows the following fields: "Name:" with a text input field, "Description:" with a text input field, and "Maximum Open Tunnels:" with a checkbox and a numeric input field. At the bottom of the "Create Tunnel" pane are three buttons: "Save", "Clear", and "Cancel".

3. Enter a name for the tunnel in the **Name** field.

Tunnel names do not need to match the actual node name of a client tunnel server. The name you assign to a tunnel must not match the name assigned to a proxy target, realm, or tunnel in your SBR Carrier configuration.

4. Enter a description for the tunnel in the **Description** field.

Tunnel descriptions are used only for administrative purposes and do not affect tunnel connections. This field is typically used to identify the user or organization that uses the tunnel.

5. If you want to limit the number of connections that can use the tunnel simultaneously, select the **Maximum Open Tunnels** check box and enter the maximum number of tunnels in the **Maximum Open Tunnels** field.

6. Click the **Attributes** tab ([Figure 56 on page 174](#)) to associate attributes and values with the tunnel.

When a tunnel is used to make a connection, the attributes associated with the tunnel are filtered according to the make or model of the RADIUS client used to establish the connection.

Figure 56: Create Tunnel Pane—Attributes

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes the title "Steel-Belted Radius Carrier" and the Juniper Networks logo. Below the navigation bar, there are tabs for "Home", "RADIUS Configuration", "Diameter Configuration", "Tools", "Help", "Logout", and "User: root". The main content area is divided into two panes: "Tunnels List" and "Create Tunnel". The "Create Tunnel" pane is active and shows three tabs: "Basic Configuration", "Attributes", and "Called Station IDs". The "Attributes" tab is selected, displaying a table with two columns: "Attribute" and "Value". The table is currently empty, with the text "No TunnelList Attributes to Display" shown below the header. At the bottom of the table, there are three buttons: "Add", "Edit", and "Delete". Below the table, there are three buttons: "Save", "Clear", and "Cancel".

Attribute	Value
No TunnelList Attributes to Display	

Add **Edit** **Delete**

Save **Clear** **Cancel**

7. Click **Add**.

The **Add Tunnel List Attribute** dialog box ([Figure 57 on page 175](#)) appears.

Figure 57: Add Tunnel List Attribute Dialog

8. Select or enter a value for the selected attribute.

The dialog changes according to the attribute you choose. Some attributes require that you enter a value, string, or IP address. Other attributes require that you choose from a fixed list of values.

The **Multivalued** check box always appears dimmed, so you cannot select or clear this check box. If the **Multivalued** check box appears cleared, an attribute can have only one value. If the **Multivalued** check box appears selected, you can add multiple values for the attribute.

The **Orderable** check box always appears dimmed, so you cannot select or clear this check box. If the **Orderable** check box appears selected, you can define the order of the multi-valued attributes. If the **Orderable** check box appears cleared, the attribute is neither a multi-valued attribute nor an orderable attribute.

9. Click **Add Attribute**.
10. Repeat steps 8 and 9 to add more attributes to the tunnel.
11. When you are finished adding AVPs, click **Close**.

The **Attributes** tab (Figure 56 on page 174) displays an updated list of selected attributes.

You can modify the attributes by using the **Edit** and **Delete** buttons. You can reorder the attributes by selecting each attribute and using the **Up** or **Down** arrow.

NOTE: The **Up** arrow is disabled, if the selected attribute is not orderable or if the selected attribute is already the first value. The **Down** arrow is disabled, if the selected attribute is not orderable or if the selected attribute is already the last value.

12. Optionally, click the **Called Station IDs** tab (Figure 58 on page 176) to add one or more called station ID numbers for the tunnel.

A called station ID number is a telephone number that was dialed to make a network connection. The called station ID number list identifies the **Called-Station-Id** attribute values that the server expects to find in tunnel connection requests.

Figure 58: Create Tunnel Pane—Called Station IDs

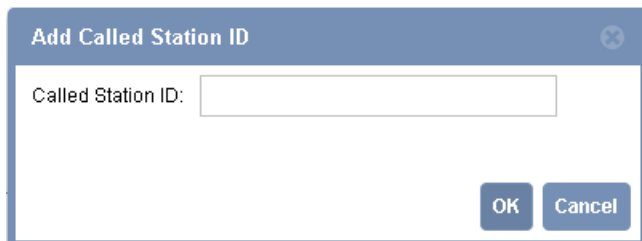
The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes the title "Steel-Belted Radius Carrier" and the Juniper Networks logo. Below the navigation bar are tabs for "Home", "RADIUS Configuration", "Diameter Configuration", "Tools", "Help", "Logout", and "User: root". The main content area is divided into two panes: "Tunnels List" and "Create Tunnel". The "Create Tunnel" pane is active and shows three sub-tabs: "Basic Configuration", "Attributes", and "Called Station IDs". The "Called Station IDs" tab is selected, displaying a large empty text area for input. At the bottom right of the "Called Station IDs" tab are buttons for "New", "Edit", and "Delete". At the bottom of the "Create Tunnel" pane are buttons for "Save", "Clear", and "Cancel".

Copyright © 2006-2015, Juniper Networks, Inc. All rights reserved.

13. Click **New**.

The **Add Called Station ID** dialog box (Figure 59 on page 177) appears.

Figure 59: Add Called Station ID Dialog

A dialog box titled "Add Called Station ID" with a close button in the top right corner. It contains a text input field labeled "Called Station ID:" and two buttons at the bottom: "OK" and "Cancel".

14. Enter the number you want to use in the **Called Station ID** field.

15. Click **OK**.

16. Repeat steps 13 through 15 to add more called station ID numbers for the tunnel.

The **Called Station IDs** tab (Figure 58 on page 176) displays an updated list of called station ID numbers.

You can modify the called station ID numbers by using the **Edit** and **Delete** buttons.

17. Click **Save** to save the tunnel configuration.

The **Tunnels List** page (Figure 54 on page 172) displays an updated list of tunnel entries.

Editing a Tunnel

To edit a tunnel entry using the Web GUI:

1. Select **RADIUS Configuration > Tunnels > Tunnels**.

The **Tunnels List** page (Figure 54 on page 172) appears.

2. Select the tunnel entry that you want to edit.

The **Selected Tunnel** pane (Figure 60 on page 178) displays the settings configured for the tunnel.

Figure 60: Selected Tunnel Pane

The screenshot displays the Steel-Belted Radius Carrier web interface. At the top, the title "Steel-Belted Radius Carrier" is visible, along with the Juniper Networks logo. A navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user "root".

The main content area is divided into two sections. The top section, titled "Tunnels List", contains buttons for "Add", "Delete", and "Clone Mode". Below these is a table with two columns: "Name" and "Description". The table lists "TUNNEL 1" with the description "Example configuration". At the bottom of this section are pagination controls showing "Page 1 of 1" and a "Displaying 1 - 1 of 1" status.

The bottom section, titled "Selected Tunnel: TUNNEL 1", contains three tabs: "Basic Configuration", "Attributes", and "Called Station IDs". The "Basic Configuration" tab is active, showing fields for "Name" (TUNNEL 1), "Description" (Example configuration), and a checkbox for "Maximum Open Tunnels" with a corresponding numeric input field. At the bottom of this section are "Save", "Reset", and "Cancel" buttons.

3. Edit the settings for the tunnel entry as appropriate.

For more information about fields in the **Selected Tunnel** pane, see [“Adding a Tunnel” on page 172](#).

NOTE: You cannot edit the name of the tunnel.

4. Click **Save** to save the changes.

The **Tunnels List** page ([Figure 54 on page 172](#)) displays an updated list of tunnel entries.

Deleting a Tunnel

To delete a tunnel entry from the SBR Carrier database:

1. Select **RADIUS Configuration > Tunnels > Tunnels**.

The **Tunnels List** page ([Figure 54 on page 172](#)) appears.

2. Select the tunnel entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to confirm the delete request.

The **Tunnels List** page ([Figure 54 on page 172](#)) displays an updated list of tunnel entries.

Configuring Tunnel Name Parsing

Tunnel name parsing enables SBR Carrier to determine whether the name string provided by a user includes a tunnel name by looking for the character configured as the delimiter for tunnel information. Tunnel name parsing options apply to all tunnels maintained by SBR Carrier. You cannot set name parsing options for individual tunnels.

To configure tunnel name parsing using the Web GUI:

1. Select **RADIUS Configuration > Tunnels > Name Parsing**.

The **Name Parsing** page (Figure 61 on page 180) appears.

Figure 61: Name Parsing Page

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration ▼ Diameter Configuration ▼ Tools ▼ Help ▼ Logout User: root

Name Parsing

☐ None
☐ Tunnel Name is Prefix
☒ Tunnel Name is Suffix

Parsing Delimiter:

Save Reset Refresh

2. Select the corresponding tunnel name parsing option:

- **None**—Tunnel name parsing is not supported. If you choose this option, the tunnel authentication sequence is bypassed for each Access-Request; the server uses the standard username and password authentication sequence only.
- **Tunnel Name is Prefix**—If the tunnel delimiter character is detected, the User-Name attribute value is assumed to be ***TunnelName<PrefixDelimiter>User***.
- **Tunnel Name is Suffix**—If the tunnel delimiter character is detected, the User-Name attribute value is assumed to be ***User<SuffixDelimiter>TunnelName***.

The option you choose applies to all tunnels defined on the server.

3. If you have selected the **Tunnel Name is Prefix** or **Tunnel Name is Suffix** option button, use the **Parsing Delimiter** field to specify the character used to separate the tunnel name and the username.

The default delimiter character for tunnel name parsing is @.

NOTE: Choose different delimiter characters and different prefix or suffix name parsing conventions for tunnels and for proxies or realms.

4. Click **Save** to save the tunnel name parsing configuration.

Administering Address Pools

IN THIS CHAPTER

- Address Pools for Standalone Servers versus Servers in a SSR Cluster | 182
- Address Pool Files | 183
- Adding an IPv4 Address Pool | 183
- Editing an IPv4 Address Pool | 186
- Deleting an IPv4 Address Pool | 188
- Specifying an IP Address Pool for User/Profile Records | 188
- NAD-Specific IP Address Pools | 189
- Service-Level IP Address Pools | 190
- Specifying IP Address Assignment from a DHCP Server | 191

This chapter describes how to set up IPv4 address pools for SBR Carrier servers running standalone. This chapter contains these topics:

Address Pools for Standalone Servers versus Servers in a SSR Cluster

The information in this section applies only to SBR Carrier servers running standalone. For servers running the optional Session State Register, information about IP address pools is centrally stored in a dedicated table in the SSR database. For information about configuring IP address pools for the Session State Register see, [“Using IP Address and IP Address Pool Scripts” on page 646](#) and the *SBR Carrier Installation Guide*.

NOTE: Please contact Juniper Networks Technical Support if you need address pools larger than 65,535 (2^{16}) addresses.

Address Pool Files

The following files establish settings for IPv4 and IPv6 address pools. For more information about these files, refer to the *SBR Carrier Reference Guide*.

Table 27: Address Pool Files

File Name	Function
dhcp.ini	Configures DHCP address pools so that IP addresses can be assigned from a back-end DHCP server.
pool.dhc	Configures specific DHCP address pools, where pool is the name of an address pool listed in dhcp.ini .
radius.ini	Specifies (among other things) the suffixes used to set up NAD-specific IP pools.

Adding an IPv4 Address Pool

An IP address pool consists of one or more ranges of IPv4 addresses. You can add or delete ranges and set an optional description for each address pool.

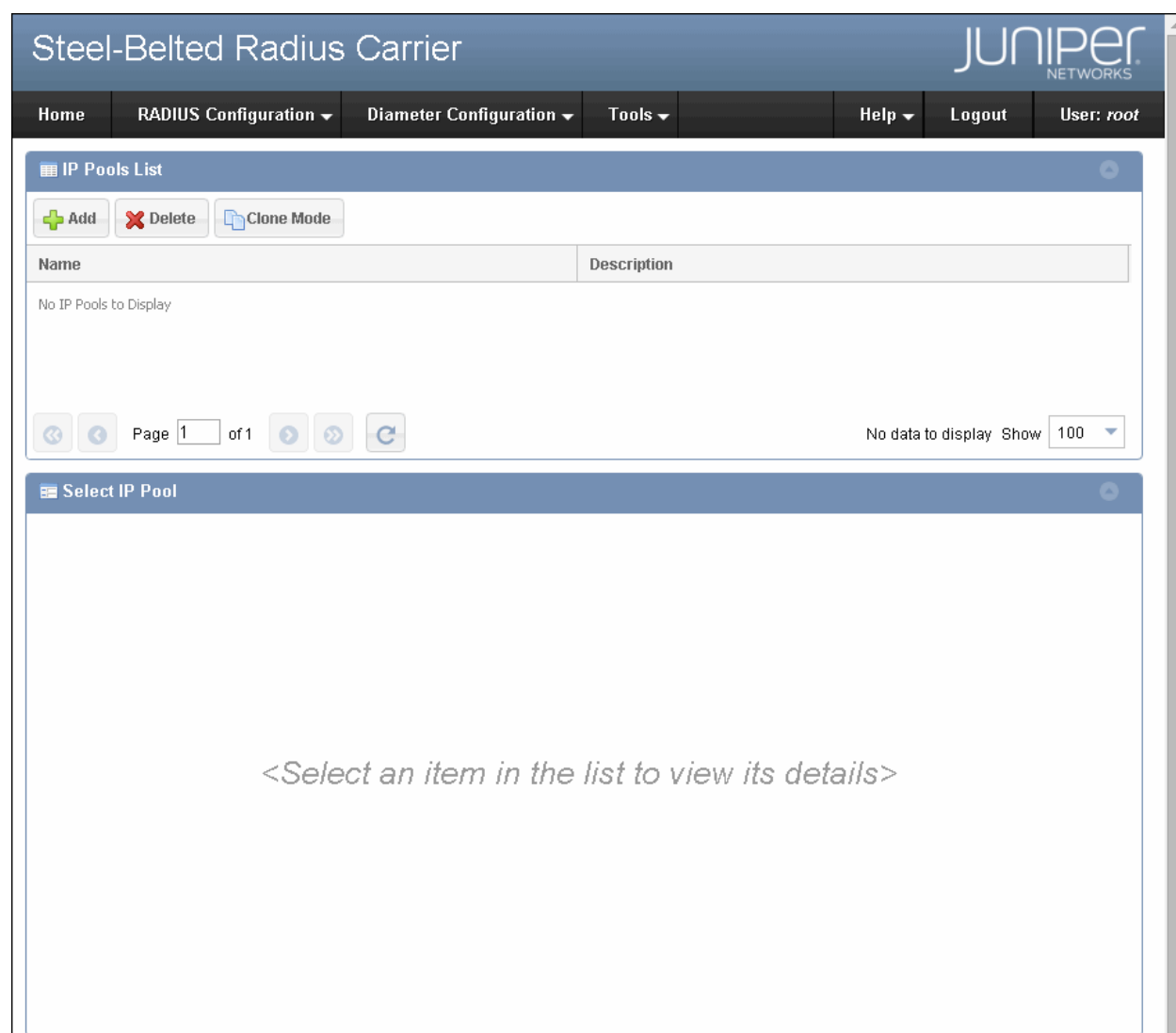
NOTE: IPv4 address pools can be used to assign IPv6 prefixes using the **Framed-IPv6-Prefix** attribute. For more information about the usage of the **Framed-IPv6-Prefix** attribute, see the *SBR Carrier Reference Guide*.

To add an IPv4 address pool using the Web GUI:

1. Select **RADIUS Configuration > Address Pools > IP**.

The **IP Pools List** page ([Figure 62 on page 184](#)) appears.

Figure 62: IP Pools List Page



NOTE: The **IP Pools List** page enables you to view the name of IPv4 address pools configured for an SSR cluster. To create and view details of IP pools in the SSR cluster you need to use administrative scripts. For details see [“Using IP Address and IP Address Pool Scripts” on page 646](#).

2. Click **Add**.

The **Create IP Pool** pane (Figure 63 on page 185) appears.

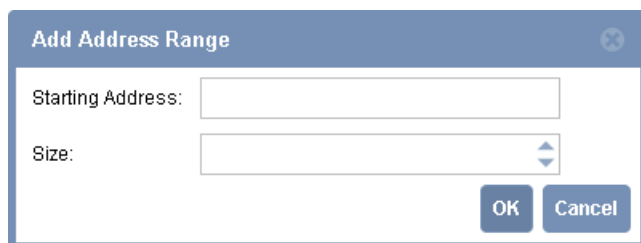
Figure 63: Create IP Pool Pane

The screenshot shows the Steel-Belted Radius Carrier web interface. At the top is a navigation bar with the title "Steel-Belted Radius Carrier" and the Juniper Networks logo. Below the navigation bar are tabs for "Home", "RADIUS Configuration", "Diameter Configuration", "Tools", "Help", "Logout", and "User: root". The main content area is divided into two panes. The top pane is titled "IP Pools List" and contains a table with one row. The bottom pane is titled "Create IP Pool" and contains a form with the following fields: "Name:" (text input), "Description:" (text input), and "Address Ranges" (a table with two columns: "Starting Address" and "Size"). The "Address Ranges" table is currently empty, displaying the message "No Address Ranges to Display". At the bottom of the "Create IP Pool" pane are three buttons: "Add", "Edit", and "Delete". At the bottom of the main content area are three buttons: "Save", "Clear", and "Cancel".

3. Enter the name for the IPv4 address pool in the **Name** field.
4. Optionally, enter a description for the address pool in the **Description** field.
5. Click **Add** in the **Address Ranges** area.

The **Add Address Range** dialog box (Figure 64 on page 186) appears.

Figure 64: Add Address Range Dialog

A dialog box titled "Add Address Range" with a close button in the top right corner. It contains two input fields: "Starting Address:" followed by a text box, and "Size:" followed by a spinner box. At the bottom right are "OK" and "Cancel" buttons.

6. Enter the starting address for the IPv4 address range in the **Starting Address** field.

7. Enter the number of addresses for the range in the **Size** field.

8. Click **OK**.

The **Create IP Pool** pane (Figure 63 on page 185) displays an updated list of address ranges.

You can modify the address range by using the **Edit** and **Delete** buttons.

9. Repeat steps 5 through 8 to add more address ranges for the IPv4 address pool.

10. Click **Save** to save the IPv4 address pool configuration.

The **IP Pools List** page (Figure 62 on page 184) displays an updated list of address pool entries.

Editing an IPv4 Address Pool

To edit an IPv4 address pool using the Web GUI:

1. Select **RADIUS Configuration > Address Pools > IP**.

The **IP Pools List** page (Figure 62 on page 184) appears.

2. Select the IPv4 address pool entry that you want to edit.

The **Selected IP Pool** pane (Figure 65 on page 187) displays the settings configured for the address pool.

Figure 65: Selected IP Pool Pane

The screenshot shows the Juniper Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and a user profile for 'root'. The main content area is divided into two sections. The top section, 'IP Pools List', contains buttons for 'Add', 'Delete', and 'Clone Mode', followed by a table with one entry: 'IP POOL 1' with the description 'Example Configuration'. Below the table are pagination controls showing 'Page 1 of 1' and a 'Show 100' dropdown. The bottom section, 'Selected IP Pool: IP POOL 1', contains input fields for 'Name' (IP POOL 1) and 'Description' (Example Configuration). Below these is an 'Address Ranges' table with one entry: '10.212.3.1' with a 'Size' of '2'. At the bottom of this section are 'Add', 'Edit', and 'Delete' buttons. The very bottom of the interface has 'Save', 'Reset', and 'Cancel' buttons.

Name	Description
IP POOL 1	Example Configuration

Starting Address	Size
10.212.3.1	2

3. Edit the settings for the address pool entry as appropriate.

For information about the fields in the **Selected IP Pool** pane, see [“Adding an IPv4 Address Pool” on page 183](#).

NOTE: You cannot edit the name of the address pool.

4. Click **Save** to save the changes.

The **IP Pools List** page ([Figure 62 on page 184](#)) displays an updated list of address pool entries.

Deleting an IPv4 Address Pool

To delete an IPv4 address pool using the Web GUI:

1. Select **RADIUS Configuration > Address Pools > IP**.

The **IP Pools List** page (Figure 62 on page 184) appears.

2. Select the IPv4 address pool entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to confirm the delete request.

The **IP Pools List** page (Figure 62 on page 184) displays an updated list of address pool entries.

Specifying an IP Address Pool for User/Profile Records

The **Framed-IP-Address** return list attribute controls how the server assigns an IP address to a user making a connection. When you add or edit the **Framed-IP-Address** attribute in the **Users** or **Profiles** page of Web GUI, the **Add Attribute** dialog box (Figure 66 on page 188) allows you to choose an IP address pool instead of specifying an IP address.

Figure 66: Editing the Framed-IP-Address Using Web GUI

Add ReturnList Attribute

fra

- Framed-AppleTalk-Link
- Framed-AppleTalk-Network
- Framed-AppleTalk-Zone
- Framed-Compression
- Framed-IP-Address**
- Framed-IP-Netmask
- Framed-IPX-Network
- Framed-IPv6-Pool
- Framed-IPv6-Prefix

Selected Attribute: Framed-IP-Address

Multivalued: ☐

Orderable: ☐

Echo: ☐

☐ IP Address ☒ IP Address Pool

IP Address Pool: IP POOL 1

Add Attribute Close

NAD-Specific IP Address Pools

Steel-Belted Radius Carrier enables you to define IP address pools that are specific to the NAD (RADIUS client) from which the user request was received. You can also define a set of suffixes that define categories of pools. For example, a pool category might correspond to the kinds of services available to users in that category. You might decide to define categories called **Bronze**, **Silver**, and **Gold**, indicating increasing packet routing priorities.

To create a NAD-specific address pool using the Web GUI:

1. Add an IPv4 address pool.

For information about how to add an IPv4 address pool, see [“Adding an IPv4 Address Pool” on page 183](#).

2. In the **RADIUS Clients List** page ([Figure 19 on page 105](#)), use the **Address Pool** check box and the **Address Pool** list to associate the IP address pool with the appropriate NAD.

3. Assign the user to a NAD-specific IP address pool and suffix.

Create this association in the **Native Users List** page ([Figure 29 on page 123](#)), **Unix Users List** page ([Figure 39 on page 136](#)), or the **Profiles List** page ([Figure 44 on page 146](#)) by adding a **Framed-IP-Address** check list or return list attribute with a value of **pool associated with RAS Client** ([Figure 67 on page 189](#)).

Figure 67: Assigning the User to a NAD-Specific IP Address Pool

The screenshot shows a web-based configuration interface titled "Add ReturnList Attribute". On the left, there is a list of attributes with "Framed-IP-Address" selected. On the right, configuration options for the selected attribute are shown. The "Selected Attribute" is "Framed-IP-Address". There are checkboxes for "Multivalued:", "Orderable:", and "Echo:". Below these are radio buttons for "IP Address" and "IP Address Pool", with "IP Address Pool" being selected. An "IP Address Pool:" dropdown menu is open, displaying "pool associated with RAS Client" as the selected item, with other visible options being "IP POOL 1" and "pool associated with RAS Client". At the bottom right are "Add Attribute" and "Close" buttons.

Service-Level IP Address Pools

Steel-Belted Radius Carrier enables you to define a set of suffixes that define categories of IP address pools. For example, a pool category might correspond to the kinds of services available to users in that category. You might decide to define categories called **Bronze**, **Silver**, and **Gold** to identify different packet routing priorities.

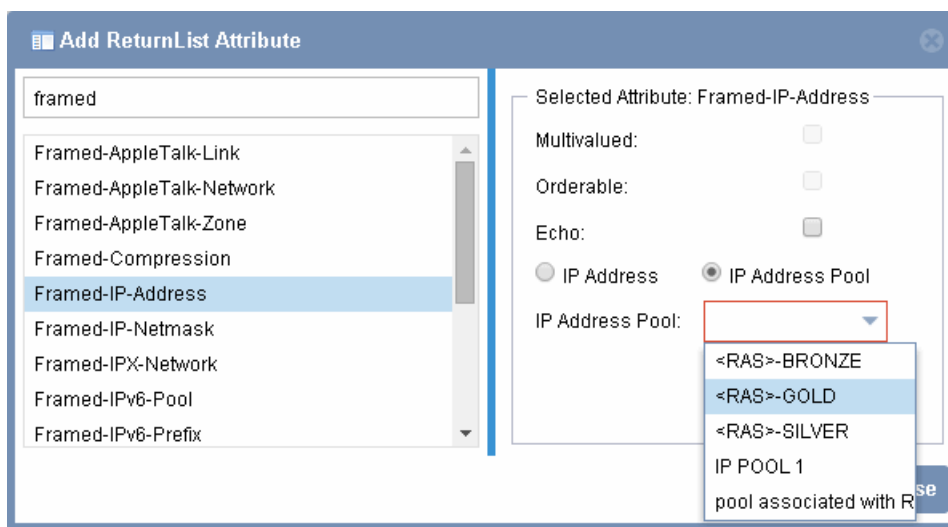
To create a set of service-level address pools using the Web GUI:

1. Define suffixes for the various service-level address pools in the [IPPoolSuffixes] section of **radius.ini**.
For example:

```
[IPPoolSuffixes]
-GOLD
-SILVER
-BRONZE
```

2. Define IP address pools using the suffixes configured in the [IPPoolSuffixes] section of **radius.ini**.
3. In the **RADIUS Clients List** page (Figure 19 on page 105), use the **Address Pool** check box and the **Address Pool** list to associate the address pool with the appropriate NAD.
4. Assign the user to a NAD-specific IP address pool and suffix in the **Native Users List** page (Figure 29 on page 123), the **Unix Users List** page (Figure 39 on page 136), or the **Profiles List** page (Figure 44 on page 146) by adding a **Framed-IP-Address** check list or return list attribute with corresponding IP address pool (Figure 68 on page 190).

Figure 68: Associating IP Address Pools with RADIUS Clients



If user **EDISON CARTER**, who has been assigned to **<RAS>-GOLD**, logs into RAS1, he receives an IP address from the **RAS1-GOLD** address pool. If he logs into RAS2, he receives an address from the **RAS2-GOLD** address pool. If, however, he logs into RAS3 but **RAS3-GOLD** has not been defined in the IP Pools dialog, he is not assigned an IP address.

To assign a **Framed-IPv6-Prefix** attribute, follow the preceding instructions for **Framed-IP-Address** and then add **Framed-IPv6-Prefix** to the reply list. The **Framed-IPv6-Prefix** value will be derived from the assigned address and the **Framed-IP-Address** suppressed. For more information on the use of **Framed-IPv6-Prefix**, see the *SBR Carrier Reference Guide*.

Specifying IP Address Assignment from a DHCP Server

IP addresses can be assigned from a back-end DHCP server, rather than from a standard IP address pool. DHCP address pools function like internal address pools—**Framed-IP-Address** can be allocated from any address pool, either internal or DHCP.

NOTE: SBR Carrier does not support DHCPv6.

DHCP address pools are defined in the **dhcp.ini** file and initialization files with the extension **.dhc**.

In addition, each DHCP address pool must be enabled by adding a placeholder IP address pool in the Web GUI. This placeholder pool should have the same name as the DHCP pool, and should have an empty list of address ranges. The placeholder pool allows the DHCP pool to appear in lists presented by the Web GUI, so it can be selected into an attribute.

When an IP address must be assigned from a DHCP pool during an Access-Request, DHCP DISCOVER and REQUEST messages are issued to trigger the allocation of an address. When an accounting Stop ends the session, DHCP RELEASE is issued to the server that allocated the address. Upon receipt of an accounting INTERIM request, a DHCP REQUEST message is issued to the server that allocated the address, attempting to extend the lease. If the server is specified as a broadcast address, DHCP *failover* occurs if the primary DHCP server goes down.

DHCP leases can be acquired, extended, and released by different servers. The server that acquires the lease adds all the information for extending and releasing the lease to the Class attribute.

Flexible configuration features allow RADIUS attributes to be mapped to DHCP options. Therefore, information from a RADIUS request can be provided to the DHCP server, and information returned from the DHCP server can be returned to the network access server.

During authentication, if an address is assigned from a pool, the pool name must refer to either a DHCP pool or an internal pool. If the pool name is not found, the request is rejected.

Address Allocation

During address allocation, a DISCOVER message is issued. If an OFFER is received from a DHCP server and the offered lease time meets the minimum lease time requirements, the server issues a REQUEST message. If an ACK message is received, the allocated address is returned in the Access-Accept.

In addition to the options required for normal DHCP operation, additional options in the DHCP DISCOVER and REQUEST messages are constructed based on the attributes in the RADIUS request and the literal values specified in the [Request] section for the pool. A Parameter Request List option is also constructed, listing all return options required for populating the RADIUS response, as specified in the [Reply] section for the pool.

If an address is assigned by means of DHCP, the **DH=** field is added to the Class attribute. This field includes:

- The unique client identifier for this lease.
- The address of the DHCP server.
- The lease time.

The unique client identifier for each user session is placed in the client hardware address field of the DHCP request as well as in the Client ID option. This information is used by the DHCP server to associate IP addresses with clients.

Address Renewal

If an INTERIM accounting message whose Class attribute includes both the **IP=** and the **DH=** fields is received, a REQUEST message is unicast to the DHCP server that allocated the address in an attempt to renew the lease. It requests the same lease time as was granted for the original request. If the server is specified as a broadcast address, DHCP *failover* occurs if the primary DHCP server goes down.

NOTE: If a renewal request is rejected, the DHCP server does not inform the network access server that the user's IP address is not renewed and may become invalid.

NOTE: If you want INTERIM accounting requests to keep the sessions from being considered stale or removed, you must enable the **UpdateOnInterim** parameter in the **radius.ini** file by setting its value to 1.

Address Release

If an accounting Stop message whose Class attribute includes both the **IP=** and the **DH=** fields is received, a RELEASE message is unicast to the DHCP server that allocated the address.

NOTE: The DHCP server does not reply to the RELEASE message.

An address to the DHCP server is also released when a session is deleted from its session database for reasons other than receiving an accounting Stop. For example, phantom session expiration or administrative deletion of a session result in the release of the temporary DHCP address.

DHCP Option Mapping

Options in a DHCP DISCOVER or REQUEST message can automatically be constructed based on attributes in the RADIUS request as well as pre-configured literal values. Also, options returned by the DHCP server in an OFFER message can be transmitted back to the network access server in RADIUS attributes.

The following applies to the mapping between RADIUS attributes and DHCP options:

- Both standard and vendor-specific DHCP options are supported. (Vendor-specific DHCP options must use standard encapsulation rules, as described in RFC 2132.)
- Format conversions between RADIUS attributes and a DHCP option are performed. For example, a DHCP option containing an IP address is formatted into dotted notation when returned in a RADIUS string attribute.
- A single RADIUS request attribute can set more than one DHCP option in a request, and a single DHCP option can set more than one RADIUS response attribute.
- A single DHCP option containing multiple values can be mapped to multiple instances of a single RADIUS attribute.

For example, a RADIUS attribute called **IP-Router** could appear multiple times in an Access-Accept. DHCP's Router option returns a list of IP addresses of routers. This single DHCP option can be configured to return multiple instances of the RADIUS IP-Router attribute -- one for each router address in the list.

- A single DHCP option containing multiple values can be mapped to multiple RADIUS attributes.

For example, two RADIUS attributes exist, **Primary-DNS-Server** and **Secondary-DNS-Server**. DHCP's **DNS Server** option returns a list of IP addresses of DNS servers. This single DHCP option can be configured to set the first DNS server address in Primary-DNS-Server and the second in Secondary-DNS-Server.

- Only attributes appropriate to the dictionary are returned.

Therefore, if network access servers from different vendors use different RADIUS attributes for the same information, each RADIUS attribute that might be required can be mapped to the same DHCP option. The correct attribute is returned to the network access server.

Using Multiple Servers

As the information required to renew or release a DHCP-assigned address is contained in the Class attribute, it is feasible to set up multiple servers, all utilizing a common DHCP server for address allocation. The network access server can issue requests to any of the servers, and addresses are assigned and released correctly even if different servers handle authentication and accounting requests for the same session.

This architecture requires that each server must be configured to be stateless—that is, the current sessions database must be turned off in the **radius.ini** file, as follows:

```
[CurrentSessions]
Enable = 0
```

For a standalone server, current sessions processing makes sense only when authentication and all accounting transactions are directed to the same server. If current sessions processing is not disabled, the current sessions database will be incorrect and always grows. For example, DHCP addresses are prematurely released when phantoms expire.

Setting Up Administrator Accounts

IN THIS CHAPTER

- Local Administrator or Group Overview | 195
- Adding a Local Administrator or Group | 196
- Deleting a Local Administrator or Group | 197
- Administrator Configuration Files | 198

This chapter describes how to set up Steel-Belted Radius Carrier administrators and specify what permissions an administrator holds. This chapter contains these topics:

Local Administrator or Group Overview

You can grant or revoke the right to use the Web GUI to configure a SBR Carrier server. Each time you log in to a SBR Carrier server, Web GUI prompts you to authenticate yourself by entering a native OS user account and password.

During initial configuration of SBR Carrier, an initial SBR administrator account is selected. The default account is the user who executes the **configure** script, typically the root user. The account has the right to use the Web GUI at its default (full) level of access. Although a UNIX group can be assigned administrator privileges that extend to all group members, there is no default group.

TIP: Be selective when adding administrator accounts. More than one SBR Carrier administrator software account can be active at any one time, to support multiple users. But you cannot prevent simultaneous access to the same screen and settings, so one administrator can change or undo another administrator's work.

Adding a Local Administrator or Group

To add a local administrator or a group to the SBR Carrier database using the Web GUI:

1. Select **RADIUS Configuration > Administrators**.

The **Administrators List** page ([Figure 69 on page 196](#)) appears.

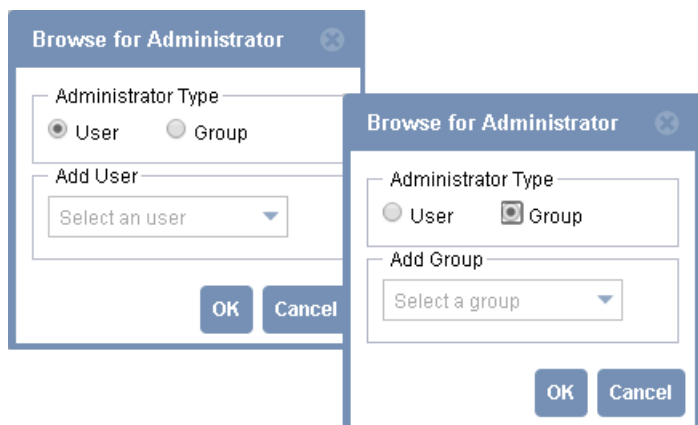
Figure 69: Administrators List Page



2. Click **Add**.

The **Browse for Administrator** dialog box ([Figure 70 on page 197](#)) appears.

Figure 70: Browse for Administrator Dialog



3. Identify the users or groups you want to add.

- If you want to add a user, select the **User** option button and select the name of a user from the drop-down list.
- If you want to add a group, select the **Group** option button and select the name of a user group from the drop-down list.

4. Click **OK**.

The **Administrators List** page ([Figure 69 on page 196](#)) displays an updated list of local administrator or group entries.

NOTE: Adding a user as an administrator by using the Web GUI overrides any access settings specified for the user in the **access.ini** configuration file.

Deleting a Local Administrator or Group

To revoke rights for a local administrator or group using the Web GUI:

1. Select **RADIUS Configuration > Administrators**.

The **Administrators List** page ([Figure 69 on page 196](#)) appears.

2. Select the local administrator or group entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

- 4. Click **Yes** to confirm the delete request.

The **Administrators List** page ([Figure 69 on page 196](#)) displays an updated list of local administrator or group entries.



CAUTION: Be careful not to revoke your own rights. If you do so, you will no longer have access to SBR Carrier administrative functions.

Administrator Configuration Files

Two files (see [Table 28 on page 198](#)) control administrative permissions. For more information about these files, refer to the *SBR Carrier Reference Guide*.

Table 28: Administrator Files

File Name	Function
access.ini	Maps operating system user or group account levels to administrative permissions.
admin.ini	Maps administrative access levels to sets of access rights.

Configuring Realm Support

IN THIS CHAPTER

- Realm Configuration Files | 199
- Stage One of Realm Configuration | 200
- Configuring a Proxy RADIUS Realm | 201
- Configuring a Directed Realm | 207
- Editing the proxy.ini File | 211
- Setting Up Smart Static Accounting | 212
- Setting Up Proxy RADIUS Realms | 213
- Setting Up Directed Realms | 214
- How to Update Realm Configuration | 215

This chapter describes how to configure proxy and directed realms in SBR Carrier. This chapter contains these topics:

A realm is a collection of authentication methods that SBR Carrier invokes to process a RADIUS request. When an authentication request is received, SBR Carrier uses the username, selected RADIUS attributes, or other properties of the request to determine which realm handles the request. The selected realm can be a proxy realm, a directed realm, or the default realm (if no explicit realm is selected).

Realm Configuration Files

The files in [Table 29 on page 200](#) establish settings for setting up users. For more information about these files, refer to the *SBR Carrier Reference Guide*.

Table 29: Realm Files

File Name	Function
filter.ini	<p>Stores filters for RADIUS attributes. Filters might be referenced from the [Auth] or [Acct] section of a RealmName.pro or RealmName.dir file.</p> <p>NOTE: You use the Web GUI to configure the filter.ini file. Do not edit the filter.ini file manually.</p>
proxy.ini	Specifies the order of realm selection methods, the realm selection rules, and other settings for all realms on the server.
proxyrl.ini	Configures settings for smart static accounting, which lets you specify that accounting packets for a proxy or directed realm should be forwarded to one or more proxy realms.
radius.ini	Enables proxy support and specifies (among other things) the realm names that indicate a server should handle a request and whether realm names and other username decorations should be stripped.
RealmName.dir	<p>Specifies settings for each directed or accounting realm you set up, where RealmName is the realm name.</p> <p>NOTE: RealmName must be listed in the [Directed] section of the proxy.ini file.</p>
RealmName.pro	<p>Specifies settings for each proxy realm you set up, where RealmName is the realm name.</p> <p>NOTE: RealmName must be listed in the [Realms] section of the proxy.ini file.</p>

Stage One of Realm Configuration

Perform the following steps to configure a realm of any type for SBR Carrier. Some steps require that you edit parameters in SBR Carrier configuration files.

1. Determine whether you want to provide proxy RADIUS or directed realms.

Does the customer have their own RADIUS servers, to which they want to direct requests? If so, you need to set up a proxy RADIUS realm for the customer.

Does the customer need you to host its RADIUS server? If so, you need to set up a directed authentication or accounting realm for the customer.

2. If you have not done so already, define delimiter conventions for realm name parsing.

The delimiter conventions that you define in **proxy.ini** are used for all realms defined in SBR Carrier. Be sure to inform the customer of the delimiter and prefix/suffix conventions you are using for realms.

```
proxy.ini
[Configuration]
RealmSuffix=
RealmPrefix=
```

3. Agree upon a realm name (or DNIS grouping) with the customer.

Keep the realm name simple if the realm is not defined by DNIS, because end users must enter it in combination with their existing usernames (for example, **User@RealmName**). The realm name configured in SBR Carrier does not need to match any names in use at the customer site. The realm name cannot match an existing target name, realm name, or tunnel name in your SBR Carrier configuration.

If the realm is defined as a DNIS grouping, the user is matched to a realm based on the Called-Station-Id.

Configuring a Proxy RADIUS Realm

A proxy RADIUS server treats a *realm* as a destination against which it performs authentication and accounting. Proxy realms are configurable only through configuration files.

[Table 30 on page 201](#) outlines the process of configuring a new proxy RADIUS realm for SBR Carrier. [Table 30 on page 201](#) also lists the sections that you must edit in configuration files to accomplish each step. You must perform each step in the process unless it is labeled as optional.

Table 30: Proxy Realm Configuration

Step	Proxy RADIUS Configuration Task	File and Section
1	Complete the preparatory steps outlined in the “Stage One of Realm Configuration” on page 200 section.	—

Table 30: Proxy Realm Configuration (continued)

Step	Proxy RADIUS Configuration Task	File and Section
2	Register the realm name with SBR Carrier. Optionally, you can use wildcards to specify matching rules for realms, and you can specify the default realm for undecorated User-Name attributes.	proxy.ini [Realms] Realm1 Realm2 = *.msn.com Realm3 = <undecorated>
3	Create a realm configuration file for each realm you register.	Realm1.pro Realm2.pro Realm3.pro
4	<p>Study the customer's current (or planned) RADIUS configuration. The customer's RADIUS servers are the target servers in the new realm.</p> <ul style="list-style-type: none"> • Are authentication and accounting packets directed to different RADIUS servers? • What is their need for a fast-fail policy, primary-secondary server strategy, or round-robin load balancing? • Are some servers used for authentication and some for accounting? • What is the IP address of each RADIUS server? • What UDP port and shared secret does each server use for authentication or accounting? 	—
5	<p>Does the customer want its RADIUS servers to receive Accounting-On and Accounting-Off messages? If so, add the new realm to your static proxy accounting configuration.</p> <p>See "Static Proxy Accounting" on page 157.</p>	proxy.ini [StaticAcct] 7=name 8=name [name] realm=RealmName

Table 30: Proxy Realm Configuration (continued)

Step	Proxy RADIUS Configuration Task	File and Section
6	Use the Web GUI to create a proxy entry for each target in the new realm. For authentication targets, verify that the Make Available as an Authentication Method check box is cleared.	Create Proxy Target pane in Web GUI Selected Target pane in Web GUI
7	Give the customer the IP address of the SBR Carrier server as well as the UDP port and shared secret it uses for authentication and accounting. Instruct the customer that for each target in the new realm, the SBR Carrier server must be added to the target's database as a RADIUS client. Presumably, someone at the customer site performs this task by running the target server's RADIUS configuration utility.	—
8	Enable authentication in this realm.	<i>RealmName.pro</i> [Auth] Enable=1
9	(Optional) Indicate that any realm names and delimiters are to be stripped from the User-Name before it is sent to the target server for authentication. <ul style="list-style-type: none"> • A value of 0 indicates realm names should not be stripped. • A value of 1 indicates realm names should be stripped. 	StripRealm=

Table 30: Proxy Realm Configuration (continued)

Step	Proxy RADIUS Configuration Task	File and Section
10	<p>Specify which target servers receive authentication packets. Configure load balancing and other details of realm and target selection for authentication packets.</p> <p>This is a multi-step process: (1) In the [Auth] section of the RealmName.pro file, set Enable to 1 and assign a name to the TargetsSection parameter; (2) create a [name] section in the file; and (3) within this section list the targets for authentication. When listing a target, use the name you assigned to it in the Proxy dialog.</p>	TargetsSection=name . . . [name] Server=
11	<p>(Optional) Specify an attribute filter to apply to authentication requests going out to the realm from SBR Carrier.</p> <p>This is a multi-step process: (1) In the [Auth] section of RealmName.pro, assign a name to the FilterOut parameter; (2) Create a [name] section in the filter.ini file; and (3) In the filter.ini [name] section, list the rules for editing the attributes in a RADIUS authentication request packet before forwarding the packet out to a proxy RADIUS realm.</p>	RealmName.pro [Auth] FilterOut=name filter.ini [name] . . .
12	<p>(Optional) Specify an attribute filter to apply to authentication responses returning into SBR Carrier from the realm.</p> <p>This is a multi-step process: (1) In the [Auth] section of RealmName.pro, assign a name to the FilterIn parameter; (2) create a [name] section in the filter.ini file; and (3) within the filter.ini [name] section list the rules for editing the attributes in an authentication response packet as it returns in from the proxy RADIUS realm, before relaying the packet back to the RADIUS client.</p>	RealmName.pro [Auth] FilterIn=name filter.ini [name] . . .

Table 30: Proxy Realm Configuration (continued)

Step	Proxy RADIUS Configuration Task	File and Section
13	Enable proxy RADIUS accounting in this realm.	<i>RealmName.pro</i> [Acct] Enable=1
14	(Optional) Indicate that any realm names and delimiters are to be stripped from the User-Name before it is sent to the target server for accounting. <ul style="list-style-type: none"> • A value of 0 indicates realm names should not be stripped. • A value of 1 indicates realm names should be stripped. 	StripRealm=1
15	(Optional) Indicate that accounting attributes should be logged locally on the SBR Carrier server as well as being directed to the realm. <ul style="list-style-type: none"> • A value of 0 indicates accounting attributes should not be logged locally. • A value of 1 indicates accounting attributes should be logged locally. 	RecordLocally=1
16	Specify which target servers receive accounting packets. Configure load balancing and other details of realm and target selection for accounting packets. This is a multi-step process: (1) In the [Acct] section of the <i>RealmName.pro</i> file, set Enable to 1 and assign a name to the TargetsSection parameter; (2) create a [name] section in the file; and (3) within this section list the targets for accounting. When listing a target, use the name you assigned to it in the Proxy dialog.	TargetsSection=name . . . [name] Server=

Table 30: Proxy Realm Configuration (continued)

Step	Proxy RADIUS Configuration Task	File and Section
17	(Optional) Specify an attribute filter to apply to accounting requests going out to the realm from SBR Carrier. This is a multi-step process: (1) In the [Acct] section of RealmName.pro , assign a name to the FilterOut parameter; (2) create a [name] section in the filter.ini file; and (3) within the filter.ini [name] section list the rules for editing the attributes in a RADIUS accounting request packet before forwarding the packet out to a proxy RADIUS realm.	RealmName.pro [Acct] FilterOut=name filter.ini [name] . . .
18	(Optional) Specify an attribute filter to apply to accounting responses returning into SBR Carrier from the realm. This is a multi-step process: (1) In the [Acct] section of RealmName.pro , assign a name to the FilterIn parameter; (2) create a [name] section in the filter.ini file; and (3) within the filter.ini [name] section list the rules for editing the attributes in an accounting response packet as it returns in from the proxy RADIUS realm, before relaying the packet back to the RADIUS client.	RealmName.pro [Acct] FilterIn=name filter.ini [name] . . .
19	(Optional) Provide DNIS information for this realm.	RealmName.pro [Called-Station-ID]
20	(Optional) Specify a proxy fast-fail policy for the realm.	[FastFail]

Table 30: Proxy Realm Configuration (continued)

Step	Proxy RADIUS Configuration Task	File and Section
21	(Optional) Enable SBR Carrier to map the presence or absence of certain attributes or values to this realm.	proxy.ini [AuthAttributeMap] <i>RealmName</i> [AcctAttributeMap] <i>RealmName</i>
22	<p>It is possible to load your new realm configuration dynamically, without stopping and restarting the server by issuing the SIGHUP (1) signal to the SBR Carrier process:</p> <pre>#./sbrd hup</pre> <p>SBR Carrier re-reads proxy.ini, filter.ini, and all .pro files in the server directory, and resets its realm configuration accordingly.</p> <p>NOTE: Rarely, you must edit radius.ini while configuring a realm. If you do edit radius.ini, you must stop and restart SBR Carrier before your new configuration is fully loaded.</p>	—

Configuring a Directed Realm

Table 31 on page 207 traces the process of configuring a directed authentication or accounting realm for SBR Carrier. Directed realms are configurable only through the configuration files and not through the GUI. Table 31 on page 207 also lists the sections that you must edit in SBR Carrier configuration files to accomplish each step. You must perform each step in the process unless it is labeled as optional.

Table 31: Configuring a Directed Realm

Step	Directed Realm Configuration Task	File and Section
1	Complete the steps outlined in “Stage One of Realm Configuration” on page 200 .	—

Table 31: Configuring a Directed Realm (continued)

Step	Directed Realm Configuration Task	File and Section
2	Register the RealmName with SBR Carrier. Optionally, you can use wildcards to specify matching rules for realms, and you can specify the default realm for undecorated User-Name attributes.	proxy.ini [Directed] Realm1 Realm2 = *.msn.com Realm3 = <undecorated>
3	Create a realm configuration file.	RealmName.dir
4	<p>Add the customer's user data to your database, which might be an external database (SQL, LDAP) or the SBR Carrier database.</p> <p>For information about how to add a limited number of users, see "Administering Users" on page 120.</p> <p>For information about adding users in batches, see "Importing and Exporting Data" on page 981. See also "Using the LDAP Configuration Interface" on page 497.</p>	—
5	<p>Configure the authentication method in SBR Carrier.</p> <p>See "Setting Up EAP Methods" on page 253. See also "Configuring SQL Authentication" on page 442 and "Configuring LDAP Authentication" on page 471.</p>	—
6	Register the authentication method with the realm.	RealmName.dir [AuthMethods]
7	Enable directed authentication in the realm.	[Auth] Enable=1
8	(Optional) Specify the name of the server certificate that must be used for EAP requests received from the directed realm.	ServerCertificate=

Table 31: Configuring a Directed Realm (continued)

Step	Directed Realm Configuration Task	File and Section
9	<p>(Optional) Indicate that any realm names and delimiters are to be stripped from the User-Name before authentication is performed.</p> <ul style="list-style-type: none"> • A value of 0 indicates realm names should not be stripped. • A value of 1 indicates realm names should be stripped. 	StripRealm=
10	Understand the data that the customer uses (or plans to use) to store accounting and billing records. This indicates the accounting methods to use.	—
11	<p>Configure the accounting method(s) in SBR Carrier.</p> <p>For more information, refer to the proxy.ini file in the <i>SBR Carrier Reference Guide</i>.</p>	
11a	<p>You can set up unique accounting log files by copying account.ini from the server directory to another directory, renaming it (if desired, but keep the .ini extension), and editing it to record accounting attributes by each customer. Use account.ini file syntax.</p> <p>For more information, refer to the account.ini file in <i>SBR Carrier Reference Guide</i>.</p>	.ini files
11b	<p>You can log to external SQL databases by copying an .acc file from the server directory to another directory, renaming it (if desired, but keep the .acc extension), and editing it to record accounting attributes by each customer. Use .acc file syntax.</p> <p>See "SQL Accounting Overview" on page 458.</p>	.acc files
12	Name each accounting method.	proxy.ini [DirectedAcct Methods]
13	Register the accounting method with the realm.	RealmName.dir [AcctMethods]

Table 31: Configuring a Directed Realm *(continued)*

Step	Directed Realm Configuration Task	File and Section
14	Enable directed accounting in the realm.	[Acct] Enable=1
15	(Optional) Indicate that any realm names and delimiters are to be stripped from the User-Name before accounting is performed. <ul style="list-style-type: none"> • A value of 0 indicates realm names should not be stripped. • A value of 1 indicates realm names should be stripped. 	StripRealm=
16	(Optional) Indicate that accounting attributes should be logged locally on the SBR Carrier server as well as being directed to the realm. <ul style="list-style-type: none"> • A value of 0 indicates accounting attributes should not be logged locally. • A value of 1 indicates accounting attributes should be logged locally. 	RecordLocally=
17	(Optional) Provide DNIS information for this realm.	[Called-Station-ID]
18	Load your new configuration. If you have added or changed any directed accounting methods, you must stop and restart the server. If you added or changed directed authentication methods in which external database (SQL or LDAP) authentication is used, you must stop and restart the server.	

Table 31: Configuring a Directed Realm (*continued*)

Step	Directed Realm Configuration Task	File and Section
19	<p>If you have added or changed directed authentication methods in which local or pass-through (Local, UNIX, Domain, Host) authentication is used, it is possible to load your new realm configuration dynamically, without stopping and restarting the server.</p> <p>Issue the SIGHUP (1) signal to the SBR Carrier process:</p> <pre>#./sbrd hup</pre> <p>SBR Carrier re-reads proxy.ini and all .dir files in the server directory, and resets its realm configuration accordingly.</p> <p>NOTE: Rarely, you must edit radius.ini while configuring a realm. If you do edit radius.ini, you must stop and restart the Radius Carrier before your new configuration is fully loaded.</p>	

Editing the proxy.ini File

The **proxy.ini** file specifies the order of realm selection methods, the realm selection rules, and other settings for all realms on the server. Settings for a realm are provided in its **RealmName.pro** or **RealmName.dir** file.

After you edit **proxy.ini**, you must apply your changes as follows:

- If you have configured any proxy RADIUS realms, it is possible to load your new realm configuration dynamically, without stopping and restarting the server.
- Issue the SIGHUP (1) signal to the SBR Carrier process:

```
#./sbrd hup
```

SBR Carrier re-reads **proxy.ini**, **filter.ini**, and all ***.pro** and ***.dir** files in the server directory, and resets its realm configuration accordingly.

- If you have configured any directed realms and if you have added or changed:

- Any directed accounting methods at all, you must stop and restart the server to load your new configuration.
- Directed authentication methods in which external database (SQL or LDAP) authentication is used, you must stop and restart the server to load your new configuration.
- Directed authentication methods in which local or pass-through (Local, UNIX, Domain, or Host) authentication is used, you can load your new realm configuration by using a SIGHUP (1) signal.

NOTE: Refer to the *SBR Carrier Reference Guide* for syntax information for the **proxy.ini** and **proxyrl.ini** files.

NOTE: If you edit **radius.ini** while configuring a realm, you must stop and restart the server before your new configuration is fully loaded.

Setting Up Smart Static Accounting

The **proxyrl.ini** file supports a feature called *smart static accounting*, which allows you to specify that the accounting packets for a proxy or directed realm should be forwarded to a list of one or more proxy or directed realms. These groups of realms can also be used for static accounting configured in **proxy.ini**.

This file consists of a number of sections that you name. Each section name is referenced in the **StaticAcctRealms** field in the [Acct] section of a **.pro** or **.dir** file. Following the section name, you can list a number of proxy realm names, in the following format:

```
[realm-list-name-1]
proxy-realm-1
proxy-realm-2
.
.
.
[realm-list-name-2]
.
.
.
```

For example:

```
[StaticAcctTargets1]
AcctSvr1
AcctSvr4
```

NOTE: To avoid an infinite loop, the list of static accounting servers must not include realms that use the list. If you include a realm in a list of static accounting servers and specified the same realm in **proxy.ini** as doing static accounting, the realm receives duplicate accounting packets.

NOTE: When you configure the **.pro** file, you must ensure at least one system is referenced in the TargetsSection.

Setting Up Proxy RADIUS Realms

For each proxy RADIUS realm that you want to configure in SBR Carrier, you must create a file called **RealmName.pro**, where **RealmName** is the name of the realm, and you must add this RealmName to the [Realms] section of the **proxy.ini** file.

If you create or edit a **RealmName.pro** file, you can apply your configuration changes dynamically, without stopping the server:

- Issue the SIGHUP (1) signal to the SBR Carrier process:

```
#./sbrd hup
```

After you do this, SBR Carrier re-reads **proxy.ini**, **filter.ini**, and all **.pro** and **.dir** files in the server directory, and resets its realm configuration accordingly.

NOTE: Rarely, you must edit **radius.ini** while configuring a realm. If you do edit **radius.ini**, you must stop and restart SBR Carrier before your new configuration is fully loaded.

Refer to the *SBR Carrier Reference Guide* for field information for a ***.pro** file.

Configuration Tasks

To set up a **Realmname.pro** file:

1. Specify proxy RADIUS target selection rules.

Each **[name]** section of a **RealmName.pro** file specifies a set of rules that SBR Carrier can use to select a target for proxy-forwarding within the proxy RADIUS realm. Each **[name]** section consists of a list of target servers. For any particular request, if the first listed server fails to respond (or is presumed down), then the other servers are tried in the order listed. A **[name]** section is activated by referencing it from the **[Auth]** or **[Acct]** sections.

2. Optionally, configure round-robin load balancing.

If you have multiple target servers in a realm, you can select whether to use them in round-robin fashion (load balancing), primary/backup fashion, or a combination of both. The value of the **RoundRobin** entry in the **[Auth]** or **[Acct]** section indicates the number of targets that are to be used in round-robin fashion.

Refer to the *SBR Carrier Reference Guide* for information about configuring round-robin options.

3. Configure proxy RADIUS fast-fail options.

You can use the **[FastFail]** section of a realm configuration file to fine-tune retry policies for individual realms, and for specific targets within a realm. If you provide a **[FastFail]** section, the **ProxyFastFail** parameter in the **radius.ini** **[Configuration]** section is ignored.

4. Specify username decoration options.

You can use the **[ModifyUser]** section of a realm configuration file to decorate a realm, where the realm is determined by other means, such as DNIS or attribute mapping. For example, if **george@gm** and **george@ford** are both in the RADIUS database, either user could log in as **george**, as SBR Carrier would determine the realm, for example, by DNIS. Based on the realm, SBR Carrier would append either **@gm** or **@ford** to the username, and then use the Local User directed method to authenticate.

Setting Up Directed Realms

A *directed realm* specifies target methods for directed authentication or directed accounting. Its realm configuration file is called **RealmName.dir**.

The *directed authentication* feature permits the server to bypass its authentication methods list and map an incoming RADIUS request to one or more specific authentication methods. SBR Carrier chooses the destination method based on routing information found in the request packet. The destination methods might be any authentication methods already configured on the local SBR Carrier server, regardless of how they were configured; for example, a method might have been configured using the Web GUI pages, the LDAP configuration interface, or an **.aut** configuration file.

If no directed authentication method is configured, every request percolates through the same authentication methods list, as defined in the **Authentication Methods** page in the Web GUI. Directed realms can also use proxy realms as an authentication method. Directed authentication allows you to tailor an authentication methods list to a customer's needs.

Directed accounting is also possible. The destination accounting method might be the SBR Carrier accounting log, an external database configured using an **.acc** file, or a distinct accounting log file that contains entries only for this customer.

To activate these features, you must create **RealmName.dir** files, place them in the SBR Carrier directory, and list them in the [Directed] section of **proxy.ini**. Subsequently, any requests that arrive addressed to one of these realm names are processed on the local server using the instructions you have provided in **proxy.ini** and in the corresponding **RealmName.dir** file.

After you edit a **RealmName.dir** file, you must apply your changes as follows. If you have added or changed:

- Any directed accounting methods at all, you must stop and restart the server to load your new configuration.
- Directed authentication methods in which external database (SQL or LDAP) authentication is used, you must stop and restart the server to load your new configuration.
- Directed authentication methods in which local or pass-through (Local, UNIX, Domain, or Host) authentication is used, you can apply your configuration changes dynamically, without stopping the server:
 - Issue the SIGHUP (1) signal to the SBR Carrier process:

```
#./sbrd hup
```

SBR Carrier re-reads **proxy.ini**, **filter.ini**, and all **.pro** and **.dir** files in the server directory, and resets its realm configuration accordingly.

NOTE: Rarely, you must edit **radius.ini** while configuring a realm. If you edit **radius.ini**, you must stop and restart SBR Carrier before your new configuration is fully loaded.

How to Update Realm Configuration

The following information explains when a SIGHUP (1) signal is sufficient, or insufficient, for updating realm configuration:

- A SIGHUP (1) signal is sufficient to load any changes that you make to **proxy.ini**, **filter.ini**, or ***.pro** files for the purpose of configuring proxy RADIUS realms.
- However, when you configure directed realms (**proxy.ini**, ***.dir** files, and possibly ***.acc**, ***.aut**, and accounting ***.ini** files as well) you must load configuration changes as follows. If you have added or changed:
 - Any directed accounting methods at all, you must stop and restart the server.
 - Directed authentication methods in which external database (SQL or LDAP) authentication is used, you must stop and restart the server.
 - Directed authentication methods in which local or pass-through (Local, UNIX, Domain, or Host) authentication is used, a SIGHUP (1) signal is sufficient.

For information about stopping and starting the server see

[“When and How to Stop and Restart Steel-Belted Radius Carrier” on page 974.](#)

Setting Up Filters

IN THIS CHAPTER

- Overview | 217
- Adding a Filter | 224
- Searching the Filter List | 227
- Editing a Filter | 229
- Deleting a Filter | 231

This chapter describes how to create and use filters in SBR Carrier. This chapter contains these topics:

Overview

A filter is a collection of rules for adding, modifying, or removing attributes or attribute values in RADIUS requests and responses. You define filters and their rules using Web GUI. You enable filters by referring to them by name when using the Web GUI or when editing certain `.ini` file sections.

A filter consists of one or more rules, which are processed in sequential order.

- Add rules specify that an attribute-value pair (AVP) is added to a RADIUS packet during processing. The AVP is added after all other rules are processed. An attribute is added to a packet only if it is legal to do so.

Some attributes can appear only once in a RADIUS packet; others can appear multiple times. If an attribute that is the subject of an Add rule is already present in the packet (after processing Allow and Exclude rules) and the attribute can only appear once, the Add rule is not processed and the second instance of the attribute is not added.

- Allow rules specify whether an attribute or AVP is allowed in a RADIUS packet.
 - If an Allow rule specifies an attribute name and an attribute value, then only attributes of the specified type and value are allowed in the RADIUS packet.
 - If an Allow rule specifies an attribute name without an attribute value, then all attributes of the specified type, regardless of value, are allowed in the RADIUS packet.

- If an Allow rule does not specify an attribute name, then all attributes, regardless of value, are allowed in the RADIUS packet.
- The Allow Unknown rule specifies that all attributes, regardless of whether they are included in the dictionary of the sending NAS, are included when proxying the message to the target (outbound filters) or before returning the proxy response (inbound filters). Optionally, a Vendor Id may accompany the directive. When used with a global Exclude Unknown, this rule overrides the exclusion of attributes from the specified vendor ID.
- Exclude rules specify an attribute or AVP to be excluded from a RADIUS packet.
 - If an Exclude rule specifies an attribute name and an attribute value, then only attributes of the specified type and value are excluded from the RADIUS packet.
 - If an Exclude rule specifies an attribute name without an attribute value, then all attributes of the specified type, regardless of value, are excluded from the RADIUS packet.
 - If an Exclude rule does not specify an attribute name, then all attributes, regardless of value, are excluded from the RADIUS packet.
 - The Exclude Unknown rule specifies that all attributes that are not included in the dictionary of the sending NAD are deleted before proxying the message to the target (outbound filters) or before returning the proxy response (inbound filters). Optionally, a vendor ID may accompany the directive. If included, only attributes from the specified vendor are excluded.
- Replace rules specify the conditions whereby one attribute (or attribute value) is replaced with another.
 - If a Replace rule specifies replacement of one named attribute of a specified value (**attr1 v1**) to be replaced with a different attribute of a specified value (**attr2 v2**), then any occurrence of the first AVP is replaced with the second AVP. Result: **attr2 v2**.
 - If a Replace rule specifies replacement of a named attribute without a specified value (**attr1**) to be replaced with a different attribute of a specified value (**attr2 v2**), then any occurrence of the first attribute (regardless of value) is replaced with the second AVP. Result: **attr2 v2**.
 - If a Replace rule specifies replacement of one named attribute of a specified value (**attr1 v1**) to be replaced with a different attribute without a specified value (**attr2**), then any occurrence of the first attribute is replaced with the second attribute, which retains the value of the original attribute. Result: **attr2 v1**.
 - If a Replace rule specifies replacement of one named attribute (without a specified value) with a different attribute without a specified value, then any occurrence of the first attribute is replaced with the second attribute, which retains the value of the original attribute. Result: **attr2 v1**.

NOTE: You cannot replace a subattribute with a parent attribute, or vice versa.

- Script rules specify when to run attribute filter scripts. For information about attribute filter scripts, refer to [“Creating Attribute Filter Scripts” on page 908](#).

The SBR Carrier dictionary file **radius.dct** provides string aliases for certain integer values defined in the RADIUS standard. You can use these strings in attribute filter rules.

NOTE: Filter rules provide you with tremendous flexibility. However, SBR Carrier does not prevent you from creating an invalid RADIUS packet. Some attributes are not appropriate for certain types of requests. For example, adding a pooled Framed-Ip-Address attribute to an accounting request can cause a loss of available IP addresses.


NOTE: You can specify structured attributes in attribute filters. Throughout this chapter, the term *attributes* refers to both standard RADIUS attributes and structured attributes. For information about specifying structured attributes, see the *SBR Carrier Reference Guide*.

NOTE: You can use a separately licensed add-on module to use JavaScript to select and create filters. For more information about the JavaScript module, refer to [“Optional Scripting Module” on page 860](#).

Order of Filter Rules

The order of rules within a filter is important. General default rules that take no parameters, such as Allow (allow all attributes unless otherwise specified) or Exclude (exclude all attributes unless otherwise specified) must appear as the first rule in the filter. Later rules supersede earlier rules; the last applicable rule takes precedence. Add and Replace rules are applied after the Allow and Exclude rules.

More specific rules with more parameters (Add **attribute value**) act as exceptions to less specific rules with fewer parameters (Allow **attribute**, EXCLUDE). For example, you might want to allow a certain attribute and exclude one or more specific values for that attribute. Or you might exclude all attributes, allow specific attributes, and add specific attribute/value pairs.

 **NOTE:** Script rules are not subject to rule ordering.

You can use two basic approaches to designing a filter:

- Start the rule list with a default Exclude rule (no parameters) and add Allow rules for any attributes or attribute/value pairs that you want to insert into the packet. Add and Replace rules may be used.
- Start the rule list with a default Allow rule (no parameters) and add Exclude rules for any attributes or attribute/value pairs that you want to remove from the packet. Add and Replace rules may be used.

The default action for SBR Carrier is Exclude. If a filter does not contain any rules, the filter removes all attributes from a packet when the filter is applied.

Values in Filter Rules

The value of an attribute is interpreted based on the type of the attribute in its attribute dictionary.

[Table 32 on page 221](#) lists the meaning of each attribute type.

Table 32: Filter Rule Values

Attribute Type	Function
hexadecimal	A hexadecimal value is specified as a string. Special characters may be included using escape codes.
int1, int4, integer	<p>1- or 4-byte unsigned decimal number (integer is equivalent to int4).</p> <p>NOTE: The SBR Carrier dictionary file radius.dct provides string aliases for certain integer values defined in the RADIUS standard. You can use these strings in attribute filter rules.</p>
ipaddr, ipaddr-pool	<p>An IP address in dotted notation; for example:</p> <p>EXCLUDE NAS-IP-Address 127.0.0.1</p>

Table 32: Filter Rule Values (continued)

Attribute Type	Function
string	<p>String attribute (includes null terminator). A string is specified as text. The text may be enclosed in double-quotes ("). The text is interpreted as a regular expression. Back slash (\) is the escape character. Escape codes are interpreted as follows:</p> <p>Code Meaning</p> <p>\a 7</p> <p>\b 8</p> <p>\f 12</p> <p>\n 10</p> <p>\r 13</p> <p>\t 9</p> <p>\v 11</p> <p>\nnn nnn is a decimal value between 0 and 255</p> <p>\xnn nn is a hexadecimal value between 00 and FF</p> <p>\c c is a single character, interpreted literally</p> <p>Prefix literal backslashes (\) within a string and double-quotes (") within quoted strings with an escape character. For example:</p> <p>ADD Reply-Message Session limit is one hour</p> <p>ADD Reply-Message "Session limit is one hour"</p> <p>ADD Reply-Message "Your username is \"George\" \"George\""</p>

Table 32: Filter Rule Values (*continued*)

Attribute Type	Function
time	<p>A time value is specified with a string indicating date and time:</p> <p><code>yyyy/mm/dd hh:mm:ss</code></p> <p>The date portion is mandatory; the time portion may be specified to whatever degree of precision is required, or may be omitted entirely. For example:</p> <p><code>2009/01/20 14:00:00</code></p> <p>and</p> <p><code>2009/01/20 14</code></p> <p>both refer to January 20, 2009 at 2:00 p.m.</p> <p>For example:</p> <p>ADD Ascend-PW-Expiration 2009/01/20</p>

Referencing Attribute Filters

SBR Carrier attribute filtering provides flexibility in packet processing. You reference filters by name in Web GUI dialogs, in various **.ini** and **.aut** configuration files, and in the FilterOut and FilterIn sections of your **.pro** and **.dir** files. You can use the same filter for all packets in all realms. You can apply filtering to some realms, and not others.

To disable filtering for a realm, omit filtering parameters from the ***.pro** or ***.dir** files and from the EAP-PEAP/EAP-TTLS configurations. Filtering is often used only for packets that are routed out to realms (the FilterOut parameter).

To reference filtering rules in proxy or directed realm configurations, you must use the FilterOut and FilterIn parameters in the [Auth] and [Acct] sections of a realm configuration file. For more information, refer to the *SBR Carrier Reference Guide*.

NOTE: Do not allocate IP addresses from SBR Carrier IP address pools in accounting filters. These addresses are allocated but never released.

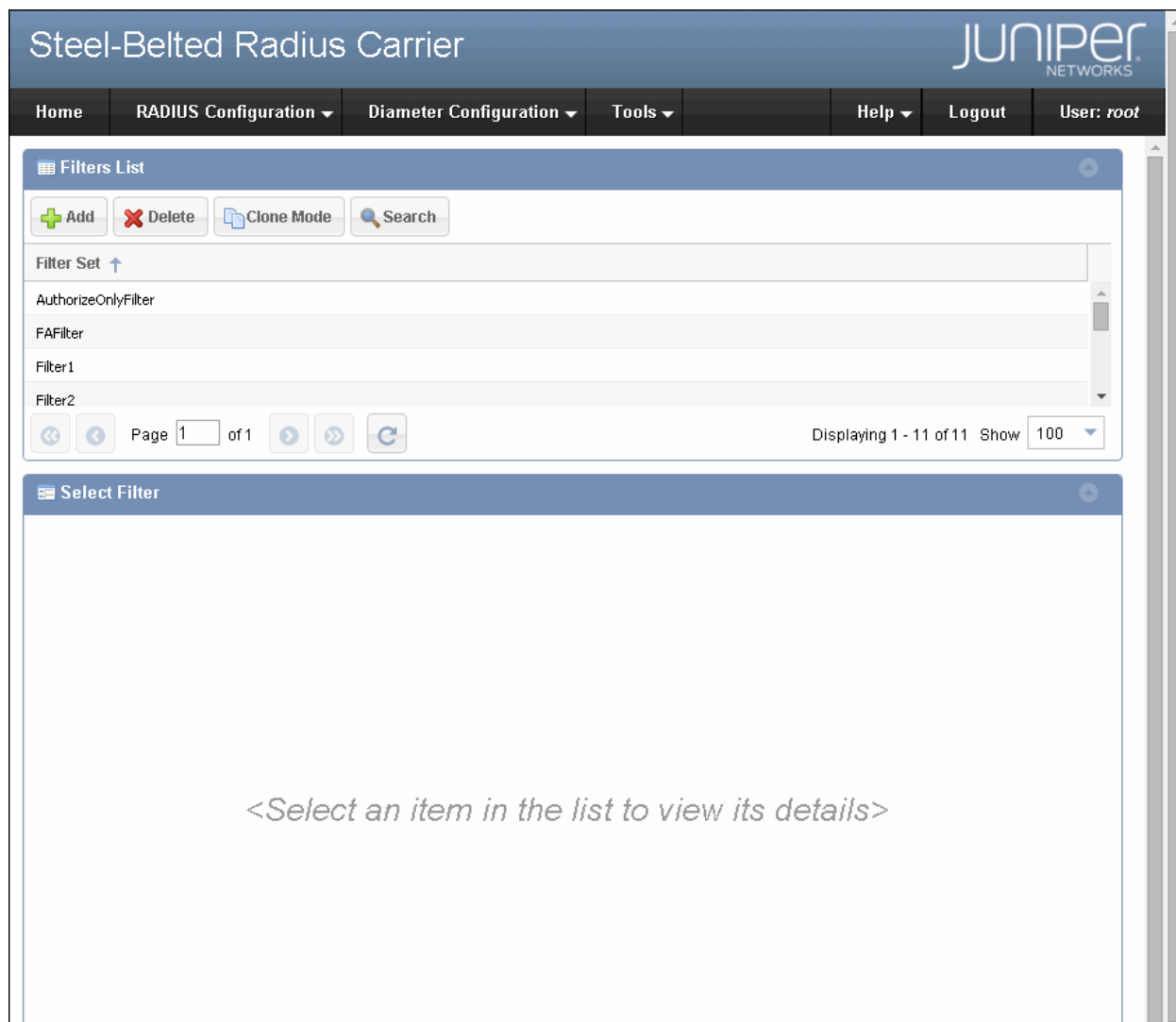
Adding a Filter

To add a filter using the Web GUI:

1. Select **RADIUS Configuration > Filters**.

The **Filters List** page ([Figure 71 on page 224](#)) appears.

Figure 71: Filters List Page



2. Click **Add**.

The **Create Filter** pane ([Figure 72 on page 225](#)) appears.

Figure 72: Create Filter Pane

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration ▼ Diameter Configuration ▼ Tools ▼ Help ▼ Logout User: root

Filters List

Create Filter

Name:

Description:

Default Rule: ☐ Allow ☒ Exclude

Rules

Rule Type	Attribute	Value	Replacemen...	Replacemen...	Script Name

Add Edit Delete

Save Clear Cancel

3. Enter a name for the filter in the **Name** field.
4. Optionally, enter a description for the filter in the **Description** field.
5. Select the **Allow** or **Exclude** option button to specify whether to allow or exclude attributes if no other rule applies to a RADIUS packet.
6. Click **Add** in the **Rules** area.

The **Add Rule** dialog box (Figure 73 on page 226) appears.

Figure 73: Add Rule Dialog

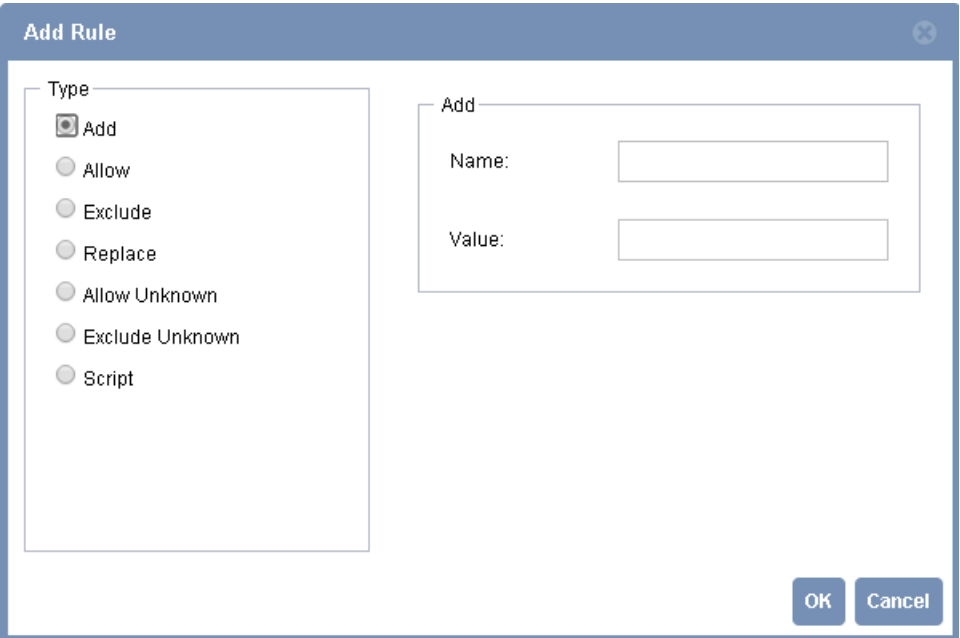


The 'Add Rule' dialog box features a title bar with a close button. Inside, a 'Type' section contains a list of radio buttons: 'Add', 'Allow', 'Exclude', 'Replace', 'Allow Unknown', 'Exclude Unknown', and 'Script'. The 'Add' option is currently selected. At the bottom right, there are 'OK' and 'Cancel' buttons.

7. Select the type of rule you want to add to the filter.

Options are **Add**, **Allow**, **Exclude**, **Replace**, **Allow Unknown**, **Exclude Unknown**, and **Script**. The fields in the **Add Rule** dialog box may change depending on the option you select. [Figure 74 on page 226](#) shows a sample **Add Rule** dialog box for the **Add** option.

Figure 74: Sample Add Rule Dialog



This version of the 'Add Rule' dialog box shows the 'Add' option selected in the 'Type' list. To the right of the list, a sub-section titled 'Add' contains two input fields: 'Name:' and 'Value:'. The 'OK' and 'Cancel' buttons remain at the bottom right.

8. Specify the attribute name and value settings you want to use for the rule.

NOTE: If you edit the filters for a SBR Carrier server, you can apply your configuration changes without stopping the server by issuing the SIGHUP (1) signal to the SBR Carrier process:

```
./sbrd hup
```

9. Click **OK**.

The **Rules** area in the **Create Filter** pane ([Figure 72 on page 225](#)) displays the updated lists of selected rules.

You can modify the rule list by using the **Edit** and **Delete** buttons. You can reorder the rules by selecting each rule and using the **Up** or **Down** arrows.

10. Repeat steps 6 through 9 to add more rules to the filter.

11. Click **Save** to save the filter configuration.

The **Filters List** page ([Figure 71 on page 224](#)) displays an updated list of filter entries.

Searching the Filter List

You can search your list of filters to identify those of a specific type or that use a specific rule.

To search your filter list using the Web GUI:

1. Select **RADIUS Configuration > Filters**.

The **Filters List** page ([Figure 71 on page 224](#)) appears.

2. Click **Search**.

The **Search Filters** dialog box ([Figure 75 on page 228](#)) appears.

Figure 75: Search Filters Dialog

Search Filters

Criteria

Rule Type:

Attribute Name:

Attribute Value:

Replacement Name:

Replacement Value:

Script:

Wildcard: *= any string, ? = any character, \ = literal

OK Close

3. Define one or more search criteria in the following fields:

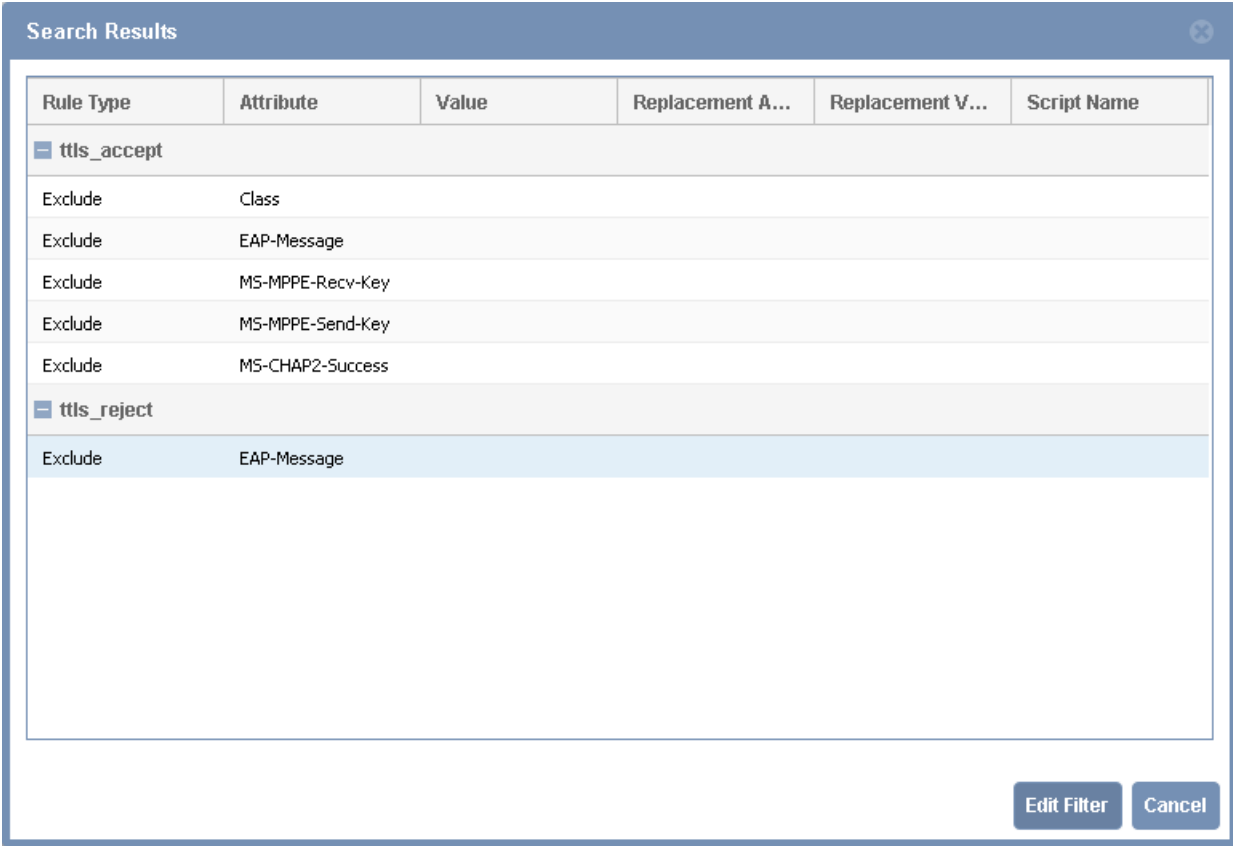
- Rule Type—Search filters based on the rule type.
- Attribute Name—Search filters based on the attribute name.
- Attribute Value—Search filters based on the attribute value.
- Replacement Name—Search filters based on the replacement attribute name.
- Replacement Value—Search filters based on the replacement attribute value.
- Script—Search filters based on the attribute filter script.

You can use the question mark (?) wildcard to represent one character in a string. You can use the asterisk (*) wildcard to represent any number of characters. For example, entering the search string *MS-MPPE-** identifies any filter that looks for the **MS-MPPE-Recv-Key** or **MS-MPPE-Send-Key** attribute.

4. Click **OK**.

The **Search Results** dialog box (Figure 76 on page 229) displays the list of filters that satisfy your search criteria.

Figure 76: Search Results Dialog



Editing a Filter

To edit a filter using the Web GUI:

1. Select **RADIUS Configuration > Filters**.

The **Filters List** page ([Figure 71 on page 224](#)) appears.

2. Select the filter entry that you want to edit.

The **Selected Filter** pane ([Figure 77 on page 230](#)) displays the settings configured for the filter.

Figure 77: Selected Filter Pane

The screenshot displays the Juniper Networks Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user (root). The main content area is divided into two sections. The top section, titled 'Filters List', contains buttons for Add, Delete, Clone Mode, and Search, along with a search input field. Below these is a list of filter sets: AuthorizeOnlyFilter, FAFilter, Filter1 (highlighted), and Filter2. A pagination bar at the bottom of this section shows 'Page 1 of 1' and 'Displaying 1 - 11 of 11'. The bottom section, titled 'Selected Filter: Filter1', contains fields for Name (Filter1), Description, and Default Rule (Allow/Exclude). Below these is a table for Rules with columns: Rule Type, Attribute, Value, Replacement Attribute, Replacement Value, and Script Name. The table is currently empty. At the bottom of the Rules section are buttons for Add, Edit, and Delete. At the very bottom of the Selected Filter pane are buttons for Save, Reset, and Cancel.

3. Edit the settings for the filter entry as appropriate.

For information about the fields in the **Selected Filter** pane, see [“Adding a Filter” on page 224](#).

NOTE: You cannot edit the name of the filter.

4. Click **Save** to save the changes.

The **Filters List** page ([Figure 71 on page 224](#)) displays an updated list of filter entries.

Deleting a Filter

To delete a filter using the Web GUI:

1. Select **RADIUS Configuration > Filters**.

The **Filters List** page ([Figure 71 on page 224](#)) appears.

2. Select the filter entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to confirm the delete request.

The **Filters List** page ([Figure 71 on page 224](#)) displays an updated list of filter entries.

Setting Up Authentication Policies

IN THIS CHAPTER

- [Authentication Policy Overview | 232](#)
- [Order of Authentication Methods | 233](#)
- [Adding EAP Methods to an Authentication Policy | 235](#)
- [Certificates | 238](#)
- [Trusted Root Certificates | 246](#)
- [Configuring a CRL Distribution Point Web Proxy | 248](#)
- [Configuring Authentication Rejection Messages | 250](#)
- [Configuring the Server | 252](#)

This chapter contains information about setting up authentication policies. This requires specifying how and when resources such as certificates, certificate authorities, and Extensible Authentication Protocol (EAP) methods and plug-ins are used to authenticate a user.

The following topics are included in this chapter:

Authentication Policy Overview

Authentication policies are linked to the type of user account being verified, or to the primary method of authentication. The default authentication methods for SBR Carrier include:

- A native user (listed in, and a user of, the SBR Carrier software)
- A UNIX account holder
- A member of an authorized UNIX group

You can set up other user types, or primary authentication methods, to support other authenticators, such as different forms of LDAP clients, SQL clients, SIMAuth, and others. The initial defaults for a particular system are configured during installation, by the configuration script. EAP authentication methods can be used in conjunction with these authentication methods to form the authentication policy. Configuration

of the EAP authentication methods supported by SBR Carrier is detailed in [“Setting Up EAP Methods” on page 253](#).

Order of Authentication Methods

When SBR Carrier receives an access request, it attempts to authenticate the request by sequentially trying its configured and enabled authentication methods. The order in which the server tries to authenticate the request is configured in the **Authentication Methods** page ([Figure 78 on page 234](#)) in Web GUI.

After you configure authentication methods for SBR Carrier, the **Authentication Methods** page displays them in the order in which the server tries them. The **Authentication Methods** page displays both active and inactive authentication methods. During an authentication transaction, the server works down the list of Active Authentication Methods. You can activate or deactivate methods, or reorder methods in the list by using the controls in the **Authentication Methods** page.

Any EAP methods you want included in an authentication policy must be configured and enabled before adding them to the authentication policy. See [“Setting Up EAP Methods” on page 253](#).

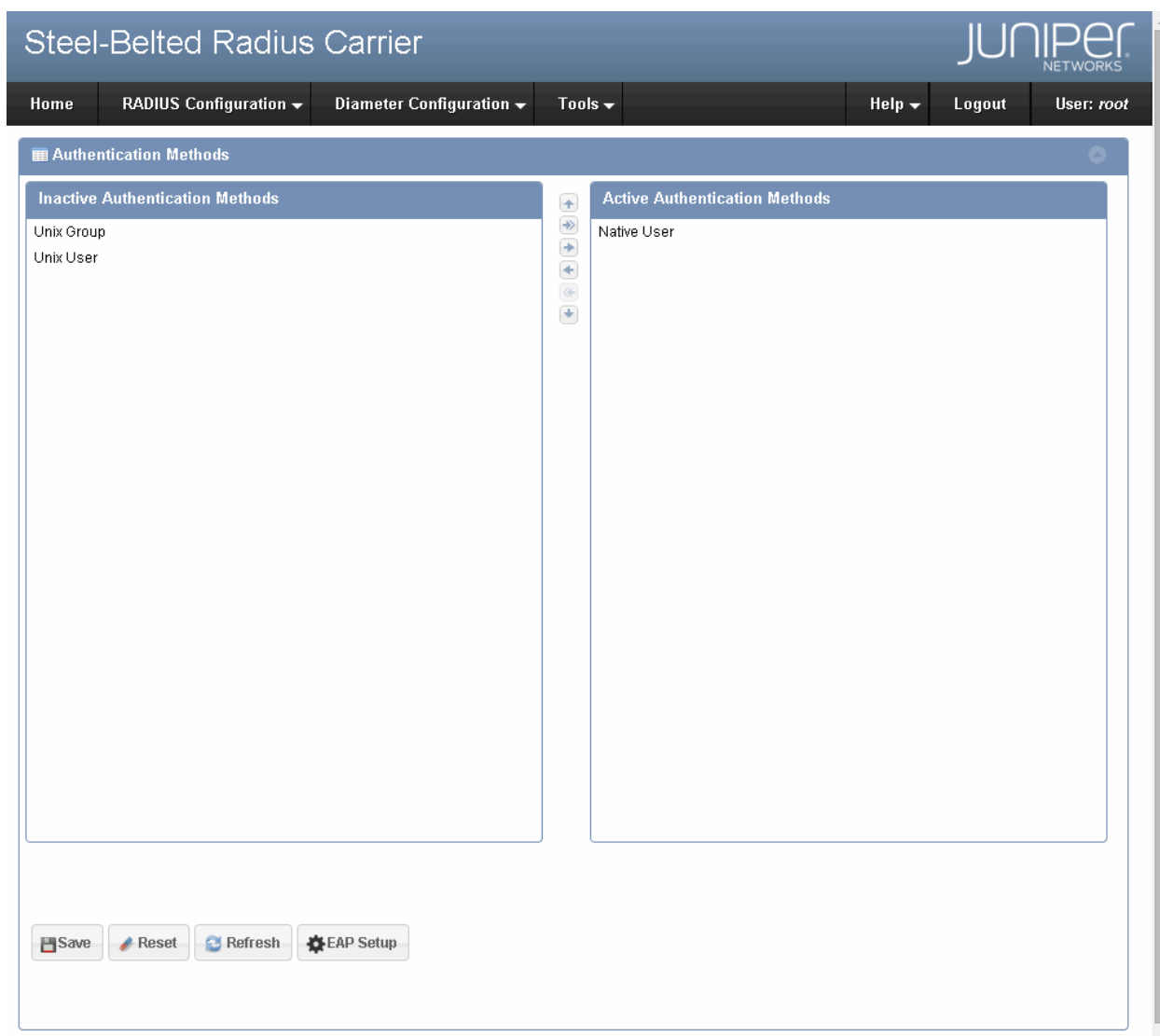
With the example configuration shown in [Figure 78 on page 234](#), the server attempts to process an Access-Request by trying the authentication methods in the following order:

1. LDAP
2. Native User
3. Unix User
4. Unix Group
5. EAP-TTLS

The default configuration for SBR Carrier does not include any EAP methods.

To view the current order that SBR Carrier is using to authenticate requests using the Web GUI, select **RADIUS Configuration > Authentication Policies > Order of Methods**. The **Authentication Methods** page ([Figure 78 on page 234](#)) appears. If an EAP method is configured and added to an authentication policy, it is listed under the **Active Authentication Methods** area.

Figure 78: Authentication Methods Page



- To activate or deactivate an authentication method, select the method and click the **Right** or **Left** arrow. To activate or deactivate all authentication method, click or to move all the methods to the corresponding area.
You can also activate or deactivate an authentication method by dragging the method to the right or left.
- To change the order in which the server tries authentication methods, select each authentication method and click the **Up** or **Down** arrow until the authentication methods are in the desired order.
- Click **Save** to save the changes. To revert to the previous settings, click **Reset**.

For more information about adding or deleting an authentication method, see the *SBR Carrier Reference Guide*.

NOTE: When you make changes to the **Authentication Methods** page, the SBR Carrier audit log does not record specific information about the performed actions.

Adding EAP Methods to an Authentication Policy

If you want your authentication policy to use one or more EAP methods along with an authentication method, you need to add the EAP methods to the authentication method.

NOTE: EAP methods must be configured before they can be added to an authentication policy. For details on configuring and enabling EAP methods see [“Setting Up EAP Methods” on page 253](#).

Enabling EAP Methods

To add an EAP method to an authentication policy, you must first enable the EAP method.

To enable an EAP method using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > EAP Methods**.

The **EAP Methods List** page ([Figure 79 on page 236](#)) appears.

Figure 79: EAP Methods List Page

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

EAP Methods List

Apply Reset Refresh

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Select EAP Method

<Select an item in the list to view its details>

2. Click the **Status** column of the EAP authentication method entry you want enable and select the appeared check box.
3. Click **Apply** to save the configuration.

Activating an EAP Method

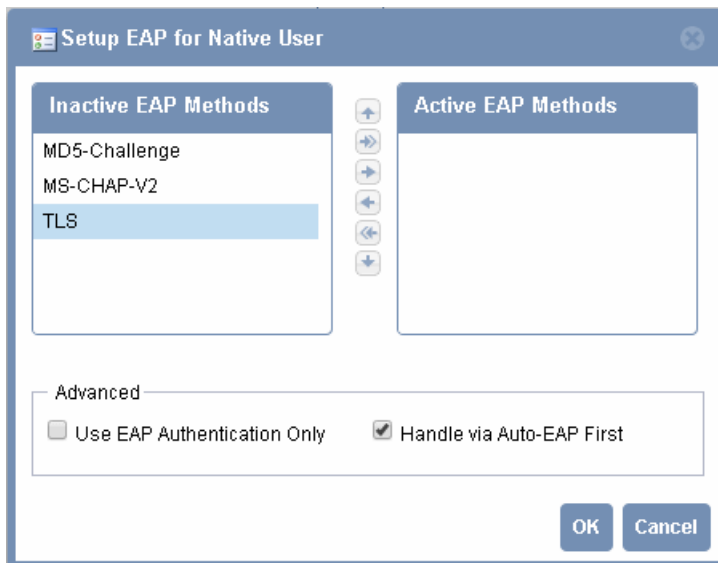
Once the EAP method has been enabled, it becomes available to be activated and added to the authentication policy. You can also specify the order in which you want SBR Carrier to use the EAP method for the authentication policy.

To activate and add an EAP method to an authentication policy using the Web GUI:



1. In the **Authentication Methods** page (Figure 78 on page 234), select the active authentication policy to which you want to add the EAP method and click **EAP Setup**.

The **Setup EAP for User** dialog box for the selected user appears. Figure 80 on page 237 shows the sample **Setup EAP for Native User** dialog box.

Figure 80: Setup EAP for Native User Dialog



NOTE: You must configure each EAP method using the respective **.aut** file before the EAP method is displayed in Web GUI.

2. To activate or deactivate an EAP method, select the method and click the **Right** or **Left** arrow to move the method to the corresponding area. To activate or deactivate all authentication method, click  or  to move all the methods to the corresponding area.

You can also activate or deactivate an authentication method by dragging the method to the right or left.

In the example shown in Figure 80 on page 237, the EAP-TLS method is being activated for the Native User authentication method.

3. If you select more than one EAP method, you can adjust the order of authentication. Select one of the methods and click the **Up** or **Down** arrow to change its order.
4. If you want to restrict use of this authentication method to requests that contain EAP credentials, select the **Use EAP Authentication Only** check box.

When this check box is selected, SBR Carrier prevents the authentication method from being called for any request that does not contain EAP credentials, and bypasses the authentication method if an

authentication request specifically requests an EAP protocol that is not listed in the authentication method's EAP-Type list in the **eap.ini** file. For more information see [“EAP-Only Setting” on page 256](#).

5. If you want SBR Carrier to use an automatic EAP helper to generate credentials for a user, select the **Handle via Auto-EAP First** check box.

You should clear this check box if an authentication method is capable of handling EAP credentials on its own (without an EAP helper). For more information, see [“First-Handle-Via-Auto-EAP Setting” on page 256](#).

6. Click **OK**.
7. Click **Save** to save the changes.

Certificates

A *certificate* is an electronic data structure used to identify an individual, a server, a company, or some other entity, and to associate that identity with a public key and an associated private key. Like a passport, a certificate provides generally recognized proof of an entity's identity. Certificates bind public key values to entities, so that remote users of an entity's public key can be certain the associated private key is owned by the correct person or system. Certificates help prevent the use of fake public keys for impersonation. Only the public key certified by the certificate works with the corresponding private key possessed by the entity identified by the certificate. The most widely accepted format for certificates is defined by the ITU-T X.509 international standard, which is described in RFC 3280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.

Certificate authorities (CAs) are entities that validate identities and issue certificates. An organization that wants to serve as its own CA can issue its own certificates, or an organization can purchase certificates from a trusted third-party CA. The methods used to validate an identity vary depending on the policies of a given CA. In general, before issuing a certificate, a CA must verify the identity of the entity and must digitally sign the certificate to ensure it cannot be modified. This ensures that a certificate issued by a CA binds a particular public key to the name of the entity the certificate identifies (such as the name of an employee).

In addition to a public key, a certificate includes the name of the entity it identifies, an expiration date, the name and URI of the CA that issued the certificate, a serial number, and the digital signature of the issuing CA, which creates a mathematical relationship between the signing CA certificate's public key and the public key of the certificate it signs. The CA's digital signature allows the certificate to function as a *letter of introduction* for users who know and trust the CA but do not know the entity identified by the certificate.

Because a certificate's expiration date is part of its signed contents, remote entities can verify that a certificate is valid and current.

Common types of certificates include the following:

- Certificate Authority certificates can sign other certificates.
- Server certificates are used on a server to enable a software client to verify the validity of the connection to a machine and to create an encrypted channel between a client and a server.
- Client certificates enable a server to verify a client's identity (certificate-based authentication) and enable a user to digitally sign or encrypt data. Client certificates, which are digitally signed by a trusted certificate authority, are stronger proof of a client's identity than username/password credentials alone.

NOTE: Digital Signature Algorithm (DSA) certificates are not supported.

Certificate Chains

A certificate chain is a sequence of certificates, where each certificate in the chain is signed by the certificate above it in the chain. At the top of the chain is a self-signed certificate. Each CA in the chain vouches for the identity in the entity to which it issues a signed digital certificate. Certificate chains establish a chain of trust; if you trust the CA at the top of the chain, this implies you can trust the signed certificates below it in the chain.

Certificate Revocation Lists

Under normal circumstances, a certificate remains valid until it reaches its expiration date. However, a certificate may become invalid before it expires. For example, if an employee whose identity is bound to a certificate terminates employment or if an enterprise suspects the confidentiality of the private key associated with a certificate's public key has been compromised, the certificate might be declared invalid and revoked before its expiration date.

When a CA revokes a certificate, it must let other entities know the certificate is no longer valid and not to accept it. A *certificate revocation list (CRL)* is a signed data structure that identifies the serial numbers of certificates that have been issued and subsequently revoked by the CA. When a remote entity is asked to use a certificate to verify a remote user's identity, it can download a current copy of the applicable CRL from a certification revocation list distribution point (CDP) and confirm that the certificate's serial number is not present. If a CRL has expired, the entity must connect to the CDP to download a new revocation list.

CRLs can be issued by a CA periodically (hourly, daily, or weekly) or as needed. When a certificate is revoked, its serial number is listed in the CRL, and that serial number remains on the CRL at least one period after the certificate's expiration date. CRLs, like certificates, can be distributed by untrusted servers and untrusted communications.

Under some circumstances, *latency* (the time between when a certificate is revoked and when the certificate's serial number appears on the CRL of the issuing CA) may be a concern. For example, if a

revocation is reported today, that revocation will not be reliably notified to certificate-using systems until all currently issued CRLs are updated, which may take hours, days, or even weeks. Online revocation checking can reduce the latency between a revocation report and the distribution of the information to relying parties.

If CRL checking is enabled, SBR Carrier uses the URI information contained in a client certificate to connect to the certificate's CDP. SBR Carrier then uses HTTP, LDAP, or a network file system to retrieve the appropriate CRLs. SBR Carrier stores these retrieved CRLs in the **CRLCache** directory under the *radiusdir* server directory.

When a client certificate is presented during EAP-TLS or EAP-TTLS authentication, SBR Carrier can evaluate the client's certificate chain against its set of stored CRLs to verify none of the certificates in the chain have been revoked.

You can configure the following settings for CRL checking:

- **Static CDPs**—A static CDP is a CDP whose address (URI) is specified in the [Static_CDPs] section of a TLS or TTLS initialization file.
- **CRL expiration**—The CRL checking feature can be configured to operate in *strict* or *lax* mode.
 - In strict mode, a cached CRL that has expired is immediately discarded; if SBR Carrier cannot acquire a new CRL in the allotted time during a CRL check on a chain, the user is rejected.
 - In lax mode, you can configure SBR Carrier to accept an expired CRL for a period past its expiration.

SBR Carrier attempts to obtain a current CRL whether it is running in strict or lax mode.

- **Missing CDP attribute**—When a CRL check is performed on a certificate chain, SBR Carrier reads the contents of the CDP attribute for each certificate past the root certificate and uses the CDP information to retrieve the appropriate CRL. If a non-root certificate in the chain does not contain a CDP attribute, no CRL checking is performed for that certificate. You can configure EAP-TLS to reject the user if it encounters a non-root certificate that is missing a CDP attribute.
- **Incomplete LDAP CDP**—Some CAs may create certificates that contain an LDAP-style CDP (*//ldap://...*) that does not specify the identity of the LDAP server to be queried. You can designate a default LDAP server that is used when such CDPs are encountered. If you do not designate a default LDAP server and an LDAP-style CDP is encountered, the CRL retrieval fails.
- **HTTP proxies for CRL checking**—Network security policy may prevent SBR Carrier from making a direct HTTP connection to a CDP. In such cases, you can configure SBR Carrier to download CRLs through an HTTP proxy server on its local network. Optionally, you can specify the hosts or domains that do not require an HTTP proxy.
- **CRL cache flushing**—You can flush the CRL caches used for EAP-TLS and EAP-TTLS authentication at any time.

Configuring Server Certificates

Valid server certificates must be in place on the SBR Carrier server before you configure the EAP-TLS, EAP-TTLS, and EAP-PEAP authentication protocols. You can add multiple server certificates to the SBR Carrier server.

NOTE: SBR Carrier does not support DSA type certificates.

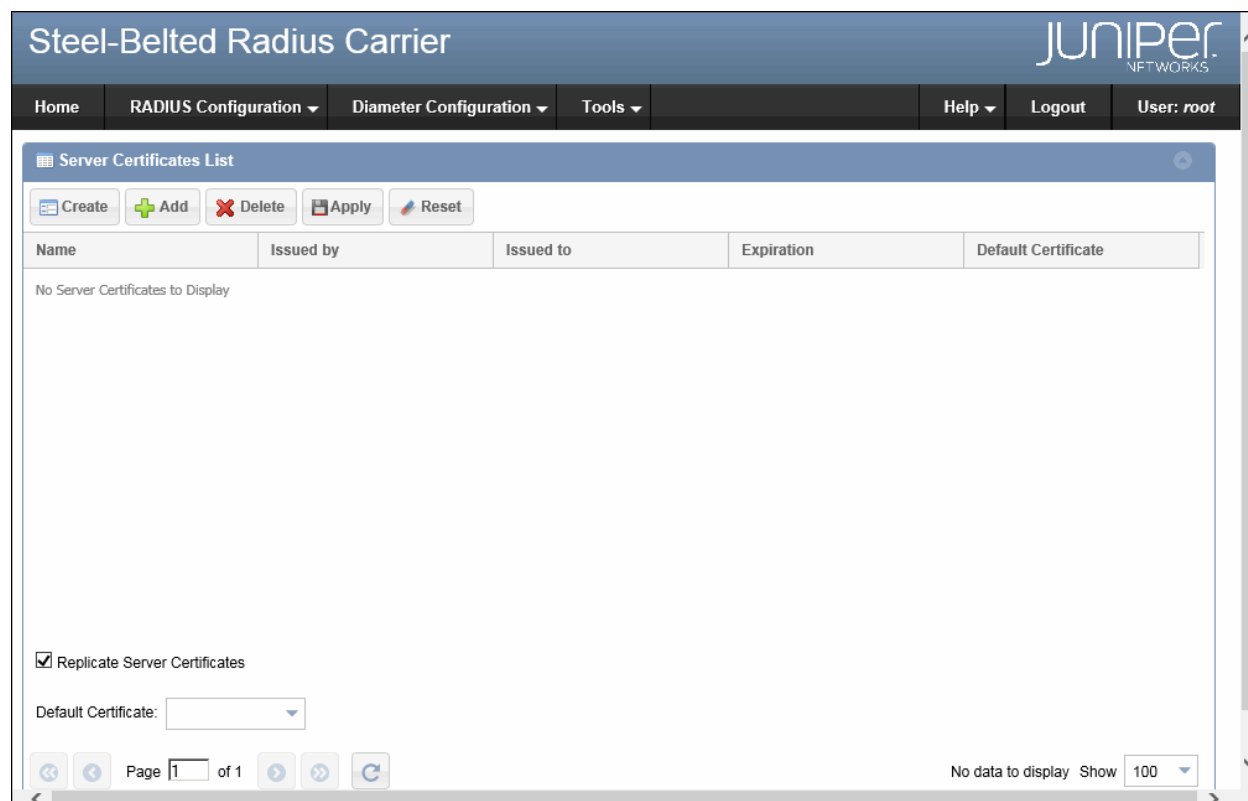
Creating a Certificate

To create a server certificate using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > Certificates**.

The **Server Certificates List** page ([Figure 81 on page 241](#)) appears.

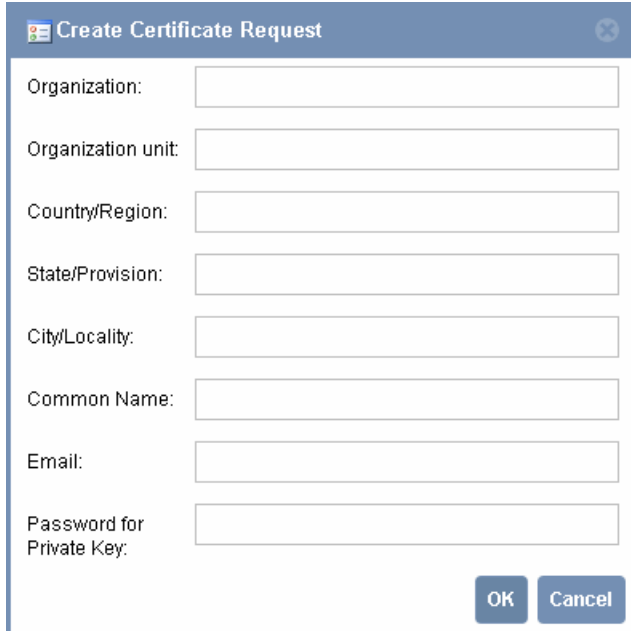
Figure 81: Server Certificates List Page



2. Click **Create**.

The **Create Certificate Request** dialog box ([Figure 82 on page 242](#)) appears.

Figure 82: Create Certificate Request Dialog

The image shows a 'Create Certificate Request' dialog box. It has a title bar with a close button. The dialog contains several text input fields: 'Organization:', 'Organization unit:', 'Country/Region:', 'State/Provision:', 'City/Locality:', 'Common Name:', 'Email:', and 'Password for Private Key:'. At the bottom right, there are 'OK' and 'Cancel' buttons.

Create Certificate Request

Organization:

Organization unit:

Country/Region:

State/Provision:

City/Locality:

Common Name:

Email:

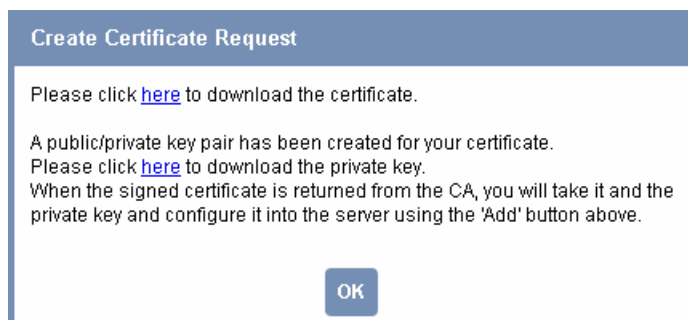
Password for Private Key:

OK Cancel

3. Enter a common name for the certificate in the **Common Name** field.
4. Enter a password for the private key in the **Password for Private Key** field.
5. Optionally, enter the organization name in the **Organization** field.
6. Optionally, enter the organization unit name in the **Organization unit** field.
7. Optionally, enter the country or region information in the **Country/Region** field.
8. Optionally, enter the state or provision information in the **State/Provision** field.
9. Optionally, enter the city or locality information in the **City/Locality** field.
10. Optionally, enter the e-mail information in the **Email** field.
11. Click **OK**.

The **Create Certificate Request** dialog box with links ([Figure 83 on page 243](#)) appears.

Figure 83: Create Certificate Request Dialog with Links



12. Click the links to download the created certificate and the private key.

13. Click **OK** to return to the **Server Certificates List** page (Figure 81 on page 241).

Adding a Certificate

To add a certificate to the SBR Carrier server using the Web GUI:

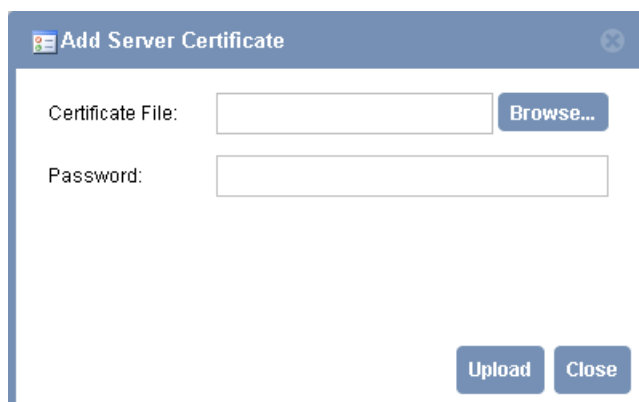
1. Select **RADIUS Configuration > Authentication Policies > Certificates**.

The **Server Certificates List** page (Figure 81 on page 241) appears.

2. Click **Add**.

The **Add Server Certificate** dialog box (Figure 84 on page 243) appears.

Figure 84: Add Server Certificate Dialog

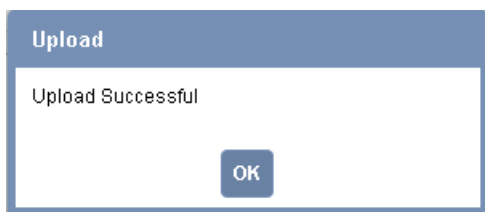


3. Click **Browse** to browse the location of your server certificate, select the certificate you want to use, and click **Open**.

4. Enter the password of the selected certificate in the **Password** field and click **Upload**.

The **Upload Successful** dialog box (Figure 85 on page 244) appears.

Figure 85: Upload Success Dialog



5. Click **OK**. The **Server Certificates List** page ([Figure 86 on page 245](#)) displays the added certificate details.

NOTE: By default, SBR Carrier notifies you about the certificate's expiration five days before the expiration date.

NOTE: If you upload a certificate file and a key file separately through the Web GUI, SBR Carrier converts these files (of type .cer, .der, or .crt) to the .pfx format and saves it in the server. The name of certificate file will be used for the server certificate name with the extension of .pfx.

Figure 86: Server Certificates List Page with Certificate Details

Steel-Belted Radius Carrier

Home | RADIUS Configuration | Diameter Configuration | Tools | Help | Logout | User: root

Server Certificates List

Create Add Delete Apply Reset

Name ↑	Issued by	Issued to	Expiration	Default Certificate
Cert_1.pfx	cert_1	cert_1	29 Jul 2017 10:14:25 GMT	✓
Cert_2.pfx	cert_example	cert_example	3 Aug 2017 07:38:19 GMT	

☐ Replicate Server Certificates

Default Certificate: Cert_1.pfx

Page 1 of 1

Displaying 1 - 2 of 2 Show 100

- Optionally, select the **Replicate Certificate** check box to replicate the added certificate.

NOTE: This certificate replication setting is applicable only if the SBR Carrier server acts as the primary server.

- Configure the **Default Certificate** from the list of added server certificates. The default certificate will be used as the server certificate for non-realm EAP requests.
- Click **Apply** to save the changes.

Deleting a Certificate

To delete a certificate from the SBR Carrier database:

- Select **RADIUS Configuration > Authentication Policies > Certificates**.

The **Server Certificates List** page (Figure 81 on page 241) appears.

- Select one or more certification entries that you want to delete.

3. Click **Delete**.
4. Click **Apply** to save the deletion in the SBR Carrier database.

Trusted Root Certificates

Some authentication methods, such as TLS, require that a trusted root certificate be present on the server.

If one is not already present, add the root certificate according to the system or CA's instructions. Then it add it to the SBR Carrier configuration so that any method that requires it has access.

Adding a Trusted Root Certificate

To add a trusted root certificate to the SBR Carrier server using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > Trusted Root Certificates**.

The **Root Certificates List** page ([Figure 87 on page 247](#)) appears.

Figure 87: Root Certificates List Page

The screenshot shows the 'Root Certificates List' page in the Steel-Belted Radius Carrier interface. The page has a dark blue header with the title 'Steel-Belted Radius Carrier' and the Juniper Networks logo. Below the header is a navigation bar with links: Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. The main content area has a title bar 'Root Certificates List' with a refresh icon. Below the title bar are four buttons: Add (green plus), Delete (red X), Apply (document with checkmark), and Reset (eraser). A table with five columns is shown: Name, Issued by, Issued to, Expiration, and Friendly name. The table is empty, with the text 'No Root Certificates to Display' below it. At the bottom left, there is a checkbox labeled 'Replicate Trusted Root Certificates' which is checked. At the bottom right, there is a pagination bar showing 'Page 1 of 1' and a 'Show 100' dropdown menu.

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

Root Certificates List

+ Add X Delete Apply Reset

Name	Issued by	Issued to	Expiration	Friendly name
No Root Certificates to Display				

☒ Replicate Trusted Root Certificates

Page 1 of 1 No data to display Show 100

2. Click **Add**.

The **Add Root Certificate** dialog box ([Figure 88 on page 248](#)) appears.

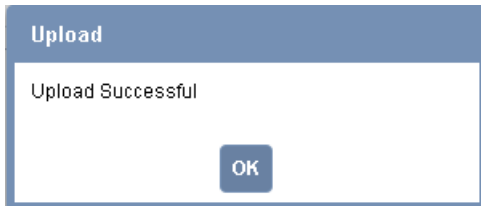
Figure 88: Add Root Certificate Dialog



3. Click **Browse** to browse the location of your trusted root server certificate. (This is normally / *ServiceRegistry-base/install/cacerts*, if the certificates were installed by **root** but can be in a different location, depending on the CA's installation program and the permissions of the user account that installed them).
4. Select the root certificate you want to use and click **Open**.
5. Click **Upload**.

The **Upload Successful** dialog box (Figure 89 on page 248) appears.

Figure 89: Upload Successful Dialog



6. Click **OK**.

The **Root Certificates List** page (Figure 87 on page 247) displays an updated list of trusted root certificate entries.

7. Optionally, select the **Replicate Trusted Root Certificates** check box to replicate the added root certificates.
8. Click **Apply** to save the changes.

Configuring a CRL Distribution Point Web Proxy

If your network security policies prohibit SBR Carrier from making a direct HTTP connection to a CRL distribution point (CDP), you can configure an HTTP proxy server to relay requests for updated certificate revocation lists to an external CDP.

To configure a CDP web proxy server using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > CDP Web Proxy Configuration**.

The **CDP Web Proxy Settings** page (Figure 90 on page 249) appears.

Figure 90: CDP Web Proxy Settings Page

2. Specify whether you want SBR Carrier to use a proxy to connect to an external CDP.
 - If you select the **Connect Directly to Internet** option button, SBR Carrier can connect to an external CDP without going through an HTTP proxy. If you select this option, you can skip the rest of this procedure.
 - If you select the **Connect to Internet via Proxy** option button, SBR Carrier must go through an HTTP proxy to connect to a CDP.
3. If you have selected the **Connect to Internet via Proxy** option button, enter the name or IP address and port number of the HTTP proxy in the **HTTP Proxy** and **Port** fields.
4. Optionally, enter names or IP addresses of hosts or the names of domains (separating each entry with a comma or semi-colon) for which no HTTP proxy is required in the **No Proxy For** field. If a CDP host matches an entry in this field, SBR Carrier bypasses the HTTP proxy and attempts to open a connection to the host directly.

SBR Carrier compares IP addresses and hostnames using an exact string match. For example, if you enter **cdp.juniper.net** in the exclusion list, that matches the CDP hostname **cdp.juniper.net** but not **host.cdp.juniper.net** or **host-cdp.juniper.net**.

To exclude all hosts within a domain (but not the hostname that matches the domain name), start the domain name with a period (**.juniper.net**). To exclude both the host and the domain **juniper.net**, create two entries in the exclusion list (**.juniper.net**, **juniper.net**).

Wildcard matching for host or domain names is not supported.

The values localhost and 127.0.0.1 are included in the **No Proxy For** list by default.

5. You can click **Flush CRL Caches** to purge all information in the TLS and TTLS CRL caches immediately. This removes all CRL entries for registered clients from the in-memory cache and deletes all files from the CRL cache directories.



CAUTION: If you click the **Flush CRL Caches** button, the caches are purged immediately. You are not prompted to confirm your action.

6. Click **Save** to save the changes.

Configuring Authentication Rejection Messages

When SBR Carrier issues an Access-Reject message in response to a failed authentication request, it can identify the reason why the request was rejected. You can configure the message text returned to the RADIUS client (and possibly to the user, if the RADIUS client forwards the message) when a particular type of error occurs. This text is inserted into the standard RADIUS attribute Reply-Message within the Access-Reject response.

To configure the text for authentication rejection messages using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > Reject Messages**.

The **Reject Messages** page (Figure 91 on page 251) appears.

Figure 91: Reject Messages Page

The screenshot shows the 'Reject Messages' configuration page in the Steel-Belted Radius Carrier interface. The page has a dark blue header with the 'Steel-Belted Radius Carrier' title and the 'JUNIPER NETWORKS' logo. Below the header is a navigation bar with links: Home, RADIUS Configuration (selected), Diameter Configuration, Tools, Help, Logout, and User: root. The main content area is titled 'Reject Messages' and contains four text input fields: 'Unknown User:', 'CheckList Failure:', 'Invalid Attribute:', and 'Other:'. At the bottom of the form are three buttons: 'Save', 'Reset', and 'Refresh'.

2. In the **Unknown User** field, enter the message text that SBR Carrier returns when the username and password authentication failed.
3. In the **CheckList Failure** field, enter the message text that SBR Carrier returns when the user was authenticated but is being rejected because the RADIUS request did not fulfill the requirements of the check list.
4. In the **Invalid Attribute** field, enter the message text that SBR Carrier returns when the request contained an attribute in violation of the RADIUS specification.
5. In the **Other** field, enter the message text that SBR Carrier returns when some other error, such as a resource failure, occurred.
6. Click **Save** to save the configuration.

To revert to the previous settings, click **Reset**.

NOTE: When you make changes to the **Reject Messages** page, the SBR Carrier audit log does not record specific information about the performed actions.

Configuring the Server

Depending on your authentication requirements, you may need to configure SBR Carrier to work with an external SQL or LDAP database service.

Configuring External Databases

If you want to use external databases for authentication or accounting purposes (and you did not configure this feature when prompted by the SBR Carrier installation script), you can set up external database configuration settings.

To configure SBR Carrier to work with an external database:

1. Optionally, perform the instructions in [“Configuring SQL Authentication” on page 442](#) or [“Configuring SQL Accounting” on page 458](#).
2. If you want to use SBR Carrier with an LDAP database, review your LDAP database vendor's documentation.
3. Perform the instructions in [“Configuring LDAP Authentication” on page 474](#).

Setting Up EAP Methods

IN THIS CHAPTER

- About the Extensible Authentication Protocol | 253
- EAP-TLS Authentication Protocol | 261
- EAP-TTLS Authentication Protocol | 286
- EAP-PEAP Authentication Protocol | 301
- EAP-MD5-Challenge Authentication Protocol | 314
- EAP-MS-CHAP-V2 Authentication Protocol | 314
- EAP-SIM and EAP-AKA Authentication Protocols | 315

This chapter presents an overview of concepts relating to the Extensible Authentication Protocol (EAP) and describes how to configure SBR Carrier to use EAP authentication methods and plug-ins.

EAP methods are used in conjunction with the other authentication methods supported by SBR Carrier to form an authentication policy. Which EAP methods are used in an authentication policy, and the order in which SBR Carrier attempts to use them to process an Access-Request, is user defined. Information on configuring your authentication policy to use EAP methods is detailed in [“Setting Up Authentication Policies” on page 232](#).

This chapter contains these topics:

About the Extensible Authentication Protocol

SBR Carrier supports the EAP, an open-ended standard for communication between NADs and servers that provides for future extensibility of authentication protocols.

EAP allows specialized knowledge about authentication protocols to be taken out of a NAD so that it acts solely as a conduit between the authentication server and client. This means that new types of authentication can be supported by adding the appropriate functionality to server and client, without any changes to PPP or NADs. When the authentication process is complete, the RADIUS server simply informs the NAD of the result.

SBR Carrier supports several EAP authentication mechanisms, such as TTLS, TLS, PEAP, AKA, and MD5-Challenge. Support for EAP has been designed to anticipate other authentication types as they become available.

For technical details about EAP, see RFC 3748, *Extensible Authentication Protocol (EAP)*, and RFC 2869, *RADIUS Extensions*.

Handling EAP Requests

The flow of RADIUS packets in an EAP environment is quite different from the transactions using standard user credentials (for example, PAP or CHAP).

Standard user credentials involve the transmission of a RADIUS request from the NAD to SBR Carrier and a response (either an Accept or Reject) from the server back to the NAD.

With EAP, the first packet sent from the NAD to SBR Carrier contains an **EAP-Message** attribute containing an EAP Identity Response. This is a signal sent by the system being authenticated that it wants to be authenticated by means of EAP. It is now up to SBR Carrier to select the EAP protocol with which it is to authenticate the end user.

The contents of the **User-Name** attribute is the only guideline available to SBR Carrier in selecting the appropriate EAP protocol. If SBR Carrier selects an EAP protocol that is not supported by the client, the client has the opportunity to send an EAP-NAK, and to request a specific alternate protocol.

NOTE: Given this general flow, a RADIUS request with EAP credentials must incur a minimum of two network round-trips between the NAD or Access Point and the SBR Carrier before reaching a successful conclusion.

NOTE: As per EAP RFC 3748, SBR Carrier can process User-Name (identity) attributes of 1020 characters in EAP-TTLS or EAP-PEAP authentication requests. Usernames containing more than 72 characters in cluster environment and 100 characters in standalone environment may be truncated during session storage.

Automatic EAP Helpers

Automatic *EAP helpers* serve as intermediaries between EAP and traditional authentication methods. These helper modules may be configured (using an associated **.eap** file) to work with existing authentication methods to shield the authentication methods from the particulars of the selected EAP protocol.

Table 33 on page 255 indicates whether each EAP type is implemented as an EAP helper or as an authentication method in SBR Carrier.

Table 33: EAP Implementations

EAP-Type	Implemented As
EAP-TTLS	Standalone Authentication Method
EAP-TLS	Standalone Authentication Method
EAP-TLS	Automatic EAP helper
EAP MD5-Challenge	Automatic EAP helper for CHAP
EAP MS-CHAP v2	Automatic EAP helper for MS-CHAP v2 (needed for PEAP)

Whether an automatic EAP helper can be used in conjunction with a specific authentication method depends on what types of credentials the authentication method supports.

The automatic EAP helper that implements EAP MD5-Challenge generates CHAP credentials. As such, EAP MD5-Challenge can be used only with authentication methods that support CHAP.

Table 34 on page 255 summarizes the support for MS-CHAP v2 and CHAP in the SBR Carrier authentication methods.

Table 34: MS-CHAP v2 and CHAP Support

Authentication Method	MS-CHAP v2	CHAP
LDAP	Yes, for BindName. (The password must be stored unencrypted or encrypted with enc-md5 by the LDAP server.) No, for Bind.	Yes, for BindName. (The password must be stored unencrypted or encrypted with enc-md5 by the LDAP server.) No, for Bind.
Local	Yes	Yes
Proxy RADIUS	Yes	Yes
SQL	Yes, if the password is stored unencrypted or encrypted with enc-md5 by the SQL database	Yes, if the password is stored unencrypted or encrypted with enc-md5 by the SQL database
UNIX User	No	No

Table 34: MS-CHAP v2 and CHAP Support *(continued)*

Authentication Method	MS-CHAP v2	CHAP
UNIX Group	No	No

Authentication Request Routing

The order in which authentication methods and automatic EAP helpers are called to handle an authentication request depends on two factors:

- The ordered list of enabled authentication methods (viewable in the **Authentication Methods** page in Web GUI). Refer to [“Order of Authentication Methods” on page 233](#) for information about defining the order of authentication methods for an authentication policy.
- The EAP-related configuration for each of the enabled authentication methods in the **eap.ini** file, which you configure from the **Authentication Methods** page.

When SBR Carrier receives an authentication request that does not contain EAP credentials, it passes the request to each enabled authentication method until one of the methods claims the request.

The EAP settings in the **eap.ini** file come into play only when a request with EAP credentials is received. An authentication request contains EAP credentials if it includes one or more **EAP-Message** attributes and contains no other form of user credentials (for example, **User-Password**).

EAP-Only Setting

When an authentication method's **EAP-Only** setting is 1, SBR Carrier prevents the authentication method from being called for any request that does not contain EAP credentials. Under this setting, the authentication method is also bypassed if an authentication request specifically requests an EAP protocol that is not listed in the authentication method's **EAP-Type** list in the **eap.ini** file.

NOTE: The PEAP authentication method plug-in converts the inner EAP credentials to PAP for security reasons. If you are using a third party authentication service with PEAP, set the EAP-Only setting to 0.

First-Handle-Via-Auto-EAP Setting

If your configuration involves clients using more than one EAP protocol, SBR Carrier must select an initial EAP protocol with which to proceed when receiving an authentication request with EAP credentials.

Selecting the incorrect EAP protocol is not fatal; the client simply sends an EAP- NAK in response to the server's selected protocol and suggests an alternate one. After one additional network round-trip, the correct EAP protocol becomes active.

Depending on the capabilities of the authentication methods being used, you may be able to cut out this additional network round-trip that affects a portion of your EAP-based authentication requests.

If an authentication method can check for the existence of a user and can retrieve the user's password information with only the information available in the authentication request (for example, the username), it is said to be *prefetch-capable*. A prefetch-capable authentication method ([Table 35 on page 257](#)) can determine whether a user exists in its database before committing to a specific EAP protocol.

If your authentication method is prefetch-capable, set **First-Handle-Via-Auto-EAP** to 0, indicating that the authentication method has the first chance to handle the request. Also set **First-Handle-Via-Auto-EAP** to 0 if the authentication method can handle EAP credentials directly, without an automatic helper EAP method.

By configuring the authentication method to be called first, SBR Carrier can delay selection of an EAP protocol until it has ascertained whether the user exists in a particular authentication method's database. This is a useful technique when you plan to use more than one EAP protocol, but you do not know which one the client wants. Even in this scenario, automatic EAP helpers may still end up performing the EAP protocol processing; they take over after the authentication method has retrieved a user's password information, rather than before.

The goal of an automatic EAP helper is to generate credentials against which traditional authentication methods (ones that do not understand EAP) can operate. After an automatic EAP helper has generated these credentials, the authentication method that triggered the use of the helper is checked first for a password/credential match. If a match is not present, the same traditional credentials are passed to all remaining enabled authentication methods in the primary list (in the order in which they appear in the list).

Table 35: Authentication Method Prefetch Capability

Authentication Method	Prefetch Capable?
LDAP	Yes, if using BindName (rather than the Bind option)
Native User	Yes
SQL	Yes, if password does not need to be used as an input parameter in the SQL statement
UNIX User	No

NOTE: If you enable the lockout facility in SBR Carrier and you use a tunneled authentication method (TTLS or PEAP) with a prefetch-capable method (Native, SQL, or LDAP) and an enabled EAP protocol (MS-CHAPv2, MD5-Challenge, TLS), then you must enable **First Handle Via Auto-EAP** in that prefetch-capable method to prevent the outer username (**anonymous**) from being added to the lockout list.

Otherwise, when SBR Carrier receives an authentication request that uses an EAP method that has not been configured, it rejects the user (because the EAP method is not considered to be valid until it is configured) and add the outer username (**anonymous**) to its lockout list. This locks out all users with an outer authentication name of **anonymous** until the lockout period expires.

EAP-NAK Notifications

If you are supporting only one type of client or only one EAP protocol, SBR Carrier selects that EAP protocol for all EAP-based authentication requests it receives. If you are planning to support multiple EAP protocols and do not intend to maintain databases that track the appropriate EAP protocol on a user-by-user basis, SBR Carrier automatically selects the appropriate EAP protocol for you.

When multiple EAP protocols are in play, configure each authentication method you plan to use with all the EAP protocols that may be used with it. In this configuration, when SBR Carrier receives an authentication request containing EAP information, it chooses the first EAP protocol listed for the first authentication method that claims the request. If the client requires a different EAP protocol, it sends back an EAP-NAK that specifies the EAP protocol it wants to use.

After receiving an EAP-NAK, SBR Carrier performs a scan of the authentication methods, in search of the first authentication method that has the requested EAP protocol listed (the authentication method may support this EAP protocol directly or with the help of an automatic EAP helper).

If the requested EAP protocol does not appear in any of the authentication methods' lists of supported EAP protocols, SBR Carrier rejects the authentication request.

Reauthenticating Connections

Most access points understand only a limited number of attributes that may be included in a RADIUS response to signal that the user has been accepted. The Session Timeout attribute is of particular significance in a WLAN realm because it instructs the Access Point how long the user can remain connected to a WLAN before having to reauthenticate with SBR Carrier.

You can configure your choice of **Session Timeout** settings using standard SBR Carrier reply-list items on a user-by-user basis. If you are using EAP-TLS or EAP-TTLS to authenticate users, you can also have these plug-ins automatically generate **Session Timeout** attributes based on policies set in their configuration

files. This level of control is necessary for EAP-TLS and EAP-TTLS as these plug-ins also support *session resumption*, a quicker method of reauthenticating users. The value in the **Session-Timeout** attribute may need to be dynamically calculated in these cases.

NOTE: Not all access points support the **Session-Timeout** attribute. Check the specifications for your access points to determine whether this configuration must be performed in a fixed manner on the Access Point or whether the Access Point defers to the server.

Certificates

A *certificate* is an electronic data structure used to identify an individual, a server, a company, or some other entity, and to associate that identity with a public key and an associated private key. Like a passport, a certificate provides generally recognized proof of an entity's identity. Certificates bind public key values to entities, so that remote users of an entity's public key can be certain the associated private key is owned by the correct person or system. Certificates help prevent the use of fake public keys for impersonation. Only the public key certified by the certificate works with the corresponding private key possessed by the entity identified by the certificate. The most widely accepted format for certificates is defined by the ITU-T X.509 international standard, which is described in RFC 5280 (which made RFC 3280 obsolete), *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.

Certificate authorities (CAs) are entities that validate identities and issue certificates. An organization that wants to serve as its own CA can issue its own certificates, or an organization can purchase certificates from a trusted third-party CA. The methods used to validate an identity vary depending on the policies of a given CA. In general, before issuing a certificate, a CA must verify the identity of the entity and must digitally sign the certificate to ensure it cannot be modified. This ensures that a certificate issued by a CA binds a particular public key to the name of the entity the certificate identifies (such as the name of an employee).

In addition to a public key, a certificate includes the name of the entity it identifies, an expiration date, the name and URI of the CA that issued the certificate, a serial number, and the digital signature of the issuing CA, which creates a mathematical relationship between the signing CA certificate's public key and the public key of the certificate it signs. The CA's digital signature allows the certificate to function as a *letter of introduction* for users who know and trust the CA but do not know the entity identified by the certificate.

Because a certificate's expiration date is part of its signed contents, remote entities can verify that a certificate is valid and current.

Common types of certificates include the following:

- Certificate Authority certificates can sign other certificates.
- Server certificates are used on a server to enable a software client to verify the validity of the connection to a machine and to create an encrypted channel between a client and a server.

- Client certificates enable a server to verify a client's identity (certificate-based authentication) and to enable a user to digitally sign or encrypt data. Client certificates, which are digitally signed by a trusted certificate authority, are stronger proof of a client's identity than username/password credentials alone.

NOTE: SBR Carrier does not support Digital Signature Algorithm (DSA) type certificates.

Certificate Chains

A certificate chain is a sequence of certificates, where each certificate in the chain is signed by the certificate above it in the chain. At the top of the chain is a self-signed certificate. Each CA in the chain vouches for the identity in the entity to which it issues a signed digital certificate. Certificate chains establish a chain of trust; if you trust the CA at the top of the chain, this implies you can trust the signed certificates below it in the chain.

Certificate Revocation Lists

Under normal circumstances, a certificate remains valid until it reaches its expiration date. However, a certificate may become invalid before it expires. For example, if an employee whose identity is bound to a certificate terminates employment or if an enterprise suspects the confidentiality of the private key associated with a certificate's public key has been compromised, the certificate might be declared invalid and revoked before its expiration date.

When a CA revokes a certificate, it must let other entities know the certificate is no longer valid and should not be accepted. A *certificate revocation list (CRL)* is a signed data structure that identifies the serial numbers of certificates that have been issued and subsequently revoked by the CA. When a remote entity is asked to use a certificate to verify a remote user's identity, it can download a current copy of the applicable CRL from a CRL distribution point (CDP) and confirm that the certificate's serial number is not present. If a CRL has expired, the entity must connect to the CDP to download a new revocation list.

CRLs can be issued by a CA periodically (hourly, daily, or weekly) or as needed. When a certificate is revoked, its serial number is listed in the CRL, and that serial number remains on the CRL at least one period after the certificate's expiration date. CRLs, like certificates, can be distributed by untrusted servers and untrusted communications.

Under some circumstances, *latency* (the time between when a certificate is revoked and when the certificate's serial number appears on the CRL of the issuing CA) may be a concern. For example, if a revocation is reported today, certificate-using systems are not reliably notified until all currently issued CRLs are updated, which may take hours, days, or even weeks. Online revocation checking can reduce the latency between a revocation report and the distribution of the information to relying parties.

If CRL checking is enabled, SBR Carrier uses the URI information contained in a client certificate to connect to the certificate's CDP. SBR Carrier then uses HTTP, LDAP, or a network file system to retrieve the appropriate CRLs. SBR Carrier stores these retrieved CRLs in the **CRLCache** directory under the *radiusdir* server directory.

When a client certificate is presented during EAP-TLS or EAP-TTLS authentication, SBR Carrier can evaluate the client's certificate chain against its set of stored CRLs to verify none of the certificates in the chain have been revoked.

You can configure the following settings for CRL checking:

- **Static CDPs**—A static CDP is a CDP whose address (URI) is specified in the [Static_CDPs] section of a TLS or TTLS initialization file.
- **CRL expiration**—The CRL checking feature can be configured to operate in *strict* or *lax* mode.
 - In strict mode, a cached CRL that has expired is immediately discarded; if SBR Carrier cannot acquire a new CRL in the allotted time during a CRL check on a chain, the user is rejected.
 - In lax mode, you can configure SBR Carrier to accept an expired CRL for a period past its expiration.

SBR Carrier attempts to obtain a current CRL whether it is running in strict or lax mode.

- **Missing CDP attribute**—When a CRL check is performed on a certificate chain, SBR Carrier reads the contents of the CDP attribute for each certificate past the root certificate and uses the CDP information to retrieve the appropriate CRL. If a non-root certificate in the chain does not contain a CDP attribute, no CRL checking is performed for that certificate. You can configure EAP-TLS to reject the user if it encounters a non-root certificate that is missing a CDP attribute.
- **Incomplete LDAP CDP**—Some CAs may create certificates that contain an LDAP-style CDP (`//ldap://...`) that does not specify the identity of the LDAP server to be queried. You can designate a default LDAP server be used when such CDPs are encountered. If you do not designate a default LDAP server and an LDAP-style CDP is encountered, the CRL retrieval fails.
- **HTTP proxies for CRL checking**—Network security policy may prevent SBR Carrier from making a direct HTTP connection to a CDP. In such cases, you can configure SBR Carrier to download CRLs through an HTTP proxy server on its local network. Optionally, you can specify the hosts or domains that do not require an HTTP proxy.
- **CRL cache flushing**—You can flush the CRL caches used for EAP-TLS and EAP-TTLS authentication at any time.

EAP-TLS Authentication Protocol

The EAP-TLS (Transport Layer Security) protocol requires that both user and authentication server have certificates for mutual authentication. While the mechanism is very strong, it requires that the corporation that deploys it maintain a certificate infrastructure for all of its users.

EAP-TLS can be deployed as an *authentication method* or as an *automatic EAP helper*.

- When EAP-TLS is deployed as an authentication method, it appears in the **Authentication Methods** page in Web GUI. You can use the **Authentication Methods** page to enable the EAP-TLS method and specify its sequence relative to other authentication methods SBR Carrier uses.

When EAP-TLS is deployed as an authentication method, you can configure it to perform CRL checking. When CRL checking is enabled, EAP-TLS confirms that the client's certificate chain traces back to one of the trusted root certificates installed at initialization and checks the serial number of each certificate in the chain against the contents of CRLs to verify that none of the certificates in the chain have been revoked.

You can configure the **tlsauth.aut** file to call a fixed profile when TLS-EAP is used. This profile specifies the attributes that are sent back in response to a successful authentication.

You cannot use secondary authorization when EAP-TLS is deployed as an authentication method.

- When EAP-TLS is deployed as an automatic EAP helper, you must list TLS in the EAP-Type list of an authentication method. When EAP-TLS is triggered, the **tlsauth** authentication goes through the TLS handshake required by the EAP-TLS specification. Assuming the user provides a certificate that the server can verify against a list of trusted root certificates, the EAP-TLS part of the exchange concludes successfully.

You may not want to grant access to your network to every user with a trusted certificate. By enabling the optional secondary authorization feature of the **tlsauth** plug-in, you can have SBR Carrier authorize users with valid certificates on a case-by-case basis. Secondary authorization also allows you to include user-specific attributes in an Access-Accept response; these attributes can be used to communicate options that are to be active for a user's connection to the NAD. Without secondary authorization, the only attributes returned on an Access-Accept are those generated by the **tlsauth** plug-in itself (**termination-action** and **session-limit**).

If you enable the TLS authentication method, secondary authorizations must be performed by local authentication methods (they cannot be proxied). The authentication method you select for secondary authorizations must be able to authenticate users in a single pass; it cannot challenge the authorization request and request additional information. The username employed during secondary authorization is derived from a field in the user's certificate. Because a user's certificate does not include a password, you must configure **tlsauth** to make the secondary authorization request with no password or with a fixed password.

If you configure secondary authorization with no password, your selected authentication method must be capable of handling requests that do not include passwords; the only authentication methods that support this style of authentication and ship with SBR Carrier are Native User, LDAP and SQL. If you configure secondary authorization with a fixed password, you can use any authentication method that supports PAP authentication. In this configuration all user records must have the same fixed password.

Configuring EAP-TLS as an EAP Authentication Method

NOTE: A valid server certificate must be in place on the SBR Carrier server before you configure the EAP-TLS authentication protocol. For information about configuring certificates, see [“Certificates” on page 238](#).

To configure EAP-TLS as an authentication method using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > EAP Methods**.

The **EAP Methods List** page (Figure 92 on page 264) appears.

Figure 92: EAP Methods List Page



2. Select **EAP-TLS**.

The **Selected EAP Method: EAP-TLS** pane (Figure 93 on page 265) appears.

Figure 93: Selected EAP Method: EAP-TLS Pane

The screenshot shows the Steel-Belted Radius Carrier web interface. At the top is a navigation bar with links: Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. Below this is the 'EAP Methods List' section, which contains a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Below the table is the 'Selected EAP Method: EAP-TLS' configuration pane. It has a tabbed interface with three tabs: 'Client Certificate Validation' (selected), 'Session Resumption', and 'Advanced Server Settings'. The 'Client Certificate Validation' tab contains the following settings:

- ☐ Enable EAP-TLS Method
- ☐ Enable CRL Checking:
 - Retrieval Timeout: 5
 - Expiration Grace Period: 0
- ☒ Allow Missing CDP Attribute:
- ☐ CRL Cache Timeout Period: 168
- Default LDAP Server Name:
- ☐ Verify that Client Certificates are Published to User Accounts:
- ☐ Include Certificate Info:

At the bottom of the pane are buttons for Save, Reset, and Cancel.

3. Select the **Enable EAP-TLS Method** check box to enable the EAP-TLS method.

NOTE: You can also enable the EAP-TLS method by using the **EAP Methods List** page. In the **EAP Methods List** page, click the **Status** column of the **EAP-TLS** entry, select the appeared check box, and click **Apply**.

4. Configure client certificate validation for the EAP-TLS protocol. For more information about configuring client certificate validation, see [“Configuring Client Certificate Validation—EAP-TLS” on page 266](#).
5. Configure session resumption for the EAP-TLS protocol. For more information about configuring session resumption, see [“Configuring Session Resumption—EAP-TLS” on page 268](#).

6. Configure advanced server settings for the EAP-TLS protocol. For more information about configuring advanced server settings, see [“Configuring Advanced Server Settings—EAP-TLS”](#) on page 271.
7. Click **Save** to save the configuration.

Configuring Client Certificate Validation—EAP-TLS

Client certificate validation settings enable you to specify how SBR Carrier performs CRL checking.

To configure client certificate validation for the EAP-TLS protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TLS** pane, click the **Client Certificate Validation** tab ([Figure 94 on page 266](#)).

Figure 94: EAP-TLS—Client Certificate Validation

The screenshot displays the Steel-Belted Radius Carrier Web GUI. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user (root). The main content area is titled "EAP Methods List" and shows a table of EAP methods. The "Selected EAP Method: EAP-TLS" pane is open, showing the "Client Certificate Validation" tab. The configuration options for this tab are as follows:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Selected EAP Method: EAP-TLS

☐ Enable EAP-TLS Method

Client Certificate Validation | Session Resumption | Advanced Server Settings

Enable CRL Checking: ☐

Retrieval Timeout: 5

Expiration Grace Period: 0

Allow Missing CDP Attribute: ☒

CRL Cache Timeout Period: ☐ 168

Default LDAP Server Name:

Verify that Client Certificates are Published to User Accounts: ☐

Include Certificate Info: ☐

Buttons: Save, Reset, Cancel

2. Select the **Enable CRL Checking** check box to enable certificate revocation list checking.

3. In the **Retrieval Timeout** field, enter the number of seconds you want EAP-TLS to wait for the CRL checking transaction to complete.

When CRL retrieval takes longer than the specified time, the user's authentication request is rejected.

4. In the **Expiration Grace Period** field, enter the number of seconds a CRL is still considered acceptable after it has expired.

EAP-TLS always attempts to retrieve a new CRL when it is presented with a certificate chain and it finds an expired CRL in its cache.

- If you enter 0 to specify strict expiration mode, EAP-TLS rejects an expired CRL.
- If you enter a value greater than 0, lax expiration mode is used. An expired CRL is acceptable if the time it expired is within the specified grace period.

5. Select the **Allow Missing CDP Attribute** check box if you want to enable SBR Carrier to accept a non-root certificate without a CDP attribute.

Without a CDP attribute, EAP-TLS cannot retrieve a CRL and cannot perform a revocation check on the certificate.

If you select the **Allow Missing CDP Attribute** check box, EAP-TLS accepts such certificates and skips CRL checking for them.

If you clear the **Allow Missing CDP Attribute** check box, EAP-TLS does not accept a CRL with a missing CDP attribute.

6. If you want to specify a CRL cache timeout period, select the **CRL Cache Timeout Period** check box and enter the number of hours in the **CRL Cache Timeout Period** field.

- If you do not enable this setting, the CRL is refreshed whenever it expires.
- If you enable this setting and enter 0, SBR Carrier always regards the CRL in the cache as expired and downloads a new CRL every time it receives a client certificate request.
- If you enable this setting and enter a number greater than 0, the CRL begins to expire when the age of the CRL in the cache exceeds the number of hours specified in this field or when the scheduled CRL expiration time occurs, whichever comes first.

After a CRL has expired (because its scheduled expiration time has passed or because the CRL cache has timed out), SBR Carrier uses the expiration grace period to determine whether to use the current CRL.

7. Enter the name of the LDAP server to use if the CDP contains a value that begins with the string `//ldap:\\` in the **Default LDAP Server Name** field.

CDPs generated by some CAs do not include the identity of the LDAP server. If you expect to encounter certificates with this style CDP, specify the name of the LDAP server that contains the CRLs.

If you do not specify a server name and such certificates are encountered, the CRL retrieval fails.

8. Select the **Verify that Client Certificates are Published to User Accounts** check box.
9. If you want the EAP-TLS plug-in to add four attributes to the outer request before the secondary authorization check is performed, select the **Include Certificate Info** check box.

When the **Include Certificate Info** check box is selected, SBR Carrier adds the following attributes to the request:

- The **Funk-Peer-Cert-Subject** attribute contains the value of the Subject attribute in the client certificate.
- The **Funk-Peer-Cert-Principal** attribute contains the value of the principal name (subject alternate name or other name) attribute of the client certificate.
- The **Funk-Peer-Cert-Issuer** attribute contains the value of the Issuer attribute in the client certificate.
- The **Funk-Peer-Cert-Hash** attribute contains a hexadecimal ASCII representation of the SHA1 hash of the client certificate.

These attributes are ignored if the authentication method that performs the authentication check does not use them.

Configuring Session Resumption—EAP-TLS

Session resumption settings control whether and under what circumstances session resumption is permitted.

NOTE: For session resumption to work, the NAD must be configured to handle the **Session-Timeout** return list attribute, so that the NAD can notify the client to reauthenticate after the session timer has expired.

To configure session resumption for the EAP-TLS protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TLS** pane, click the **Session Resumption** tab (Figure 95 on page 269).

Figure 95: EAP-TLS—Session Resumption

The screenshot shows the Steel-Belted Radius Carrier web interface. At the top, there is a navigation bar with the title "Steel-Belted Radius Carrier" and the Juniper Networks logo. Below the navigation bar, there are tabs for "Home", "RADIUS Configuration", "Diameter Configuration", "Tools", "Help", "Logout", and "User: root".

The main content area is divided into two sections. The top section is titled "EAP Methods List" and contains a table with two columns: "Name" and "Status". The table lists four EAP methods: EAP-TLS, EAP-TTLS, EAP-PEAP, and EAP-TLS Helper, all of which are currently "Disabled".

The bottom section is titled "Selected EAP Method: EAP-TLS" and contains a sub-section for "Session Resumption". This sub-section has three tabs: "Client Certificate Validation", "Session Resumption", and "Advanced Server Settings". The "Session Resumption" tab is currently selected. It contains three input fields, each with a value of "0": "Session Timeout(In Seconds)", "Termination Action:", and "Resumption Limit(In Seconds)".

At the bottom of the "Session Resumption" section, there are three buttons: "Save", "Reset", and "Cancel".

2. In the **Session Timeout(In Seconds)** field, enter the maximum number of seconds you want the client to remain connected to the NAD before having to reauthenticate.

If you enter a number greater than 0, the lesser of this value and the remaining resumption limit is sent in a **Session-Timeout** attribute to the RADIUS client on the RADIUS Access-Accept response.

If you enter 0, a **Session-Timeout** attribute is not generated directly. A 0 does not prevent the authentication methods that perform secondary authorization from providing a value.

Entering a value such as 600 seconds (10 minutes) does not necessarily cause a full reauthentication to occur every 10 minutes. You can configure the resumption limit to make most reauthentications fast and computationally efficient.

BEST PRACTICE: Using the Resumption Limit Option Effectively

Two scenarios where the resumption limit can be used effectively:

- In a wireless environment, the client is moving between access points. The resumption limit can be tuned to make the handover between access points smoother by not forcing a complete reauthorization that requires repeated verification of user information.

When the new access point queries SBR Carrier, the server replies that the session ID is already valid. Because it is known to be good, repeating the inner authentication is not required, which saves some time. The access point acknowledges the *reauthorization not required* message and the session continues.

- Another use for resumption limit occurs when the server ordinarily requires the client to reauthorize every 10 minutes or so, to ensure the client is still connected. Setting the resumption limit to 3600 seconds with a session timeout of 600 seconds means that the interval reauthorizations are fast and efficient, and a complete reauthorization is required just once an hour instead of every 10 minutes.

3. In the **Termination Action** field, enter the value to be returned in a **Termination-Action** attribute.

The **Termination-Action** attribute is a standard attribute supported by most access points and determines what happens when the session timeout is reached. Valid values are:

- -1: Do not send the attribute, the default. This does not prevent the authentication methods performing secondary authorization from providing a value.
- 0: Send the **Termination-Action** attribute with a value of 0.
- 1: Send the **Termination-Action** attribute with a value of 1.

4. Enter the maximum number of seconds you want the client to be able to reauthenticate using the TLS session resumption feature in the **Resumption Limit(In Seconds)** field.

This type of reauthentication is fast and computationally efficient. It does, however, depend on previous authentications and is not as secure as a complete (but computationally expensive) authentication. Specifying a value of 0 disables the session resumption feature.

BEST PRACTICE: Using the Resumption Limit Option Effectively

Two scenarios where the resumption limit can be used effectively:

- In a wireless environment, the client is moving between access points. The resumption limit can be tuned to make the handover between access points smoother by not forcing a complete reauthorization that requires repeated verification of user information.

When the new access point queries SBR Carrier, the server replies that the session ID is already valid. Because it is known to be good, repeating the inner authentication is not required, which saves some time. The access point acknowledges the *reauthorization not required* message and the session continues.

- Another use for resumption limit occurs when the server ordinarily requires the client to reauthorize every 10 minutes or so, to ensure the client is still connected. Setting the resumption limit to 3600 seconds with a session timeout of 600 seconds means that the interval reauthorizations are fast and efficient, and a complete reauthorization is required just once an hour instead of every 10 minutes.

Configuring Advanced Server Settings—EAP-TLS

You use advanced server settings to specify the manner in which the inner authentication step operates.

To configure advanced server settings for the EAP-TLS protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TLS** pane, click the **Advanced Server Settings** tab (Figure 96 on page 272).

Figure 96: EAP-TLS—Advanced Server Settings

The screenshot shows the Steel-Belted Radius Carrier configuration interface. At the top, there's a navigation bar with 'Home', 'RADIUS Configuration', 'Tools', 'Help', 'Logout', and 'User: root'. Below this is the 'EAP Methods List' section, which contains a table with columns 'Name' and 'Status'. The table lists EAP-PEAP, EAP-TLS (highlighted), EAP-TTLS, and EAP-TLS Helper, all with a status of 'Disabled'. Below the table are 'Apply', 'Reset', and 'Refresh' buttons. The main section is titled 'Selected EAP Method: EAP-TLS'. It has a checkbox for 'Enable EAP-TLS Method' which is unchecked. Below this are three tabs: 'Client Certificate Validation', 'Session Resumption', and 'Advanced Server Settings' (which is selected). The 'Advanced Server Settings' tab contains several configuration fields: 'TLS Message Fragment Length' (set to 1020), 'Max Transaction Time' (set to 120), 'Challenge Timeout' (set to 30), 'Return MPPE Keys' (checked), 'TLS Protocol Version' (set to TLSv1), 'DH Prime Bits' (set to 1024), 'Cipher Suites' (set to 0x003C,0x003D,0x0067,C), 'Verify User Name is Principal' (unchecked), and 'Use Profile' (unchecked). At the bottom of the configuration pane are 'Save', 'Reset', and 'Cancel' buttons.

Name	Status
EAP-PEAP	Disabled
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-TLS Helper	Disabled

Selected EAP Method: EAP-TLS

☐ Enable EAP-TLS Method

Advanced Server Settings

TLS Message Fragment Length: 1020

Max Transaction Time: 120

Challenge Timeout: 30

Return MPPE Keys: ☒

TLS Protocol Version: TLSv1

DH Prime Bits: 1024

Cipher Suites: 0x003C,0x003D,0x0067,C

Verify User Name is Principal: ☐

Use Profile: ☐

2. In the **TLS Message Fragment Length** field, enter the maximum length of the TLS message that may be generated during each iteration of the TLS exchange.
Enter a number in the range 500 through 4096 bytes.
 - The default length for TLS messages is 1020 bytes, which prevents the RADIUS challenge response (carried in a UDP packet) from exceeding one Ethernet frame.
 - Some access points may have problems with RADIUS responses or EAP messages that exceed the size of one Ethernet frame (1500 bytes including IP/UDP headers).
3. In the **Max Transaction Time** field, enter the maximum number of seconds you want for the EAP-TLS authentication sequence to complete.

- Enter a value in the range 1 through 3600 seconds. The default value is 120 seconds.
 - If the authentication sequence takes longer than this setting, user authentication is terminated.
4. In the **Challenge Timeout** field, enter the number of seconds after which a challenge request times out.
- You can enter a value greater than or equal to 1 second, but this value must not exceed the value specified in the **Max Transaction Time** field. The default value is 30 seconds.
5. Select the **Return MPPE Keys** check box to specify whether the TLS authentication method includes **RADIUS MS-MPPE-Send-Key** and **MS-MPPE-Recv-Key** attributes in the final RADIUS Access-Accept response sent to the access point.
- Disable this option for WiMAX.
- Select this check box if the access point needs to key the Wired Equivalent Privacy (WEP) encryption. If the access point is authenticating only end users and WEP is not being used, you can clear this check box.
6. Use the **TLS Protocol Version** list to specify the TLS protocol version on which the server expects the client to initiate the handshake process.
- Valid values are TLSv1, TLSv1.1, and TLSv1.2.
7. Use the **DH Prime Bits** list to specify the number of bits in the prime number that the module uses for Diffie-Hellman exponentiation.
- Selecting a longer prime number makes the system less susceptible to certain types of attacks but requires more CPU processing to compute the Diffie-Hellman key agreement operation.
- Valid values are 512, 1024, 1536, 2048, 3072, and 4096 bits.
8. In the **Cipher Suites** field, enter the TLS cipher suites (in order of preference) that the server is to use.
- These cipher suites are documented in RFC 2246, *The TLS Protocol Version 1*, RFC 4346, *The TLS Protocol Version 1.1*, and RFC 5246, *The TLS Protocol Version 1.2*.

The default value is: 0x0067,0x006B,0xC030,0xC028,0xC014,0xC013.

[Table 36 on page 273](#) lists the tested cipher suites and their TLS protocol versions.

Table 36: Tested Cipher Suites

Tested Cipher Suites	TLS Protocol Version
0xC013	TLS 1.0
0xC014	TLS 1.0
0x003C	TLS 1.2
0x003D	TLS 1.2

Table 36: Tested Cipher Suites (continued)

Tested Cipher Suites	TLS Protocol Version
0x0067	TLS 1.2
0x006B	TLS 1.2
0x009C	TLS 1.2
0x009D	TLS 1.2
0x009E	TLS 1.2
0x009F	TLS 1.2
0xC027	TLS 1.2
0xC028	TLS 1.2
0xC02F	TLS 1.2
0xC030	TLS 1.2

NOTE: SBR Carrier does not provide support for TLSv1.3.

SBR Carrier supports the following weak cipher suites: 0x002F, 0x0033, 0x0035, 0x0039, 0x003C, and 0x003D. These weak ciphers are not supported by default and need to be configured in the **Cipher Suites** field.

9. Select the **Verify User Name is Principal** check box to verify whether the username is used as the principal name (subject alternate name or other name) of the client certificate.
10. If you want to associate a profile with the EAP-TLS protocol, select the **Use Profile** check box and use the **Use Profile** list to select the profile.

Configuring EAP-TLS as an Automatic EAP Helper

NOTE: You must configure the server certificate for the SBR Carrier server before you use the EAP TLS helper. For information about configuring your server certificate, see [“Configuring Server Certificates” on page 241](#).

To configure EAP-TLS as an EAP helper using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > EAP Methods**.

The **EAP Methods List** page ([Figure 92 on page 264](#)) appears.

2. Select **EAP-TLS Helper**.

The **Selected EAP Method: EAP-TLS Helper** pane ([Figure 97 on page 276](#)) appears.

Figure 97: Selected EAP Method: EAP-TLS Helper Pane

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. The main content area is divided into two sections.

The first section, titled "EAP Methods List", contains a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

The second section, titled "Selected EAP Method: EAP-TLS Helper", contains a checkbox labeled "Enable EAP-TLS Helper Method" which is currently unchecked. Below this are four tabs: "Client Certificate Validation", "Secondary Authorization", "Session Resumption", and "Advanced Server Settings". The "Client Certificate Validation" tab is active, showing the following configuration options:

- Enable CRL Checking: ☐
- Retrieval Timeout: 5
- Expiration Grace Period: 0
- Allow Missing CDP Attribute: ☒
- CRL Cache Timeout Period: ☐ (dropdown menu)
- Default LDAP Server Name: (text input field)

At the bottom of the configuration pane are buttons for Save, Reset, and Cancel.

3. Select the **Enable EAP-TLS Helper Method** check box to enable the EAP-TLS helper method.

NOTE: You can also enable the EAP-TLS helper method by using the **EAP Methods List** page. In the **EAP Methods List** page, click the **Status** column of the **EAP-TLS Helper** entry, select the appeared check box, and click **Apply**.

4. Configure client certificate validation for the EAP-TLS helper protocol. For more information about configuring client certificate validation, see [“Configuring Client Certificate Validation—EAP-TLS Helper” on page 277](#).
5. Configure secondary authorization for the EAP-TLS helper protocol. For more information about configuring secondary authorization, see [“Configuring Secondary Authentication—EAP-TLS Helper” on page 279](#).

6. Configure session resumption for the EAP-TLS helper protocol. For more information about configuring session resumption, see [“Configuring Session Resumption—EAP-TLS Helper” on page 282](#).
7. Configure advanced server settings for the EAP-TLS helper protocol. For more information about configuring advanced server settings, see [“Configuring Advanced Server Settings—EAP-TLS Helper” on page 284](#).
8. Click **Save** to save the configuration.

Configuring Client Certificate Validation—EAP-TLS Helper

You use client certificate validation settings to specify how SBR Carrier performs CRL checking.

To configure client certification validation for the EAP-TLS helper protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TLS Helper** pane, click the **Client Certificate Validation** tab (Figure 98 on page 278).

Figure 98: EAP-TLS Helper—Client Certificate Validation

The screenshot shows the Steel-Belted Radius Carrier configuration interface. At the top, there is a navigation bar with links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. Below this is the 'EAP Methods List' section, which contains a table with columns for Name and Status. The table lists four methods: EAP-TLS, EAP-TTLS, EAP-PEAP, and EAP-TLS Helper, all of which are currently Disabled. The EAP-TLS Helper method is highlighted in blue. Below the table are buttons for Apply, Reset, and Refresh. The 'Selected EAP Method: EAP-TLS Helper' pane is open, showing the 'Client Certificate Validation' tab. This tab contains several configuration options: 'Enable EAP-TLS Helper Method' (unchecked), 'Enable CRL Checking' (unchecked), 'Retrieval Timeout' (set to 5), 'Expiration Grace Period' (set to 0), 'Allow Missing CDP Attribute' (checked), 'CRL Cache Timeout Period' (unchecked), and 'Default LDAP Server Name' (empty). At the bottom of the pane are buttons for Save, Reset, and Cancel.

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Selected EAP Method: EAP-TLS Helper

☐ Enable EAP-TLS Helper Method

Client Certificate Validation | Secondary Authorization | Session Resumption | Advanced Server Settings

Enable CRL Checking: ☐

Retrieval Timeout:

Expiration Grace Period:

Allow Missing CDP Attribute: ☒

CRL Cache Timeout Period: ☐

Default LDAP Server Name:

2. Select the **Enable CRL Checking** check box to enable CRL checking.
3. In the **Retrieval Timeout** field, enter the number of seconds you want the EAP-TLS helper to wait for a CRL retrieval transaction to complete.

When CRL retrieval takes longer than the specified time, the user's authentication request is rejected.
4. The **Expiration Grace Period** field contains the number of seconds during which an expired CRL may still be accepted.

The EAP-TLS helper always attempts to retrieve a new CRL when it is presented with a certificate chain and it finds an expired CRL in its cache.

- If you enter 0 to specify strict expiration mode, the EAP-TLS helper does not accept an expired CRL.
- If you enter a value greater than 0 (lax expiration mode), the EAP-TLS helper considers the expired CRL as an acceptable stand-in from the time the CRL expires to the time the grace period ends.

5. Select the **Allow Missing CDP Attribute** check box if you want SBR Carrier to accept a non-root certificate that does not have a CDP attribute.

Without a CDP attribute, the EAP-TLS helper cannot retrieve a CRL and cannot perform a revocation check on the certificate.

If you select the **Allow Missing CDP Attribute** check box, the EAP-TLS helper accepts such certificates and skips CRL checking for them.

If you clear the **Allow Missing CDP Attribute** check box, the EAP-TLS helper does not accept a CRL with a missing CDP attribute.

6. If you want to specify a CRL cache timeout period, select the **CRL Cache Timeout Period** check box and enter the number of hours in the **CRL Cache Timeout Period** field.

- If you do not enable this setting, the CRL is refreshed whenever it expires.
- If you enable this setting and enter 0, SBR Carrier always regards the CRL in the cache as expired and downloads a new CRL every time it receives a client certificate request.
- If you enable this setting and enter a number greater than 0, the CRL begins to expire when the age of the CRL in the cache exceeds the number of hours specified in this field or when the scheduled CRL expiration time occurs, whichever comes first.

After a CRL has expired because its scheduled expiration time has passed or because the CRL cache has timed out), SBR Carrier uses the expiration grace period to determine whether to use the current CRL.

7. If the CDP contains a value that begins with the string `//ldap:\\`, enter the name of the LDAP server to use in the **Default LDAP Server Name** field.

CDPs generated by some CAs do not include the identity of the LDAP server. If you expect to encounter certificates with this style CDP, specify the name of the LDAP server that contains the CRLs.

If you do not specify a server name and such certificates are encountered, the CRL retrieval fails.

Configuring Secondary Authentication—EAP-TLS Helper

You use secondary authorization settings to specify whether secondary authorization is performed and, if it is, what information is used in the secondary authorization request.

To configure secondary authentication for the EAP-TLS helper protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TLS Helper** pane, click the **Secondary Authorization** tab (Figure 99 on page 280).

Figure 99: EAP-TLS Helper—Secondary Authorization

The screenshot shows the Steel-Belted Radius Carrier web interface. At the top, there's a navigation bar with links: Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. Below this is a section titled "EAP Methods List" with buttons for Apply, Reset, and Refresh. It contains a table with columns "Name" and "Status".

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Below the table is a section titled "Selected EAP Method: EAP-TLS Helper". It has a checkbox for "Enable EAP-TLS Helper Method". Below this are four tabs: Client Certificate Validation, Secondary Authorization (selected), Session Resumption, and Advanced Server Settings.

Under the "Secondary Authorization" tab, there's a checkbox for "Enable Secondary Authorization". Below it are radio buttons for "User Name Source": Subject CN (selected), Principal Name, User-Name, and Calling-Station-Id. There's a "Fixed Password" field. Below that is an "Include Certificate" checkbox and an "Info" section. The "Info" section contains an "Inner Authentication" box with checkboxes for "Use Inner Radius", "Edit Inner Request" (with a dropdown), and "Edit Outer Response" (with a dropdown). Below these are "Profile Attribute" and "Realm" fields. At the bottom of the configuration pane are "Save", "Reset", and "Cancel" buttons.

2. Select the **Enable Secondary Authorization** check box to enable secondary authorization checking.

If secondary authorization is disabled, the EAP-TLS plug-in accepts the user upon proof of ownership of a private key that matches a valid certificate.

If secondary authorization is enabled, a secondary authorization check against a traditional authentication method such as an SQL plug-in is performed.

3. For **User Name Source**, select one of the following option buttons: **Subject CN name**, **Principal Name**, **User-Name**, and **Calling-Station-Id**. After the EAP-TLS module has concluded its processing, it may still defer to a traditional authentication method (core or plug-in) for final authorization. To do so, that method must provide a username and password to the traditional authentication method.

NOTE: User-Name and Calling-Station-Id are only available for inner authentication.

- If you select the **Subject CN** option button, the EAP-TLS module parses the Subject attribute of the client's certificate for the least significant 'CN=' and takes the value of this attribute (for example, 'George Washington') as the username being passed to the traditional authentication method.
 - If you select the **Principal Name** option button, the EAP-TLS module uses the principal name (subject alternate name or other name) from the client certificate (for example, joe@acme.com) as the username being passed to the traditional authentication method.
 - If you select the **User-Name** option button, the EAP-TLS module uses the RADIUS username to the traditional authentication method. This is available only for inner authentication.
 - If you select the **Calling-Station-Id** option button, the EAP-TLS module uses the calling station ID as the username being passed to the traditional authentication method. This is available only for inner authentication.
4. If you plan to use secondary authorization against an authentication method (for example, LDAP) that cannot be configured to ignore the lack of user credentials, specify a fixed password that the plug-in uses on all secondary authorization checks in the **Fixed Password** field.

By default, the secondary authorization check includes a username but no other user credentials because no password or similar credential for the client is available at the conclusion of the TLS handshake. Some authentication methods (Native User, LDAP, and SQL) can be configured to not require user credentials.

5. If you want the EAP-TLS plug-in to add four attributes to the request before the secondary authorization check is performed, select the **Include Certificate Info** check box.

When the **Include Certificate Info** check box is selected, SBR Carrier adds the following attributes to the request:

- The **Funk-Peer-Cert-Subject** attribute contains the value of the Subject attribute in the client certificate.
- The **Funk-Peer-Cert-Principal** attribute contains the value of the principal name (subject alternate name or other name) attribute of the client certificate.
- The **Funk-Peer-Cert-Issuer** attribute contains the value of the Issuer attribute in the client certificate.
- The **Funk-Peer-Cert-Hash** attribute contains a hexadecimal ASCII representation of the SHA1 hash of the client certificate.

These attributes are ignored if the authentication method that performs the authentication check does not use them.

6. You can use the inner authentication settings to specify the way in which the inner authentication process operates.

NOTE: The inner authentication settings is applicable only for PAP authentication.

To configure inner authentication settings:

- Select the **Use Inner Radius** check box to enable inner authentication when secondary authorization is enabled.
- Select the **Edit Inner Request** check box and select the filter you want to use from the drop-down list.

This filter affects the inner authentication request. The filter can be used to modify attributes to influence routing of the inner authentication through editing an attribute and selecting a realm.

- Select the **Edit Outer Response** check box and select the filter you want to use from the drop-down list.

This filter is used to edit attributes in the inner authentication response.

- If this filter is specified, the filter is applied to the inner authentication response and all resulting attributes are transferred to the outer authentication response.
- If this filter is not specified, no inner authentication response attributes are transferred to the outer authentication response.
- Enter the attribute name that contains the profile name in the **Profile Attribute** field.

This profile name is present in the attribute returned from the inner authentication response. If the profile name is not available in the SBR, an Access-Reject message is sent.

- Enter the directed or proxy realm name in the **Realm** field.

If a realm name is configured in the SBR, all inner authentications are forwarded to the realm. If a realm name is not configured, then standard authentication takes place as defined in the proxy.ini file.

Configuring Session Resumption—EAP-TLS Helper

You use session resumption settings to specify under what circumstances session resumption is performed.

NOTE: For session resumption to work, the NAD must be configured to handle the **Session-Timeout** return list attribute, so that the NAD can notify the client to reauthenticate after the session timer has expired.

To configure session resumption for the EAP-TLS helper protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TLS Helper** pane, click the **Session Resumption** tab (Figure 100 on page 283).

Figure 100: EAP-TLS Helper—Session Resumption

The screenshot shows the Steel-Belted Radius Carrier configuration interface. At the top, there is a navigation bar with links: Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. Below this is the 'EAP Methods List' section, which contains a table with columns 'Name' and 'Status'. The table lists four methods: EAP-TLS, EAP-TTLS, EAP-PEAP, and EAP-TLS Helper, all with a status of 'Disabled'. The 'EAP-TLS Helper' method is highlighted. Below the table are buttons for 'Apply', 'Reset', and 'Refresh'. The 'Selected EAP Method: EAP-TLS Helper' section is open, showing a checkbox for 'Enable EAP-TLS Helper Method' which is unchecked. Below this are four tabs: 'Client Certificate Validation', 'Secondary Authorization', 'Session Resumption', and 'Advanced Server Settings'. The 'Session Resumption' tab is selected, displaying three fields: 'Session Timeout(In Seconds):', 'Termination Action:', and 'Resumption Limit(In Seconds):'. Each field has a numeric input box with a value of '0' and a dropdown arrow. At the bottom of this section are buttons for 'Save', 'Reset', and 'Cancel'.

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Selected EAP Method: EAP-TLS Helper

☐ Enable EAP-TLS Helper Method

Client Certificate Validation **Secondary Authorization** **Session Resumption** **Advanced Server Settings**

Session Timeout(In Seconds): 0

Termination Action: 0

Resumption Limit(In Seconds): 0

Save **Reset** **Cancel**

2. In the **Session Timeout(In Seconds)** field, enter the number of seconds the client may remain connected to the NAD before having to reauthenticate.

If you enter a number greater than 0, the lesser of this value and the remaining resumption limit is sent in a **Session-Timeout** attribute to the RADIUS client on the RADIUS Access-Accept response.

If you enter 0, a **Session-Timeout** attribute is not generated. This does not prevent the authentication methods performing secondary authorization from providing a value for this attribute.

Entering a value such as 600 seconds (10 minutes) does not necessarily cause a full reauthentication to occur every 10 minutes. You can configure the resumption limit to make most reauthentications fast and computationally efficient.

3. Enter the value to be returned in a **Termination-Action** attribute in the **Termination Action** field.

The **Termination-Action** attribute is a standard attribute supported by most access points and determines what happens when the session timeout is reached. Valid values are:

- -1: Do not send the attribute; the default value. This does not prevent any authentication method that performs secondary authorization from providing a value for this attribute.
 - 0: Send the **Termination-Action** attribute with a value of 0.
 - 1: Send the **Termination-Action** attribute with a value of 1.
4. Enter the maximum number of seconds you want the client to be able to reauthenticate using the TLS session resumption feature in the **Resumption Limit(In Seconds)** field.

This type of reauthentication is fast and computationally efficient. It does, however, depend on previous authentications and is not as secure as a complete (computationally expensive) authentication. Specifying a value of 0 disables the session resumption feature.

BEST PRACTICE: Using the Resumption Limit Option Effectively

Two scenarios where the resumption limit can be used effectively:

- In a wireless environment, the client is moving between access points. The resumption limit can be tuned to make the handover between access points smoother by not forcing a complete reauthorization that requires repeated verification of user information.

When the new access point queries SBR Carrier, the server replies that the session ID is already valid. Because it is known to be good, repeating the inner authentication is not required, which saves some time. The access point acknowledges the *reauthorization not required* message and the session continues.

- Another use for resumption limit occurs when the server ordinarily requires the client to reauthorize every 10 minutes or so, to ensure the client is still connected. Setting the resumption limit to 3600 seconds with a session timeout of 600 seconds means that the interval reauthorizations are fast and efficient, and a complete reauthorization is required just once an hour instead of every 10 minutes.

Configuring Advanced Server Settings—EAP-TLS Helper

You use advanced server settings to specify the manner in which the inner authentication step operates.

To configure advanced server settings for the EAP-TLS helper protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TLS Helper** pane, click the **Advanced Server Settings** tab (Figure 101 on page 285).

Figure 101: EAP-TLS Helper—Advanced Server Settings

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes 'Home', 'RADIUS Configuration', 'Tools', 'Help', 'Logout', and 'User: root'. The main content area is titled 'EAP Methods List' and contains a table with the following data:

Name	Status
EAP-PEAP	Disabled
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-TLS Helper	Disabled

Below the table, the 'Selected EAP Method: EAP-TLS Helper' pane is shown. It includes a checkbox for 'Enable EAP-TLS Helper Method' and four tabs: 'Client Certificate Validation', 'Secondary Authorization', 'Session Resumption', and 'Advanced Server Settings'. The 'Advanced Server Settings' tab is active, displaying the following configuration fields:

- TLS Message Fragment Length: 1020
- Max Transaction Time: 120
- Challenge Timeout: 30
- Return MPPE Keys: ☒
- TLS Protocol Version: TLSv1
- DH Prime Bits: 1024
- Cipher Suites: 0x003C,0x003D,0x0067,C

At the bottom of the pane are buttons for 'Save', 'Reset', and 'Cancel'.

2. In the **TLS Message Fragment Length** field, enter the maximum length of the TLS message that may be generated during each iteration of the TLS exchange.

Enter a number in the range 500 through 4096 bytes. This value affects the number of RADIUS challenge or response round-trips required to conclude the TLS exchange.

Some access points may have problems with RADIUS responses or EAP messages that exceed the size of one Ethernet frame (1500 bytes including IP/UDP headers).

The default length for TLS messages is 1020 bytes, which prevents the RADIUS challenge response (carried in a UDP packet) from exceeding one Ethernet frame.
3. In the **Max Transaction Time** field, enter the maximum number of seconds you want for the EAP-TLS helper authentication sequence to complete.

- Enter a value in the range 1 through 3600 seconds. The default value is 120 seconds.
 - If the authentication sequence takes longer than this setting, user authentication is terminated.
4. In the **Challenge Timeout** field, enter the number of seconds after which a challenge request times out.
You can enter a value greater than or equal to 1 second, but this value must not exceed the value specified in the **Max Transaction Time** field. The default value is 30 seconds.
 5. Select the **Return MPPE Keys** check box to specify whether the EAP-TLS helper includes RADIUS **MS-MPPE-Send-Key** and **MS-MPPE-Recv-Key** attributes in the final RADIUS Access-Accept response sent to the access point.

Enable this option if the access point needs to key the WEP encryption. If the access point is authenticating only end users and WEP is not being used, you can clear this check box.
 6. Use the **TLS Protocol Version** list to specify the TLS protocol version on which the server expects the client to initiate the handshake process.

Valid values are TLSv1, TLSv1.1, and TLSv1.2.
 7. Use the **DH Prime Bits** list to specify the number of bits in the prime number that the module uses for Diffie-Hellman exponentiation.

Selecting a longer prime number makes the system less susceptible to certain types of attacks but requires more CPU processing to compute the Diffie-Hellman key agreement operation.

Valid values are 512, 1024, 1536, 2048, 3072, and 4096 bits.
 8. In the **Cipher Suites** field, enter the TLS cipher suites (in order of preference) that the server is to use.

These cipher suites are documented in RFC 2246, *The TLS Protocol Version 1*, RFC 4346, *The TLS Protocol Version 1.1*, and RFC 5246, *The TLS Protocol Version 1.2*.

The default value is: 0x0067,0x006B,0xC030,0xC028,0xC014,0xC013.

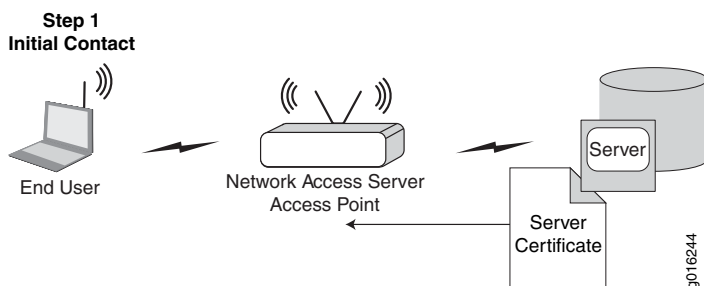
See [Table 36 on page 273](#) for the list of tested cipher suites and their TLS protocol versions.

EAP-TTLS Authentication Protocol

EAP-TTLS (Tunneled Transport Layer Security) is designed to provide authentication that is as strong as EAP-TLS, but it does not require that each user be issued a certificate. Instead, only the authentication servers are issued certificates. User authentication is performed by password, but the password credentials are transported in a securely encrypted *tunnel* established based upon the server certificates.

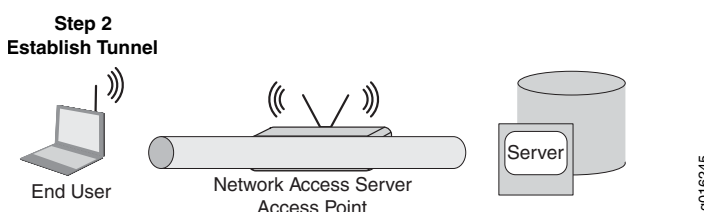
1. After the authentication server determines that the user has made an authentication request, it sends its certificate to the user's system ([Figure 102 on page 287](#)).

Figure 102: TTLS Server Certificate Sent to NAD



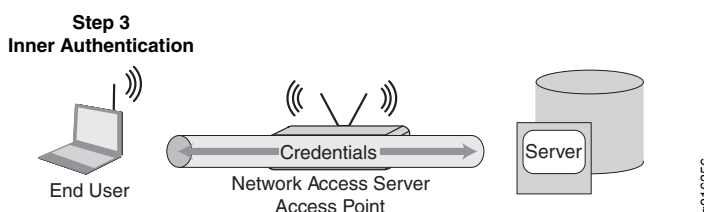
2. The authentication server's certificate is used to establish a tunnel between the user and the server ([Figure 103 on page 287](#)).

Figure 103: TTLS Tunnel Established



3. After the tunnel is established, credentials can be exchanged safely between the server and the user because tunnels encrypt all data in a secure fashion. This stage is called *inner authentication* ([Figure 104 on page 287](#)).

Figure 104: TTLS Inner Authentication



With EAP-TTLS, you do not need to create a new infrastructure of user certificates. User authentication is performed against the same security database that is already in use on the corporate LAN; for example, SQL or LDAP databases.

The routing of the inner authentication request is handled either by means of standard SBR Carrier authentication request routing, or by means of a directed realm. If your EAP-TTLS tunnel ends at a dedicated server, and you want all the inner authentication requests to be performed by other servers, use standard request routing so the proxy realm target can be determined in a standard fashion (that is, the decoration of the username revealed by inner authentication). If your EAP-TTLS tunnel and inner authentication are handled by the same server, you can use a directed realm to specify which authentication methods handle the inner authentication.

Configuring EAP-TTLS as an EAP Authentication Method

To configure EAP-TTLS using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > EAP Methods**.

The **EAP Methods List** page (Figure 92 on page 264) appears.

2. Select **EAP-TTLS**.

The **Selected EAP Method: EAP-TTLS** pane (Figure 105 on page 288) appears.

Figure 105: Selected EAP Method: EAP-TTLS Pane

The screenshot shows the Juniper Steel-Belted Radius Carrier Web GUI. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. The main content area is divided into two panes. The top pane, titled 'EAP Methods List', contains a table with columns 'Name' and 'Status'. The bottom pane, titled 'Selected EAP Method: EAP-TTLS', contains a section for 'Enable EAP-TTLS Method' and a tabbed interface for configuration. The 'Client Certificate Validation' tab is active, showing various settings for certificate validation.

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Selected EAP Method: EAP-TTLS

☐ Enable EAP-TTLS Method

Client Certificate Validation | Request Filters | Response Filters | Session Resumption | Inner Authentication | Advanced Server Settings

Enable CRL Checking: ☐

Require Client Certificate: ☐

Retrieval Timeout: 5

Expiration Grace Period: 0

Allow Missing CDP Attribute: ☒

CRL Cache Timeout Period: ☐ 168

Default LDAP Server Name:

Include Certificate Info: ☐

Save | Reset | Cancel

3. Select the **Enable EAP-TTLS Method** check box to enable the EAP-TTLS method.

NOTE: You can also enable the EAP-TTLS method by using the **EAP Methods List** page. In the **EAP Methods List** page, click the **Status** column of the **EAP-TTLS** entry, select the appeared check box, and click **Apply**.

4. Configure client certificate validation for the EAP-TTLS protocol. For more information about configuring client certificate validation, see [“Configuring Client Certificate Validation—EAP-TTLS” on page 289](#).
5. Configure request filters for the EAP-TTLS protocol. For more information about configuring request filters, see [“Configuring Request Filters—EAP-TTLS” on page 292](#).
6. Configure response filters for the EAP-TTLS protocol. For more information about configuring response filters, see [“Configuring Response Filters—EAP-TTLS” on page 294](#).
7. Configure session resumption for the EAP-TTLS protocol. For more information about configuring session resumption, see [“Configuring Session Resumption—EAP-TTLS” on page 296](#).
8. Configure inner authentication for the EAP-TTLS protocol. For more information about configuring inner authentication, see [“Configuring Inner Authentication Settings—EAP-TTLS” on page 298](#).
9. Configure advanced server settings for the EAP-TTLS protocol. For more information about configuring advanced server settings, see [“Configuring Advanced Server Settings—EAP-TTLS” on page 299](#).
10. Click **Save** to save the configuration.

Configuring Client Certificate Validation—EAP-TTLS

You use client certificate validation settings to specify how SBR Carrier performs CRL checking.

To configure client certificate validation for the EAP-TTLS protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TTLS** pane, click the **Client Certificate Validation** tab (Figure 106 on page 290).

Figure 106: EAP-TTLS—Client Certificate Validation

The screenshot shows the Steel-Belted Radius Carrier configuration interface. At the top, there is a navigation bar with links: Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. Below this is the 'EAP Methods List' section, which contains a table with columns 'Name' and 'Status'. The table lists five methods: EAP-TLS, EAP-TTLS (highlighted), EAP-PEAP, and EAP-TLS Helper, all with a status of 'Disabled'. Below the table is the 'Selected EAP Method: EAP-TTLS' section. It starts with a checkbox 'Enable EAP-TTLS Method'. Below this is a tabbed interface with six tabs: 'Client Certificate Validation' (selected), 'Request Filters', 'Response Filters', 'Session Resumption', 'Inner Authentication', and 'Advanced Server Settings'. The 'Client Certificate Validation' tab contains several configuration options: 'Enable CRL Checking' (checkbox), 'Require Client Certificate' (checkbox), 'Retrieval Timeout' (spinner set to 5), 'Expiration Grace Period' (spinner set to 0), 'Allow Missing CDP Attribute' (checkbox checked), 'CRL Cache Timeout Period' (checkbox and spinner set to 168), 'Default LDAP Server Name' (text field), and 'Include Certificate Info' (checkbox). At the bottom of the configuration pane are 'Save', 'Reset', and 'Cancel' buttons.

2. Select the **Enable CRL Checking** check box to enable CRL checking.
3. If you want to require the client to provide a certificate as part of the TTLS exchange, select the **Require Client Certificate** check box.

NOTE: To validate client certificates, enable both the **Enable CRL Checking** and **Require Client Certificate** check boxes.

4. In the **Retrieval Timeout** field, enter the number of seconds you want EAP-TTLS to wait for a CRL checking transaction to complete, when the CRL check involves a CRL retrieval.

When CRL retrieval takes longer than the specified time, the user's authentication request results in a reject.

5. Enter the number of seconds during which a CRL is still considered acceptable after it has expired in the **Expiration Grace Period** field.

EAP-TTLS always attempts to retrieve a new CRL when it is presented with a certificate chain and it finds an expired CRL in its cache.

- If you enter 0 (strict expiration mode), EAP-TTLS does not accept a CRL that has expired.
- If you enter a value greater than 0 (lax expiration mode), EAP-TTLS considers the expired CRL as an acceptable stand-in from the time the CRL expires to the time the grace period ends.

6. Select the **Allow Missing CDP Attribute** check box if you want SBR Carrier to accept a non-root certificate that does not have a CDP attribute.

Without a CDP attribute, EAP-TTLS cannot retrieve a CRL and cannot perform a revocation check on the certificate.

- If you select the **Allow Missing CDP Attribute** check box, EAP-TTLS accepts such certificates and skips CRL checking for them.
- If you clear the **Allow Missing CDP Attribute** check box, EAP-TTLS does not accept a CRL with a missing CDP attribute.

7. If you want to specify a CRL cache timeout period, select the **CRL Cache Timeout Period** check box and enter the number of hours in the **CRL Cache Timeout Period** field.

- If you do not enable this setting, the CRL is refreshed whenever it expires.
- If you enable this setting and enter 0, SBR Carrier always regards the CRL in the cache as expired and downloads a new CRL every time it receives a client certificate request.
- If you enable this setting and enter a number greater than 0, the CRL begins to expire when the age of the CRL in the cache exceeds the number of hours specified in this field or when the scheduled CRL expiration time occurs, whichever comes first.

After a CRL has expired (because its scheduled expiration time has passed or because the CRL cache has timed out), SBR Carrier uses the expiration grace period to determine whether to use the current CRL.

8. In the **Default LDAP Server Name** field, enter the name of the LDAP server to use if the CDP contains a value that begins with the string `//ldap:\\`.

CDPs generated by some CAs do not include the identity of the LDAP server. If you expect to encounter certificates with this style CDP, specify the name of the LDAP server that contains the CRLs.

If you do not specify a server name and such certificates are encountered, the CRL retrieval fails.

9. If you want the EAP-TTLS plug-in to add four attributes to the outer request before the secondary authorization check is performed, select the **Include Certificate Info** check box.

When the **Include Certificate Info** check box is selected, SBR Carrier adds the following attributes to the request:

- The **Funk-Peer-Cert-Subject** attribute contains the value of the Subject attribute in the client certificate.
- The **Funk-Peer-Cert-Principal** attribute contains the value of the principal name (Subject Alternate Name or Other Name) attribute of the client certificate.
- The **Funk-Peer-Cert-Issuer** attribute contains the value of the Issuer attribute in the client certificate.
- The **Funk-Peer-Cert-Hash** attribute contains a hexadecimal ASCII representation of the SHA1 hash of the client certificate.

If these attributes are to be used in the inner authentication, they must be transferred from the outer request by using a filter (see [“Configuring Request Filters—EAP-TTLS” on page 292](#)). These attributes are ignored if the authentication method that performs the authentication check does not use them. The attributes are available in Realm Selection for EAP-TTLS inner authentication, and proxy filtering from EAP-TTLS inner authentication.

Special-purpose VSAs carry the certificate information and must be filtered into the inner authentication method in order to be made available for validation. For example, these lines could be added to the ttls transfer filter:

```
Allow Funk-Peer-Cert-Subject
Allow Funk-Peer-Cert-Principal
Allow Funk-Peer-Cert-Hash
Allow Funk-Peer-Cert-Issuer
```

Configuring Request Filters—EAP-TTLS

Request filters affect the attributes of inner authentication requests. By default, SBR Carrier does not use request filters.

NOTE: You must configure filters before you can associate them with an authentication method. For information about configuring filters, refer to [“Setting Up Filters” on page 217](#).

To configure request filtering for the EAP-TTLS protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TTLS** pane, click the **Request Filters** tab (Figure 107 on page 293).

Figure 107: EAP-TTLS—Request Filters

The screenshot shows the Steel-Belted Radius Carrier Web GUI. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. The main content area is titled "EAP Methods List" and contains a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Below the table is the "Selected EAP Method: EAP-TTLS" pane. It features a checkbox for "Enable EAP-TTLS Method" and a tabbed interface with the following tabs: Client Certificate Validation, Request Filters (selected), Response Filters, Session Resumption, Inner Authentication, and Advanced Server Settings. The "Request Filters" tab contains the following configuration options:

- Transfer Outer Attribs to New: ☐ [Dropdown]
- Transfer Outer Attribs to continue: ☐ [Dropdown]
- Edit New: ☐ [Dropdown]
- Edit Continue: ☐ [Dropdown]

At the bottom of the pane are buttons for Save, Reset, and Cancel.

2. Optionally, select the **Transfer Outer Attribs to New** check box and select the filter you want to use from the **Transfer Outer Attribs to New** list.

This filter affects only a new inner authentication request.

- If this filter is specified, all attributes from the outer request are transferred to the inner request and this filter is applied. The transfer occurs and the filter is applied before any attributes specified in the inner authentication are added to the request.
- If this filter is not specified, no attributes from the outer request are transferred to the inner request.

3. Optionally, select the **Transfer Outer Attrs to continue** check box and select the filter you want to use from the **Transfer Outer Attrs to continue** list.

This filter affects only a continued inner authentication request (rather than the first inner authentication request discussed in the previous step). If this filter is specified, all attributes from the outer request are transferred to the inner request and this filter is applied. The transfer occurs and the filter is applied before any attributes specified in the inner authentication are added to the request.

If this filter is not specified, no attributes from the outer request are transferred to the inner request.

4. Optionally, select the **Edit New** check box and select the filter you want to use from the **Edit New** list.

This filter affects only a new inner authentication request. If this filter is specified, it is applied to the inner request that is the cumulative result of attributes transferred from the outer request (by the filter specified in Step 2) and attributes included in the inner authentication request sent through the tunnel by the client.

If this filter is not specified, the request remains unaltered.

5. Optionally, select the **Edit Continue** check box and select the filter you want to use from the **Edit Continue** list.

This filter affects only a continued inner authentication request (rather than a new inner authentication request). If this filter is specified, it is applied to the inner request that is the cumulative result of attributes transferred from the outer request (by the filter specified in Step 3) and attributes included in the inner authentication request sent through the tunnel by the client.

If this filter is not specified, the request remains unaltered.

Configuring Response Filters—EAP-TTLS

Response filters affect the attributes in the final response (Access-Accept or Access-Reject) returned to the originating NAD. By default, SBR Carrier does not use response filters.

To configure response filtering for the EAP-TTLS protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TTLS** pane, click the **Response Filters** tab (Figure 108 on page 295).

Figure 108: EAP-TTLS—Response Filters

The screenshot shows the Steel-Belted Radius Carrier configuration page. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. The main content area is titled "EAP Methods List" and contains a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Below the table, the "Selected EAP Method: EAP-TTLS" section is active. It includes a checkbox for "Enable EAP-TTLS Method" and a tabbed interface with the following tabs: Client Certificate Validation, Request Filters, Response Filters, Session Resumption, Inner Authentication, and Advanced Server Settings. The "Response Filters" tab is selected, showing two sections:

- Transfer Inner Attrs To Accept:** A checkbox and a dropdown menu.
- Transfer Inner Attrs To Reject:** A checkbox and a dropdown menu.

At the bottom of the configuration area are buttons for Save, Reset, and Cancel.

2. Optionally, select the **Transfer Inner Attrs To Accept** check box and select the filter you want to use from the **Transfer Inner Attrs To Accept** list.

This filter affects only an outer Access-Accept response that is sent back to a NAD.

- If this filter is specified, the filter is applied to the inner authentication response and all resulting attributes are transferred to the outer authentication response.
- If this filter is not specified, no inner authentication response attributes are transferred to the outer authentication response.

3. Optionally, select the **Transfer Inner Attrs To Reject** check box and select the filter you want to use from the **Transfer Inner Attrs To Reject** list.

This filter affects only a continued inner authentication request (rather than the first inner authentication request). If this filter is specified, all attributes from the outer request are transferred to the inner

request and this filter is applied. The transfer occurs and the filter is applied before any attributes specified in the inner authentication are added to the request.

If this filter is not specified, no attributes from the outer request are transferred to the inner request.

Configuring Session Resumption—EAP-TTLS

You use session resumption settings to specify whether session resumption is permitted and under what circumstances session resumption is performed.

NOTE: For session resumption to work, the NAD must be configured to handle the **Session-Timeout** return list attribute, so that the NAD can notify the client to reauthenticate after the session timer has expired.

To configure session resumption for the EAP-TTLS protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TTLS** pane, click the **Session Resumption** tab (Figure 109 on page 297).

Figure 109: EAP-TTLS—Session Resumption

The screenshot shows the Steel-Belted Radius Carrier configuration interface. At the top, there is a navigation bar with links: Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. Below this is the 'EAP Methods List' section, which contains a table with columns 'Name' and 'Status'. The table lists five EAP methods: EAP-TLS, EAP-TTLS (highlighted in blue), EAP-PEAP, and EAP-TLS Helper, all with a status of 'Disabled'. Below the table are buttons for Apply, Reset, and Refresh. The main section is titled 'Selected EAP Method: EAP-TTLS'. It features a checkbox for 'Enable EAP-TTLS Method' and a tabbed interface with six tabs: Client Certificate Validation, Request Filters, Response Filters, Session Resumption (selected), Inner Authentication, and Advanced Server Settings. The 'Session Resumption' tab contains three input fields: 'Session Timeout(In Seconds):' with a value of 0, 'Termination Action:' with a value of 0, and 'Resumption Limit(In Seconds):' with a value of 0. At the bottom of this section are buttons for Save, Reset, and Cancel.

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Selected EAP Method: EAP-TTLS

☐ Enable EAP-TTLS Method

Client Certificate Validation | Request Filters | Response Filters | **Session Resumption** | Inner Authentication | Advanced Server Settings

Session Timeout(In Seconds): 0

Termination Action: 0

Resumption Limit(In Seconds): 0

Save | Reset | Cancel

2. In the **Session Timeout(In Seconds)** field, enter the maximum number of seconds you want the client to remain connected to the NAD before having to reauthenticate.
 - If you enter 0, no **Session-Limit** attribute is generated. This does not prevent the authentication methods performing secondary authorization from providing a value for this attribute.
 - If you enter a number greater than 0, the lesser of this value and the remaining resumption limit is sent in a **Session-Limit** attribute to the RADIUS client on the RADIUS Access-Accept response.

Entering a value such as 600 seconds (10 minutes) does not necessarily cause a full reauthentication to occur every 10 minutes. You can configure the resumption limit to make most reauthentications fast and computationally efficient.

3. Enter the value to be returned in a **Termination-Action** attribute in the **Termination Action** field.

The **Termination-Action** attribute is a standard attribute supported by most access points and determines what happens when the session timeout is reached. Valid values are:

- -1: Do not send the attribute; the default value. This does not prevent any authentication method that performs secondary authorization from providing a value for this attribute.
- 0: Send the **Termination-Action** attribute with a value of 0, to terminate the session.
- 1: Send the **Termination-Action** attribute with a value of 1, which forces reauthentication when the Session-Time expires.

4. In the **Resumption Limit(In Seconds)** field, enter the maximum number of seconds you want the client to be able to reauthenticate using the TTLS session resumption feature.

This type of reauthentication is fast and computationally efficient. It does, however, depend on previous authentications and is not as secure as a complete (computationally expensive) authentication. Specifying a value of 0 disables the session resumption feature.

BEST PRACTICE: Using the Resumption Limit Option Effectively

Two scenarios where the resumption limit can be used effectively:

- In a wireless environment, the client is moving between access points. The resumption limit can be tuned to make the handover between access points smoother by not forcing a complete reauthorization that requires repeated verification of user information.

When the new access point queries SBR Carrier, the server replies that the session ID is already valid. Because it is known to be good, repeating the inner authentication is not required, which saves some time. The access point acknowledges the *reauthorization not required* message and the session continues.

- Another use for resumption limit occurs when the server ordinarily requires the client to reauthorize every 10 minutes or so, to ensure the client is still connected. Setting the resumption limit to 3600 seconds with a session timeout of 600 seconds means that the interval reauthorizations are fast and efficient, and a complete reauthorization is required just once an hour instead of every 10 minutes.

Configuring Inner Authentication Settings—EAP-TTLS

You use the inner authentication settings to specify the way in which the inner authentication step operates.

To configure inner authentication settings for the EAP-TTLS protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TTLS** pane, click the **Inner Authentication** tab ([Figure 110 on page 299](#)).

Figure 110: EAP-TTLS—Inner Authentication

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. The main content area is titled "EAP Methods List" and contains a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Below the table, the "Selected EAP Method: EAP-TTLS" configuration panel is shown. It includes a checkbox for "Enable EAP-TTLS Method" and a tabbed interface with the following tabs: Client Certificate Validation, Request Filters, Response Filters, Session Resumption, Inner Authentication (selected), and Advanced Server Settings. The Inner Authentication tab contains the following fields:

- Directed Realm:
- Realm Selection Script:

At the bottom of the configuration panel are buttons for Save, Reset, and Cancel.

2. If you want requests to be routed based on the methods listed in the directed realm, enter the name of a directed realm in the **Directed Realm** field.

Omitting this setting causes the inner authentication request to be handled like any other request received from a NAD.

3. If you want requests to be processed by means of a realm selection script, enter the name of a script in the **Realm Selection Script** field.

You must license the optional JavaScripting module to use realm selection scripts. For information about realm selection scripting, refer [“Creating Realm Selection Scripts” on page 895](#).

Configuring Advanced Server Settings—EAP-TTLS

You use advanced server settings to specify the manner in which the inner authentication step operates.

To configure advanced server settings for the EAP-TTLS protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-TTLS** pane, click the **Advanced Server Settings** tab (Figure 111 on page 300).

Figure 111: EAP-TTLS—Advanced Server Settings

The screenshot displays the Steel-Belted Radius Carrier Web GUI. The top navigation bar includes 'Home', 'RADIUS Configuration', 'Tools', 'Help', 'Logout', and 'User: root'. The main content area is titled 'EAP Methods List' and contains a table with the following data:

Name	Status
EAP-PEAP	Disabled
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-TLS Helper	Disabled

Below the table, the 'Selected EAP Method: EAP-TTLS' pane is shown. It includes a checkbox for 'Enable EAP-TTLS Method' and a series of tabs: 'Client Certificate Validation', 'Request Filters', 'Response Filters', 'Session Resumption', 'Inner Authentication', and 'Advanced Server Settings'. The 'Advanced Server Settings' tab is active, displaying the following configuration fields:

- TLS Message Fragment Length: 1020
- Max Transaction Time: 120
- Challenge Timeout: 30
- Return MPPE Keys: ☒
- TLS Protocol Version: TLSv1
- DH Prime Bits: 1024
- Cipher Suites: 0x003C,0x003D,0x0067,C

At the bottom of the pane are 'Save', 'Reset', and 'Cancel' buttons.

2. In the **TLS Message Fragment Length** field, enter the maximum length of the TLS message that may be generated during each iteration of the TLS exchange.

Enter a number in the range 500 through 4096 bytes. This value affects the number of RADIUS challenge/response round-trips required to conclude the TLS exchange. A value of 1400 bytes may result in 6 round-trips, while a value of 500 bytes may result in 15 round-trips.

Some access points may have problems with RADIUS responses or EAP messages that exceed the size of one Ethernet frame (1500 bytes including IP/UDP headers).

The default length for TLS messages is 1020 bytes, which prevents the RADIUS challenge response (carried in a UDP packet) from exceeding one Ethernet frame.

3. In the **Max Transaction Time** field, enter the maximum number of seconds you want for the EAP-TTLS authentication sequence to complete.

- Enter a value in the range 1 through 3600 seconds. The default value is 120 seconds.
- If the authentication sequence takes longer than this setting, user authentication is terminated.

4. In the **Challenge Timeout** field, enter the number of seconds after which a challenge request times out.

You can enter a value greater than or equal to 1 second, but this value must not exceed the value specified in the **Max Transaction Time** field. The default value is 30 seconds.

5. Select the **Return MPPE Keys** check box to specify whether the TTLS authentication method includes RADIUS **MS-MPPE-Send-Key** and **MS-MPPE-Recv-Key** attributes in the final RADIUS Access-Accept response sent to the access point.

Select this check box if the access point needs to key the WEP encryption. If the access point is authenticating only end users and WEP is not being used, you can clear this check box. Disable this option for WiMAX.

6. Use the **TLS Protocol Version** list to specify the TLS protocol version on which the server expects the client to initiate the handshake process.

Valid values are TLSv1, TLSv1.1, and TLSv1.2.

7. Use the **DH Prime Bits** list to specify the number of bits in the prime number that the module uses for Diffie-Hellman exponentiation.

Selecting a longer prime number makes the system less susceptible to certain types of attacks but requires more CPU processing to compute the Diffie-Hellman key agreement operation.

Valid values are 512, 1024, 1536, 2048, 3072, and 4096 bits.

8. In the **Cipher Suites** field, enter the TLS cipher suites (in order of preference) that the server is to use.

These cipher suites are documented in RFC 2246, *The TLS Protocol Version 1*, RFC 4346, *The TLS Protocol Version 1.1*, and RFC 5246, *The TLS Protocol Version 1.2*.

The default value is: 0x0067,0x006B,0xC030,0xC028,0xC014,0xC013.

See [Table 36 on page 273](#) for the list of tested cipher suites and their TLS protocol versions.

EAP-PEAP Authentication Protocol

The EAP-PEAP (Protected EAP) protocol is similar to EAP-TTLS. Unlike EAP-TTLS, which can tunnel any kind of authentication request (such as PAP or CHAP) and extended attributes, PEAP can tunnel only other EAP protocols inside its connection.

EAP-PEAP works in two phases:

- In Phase 1, the client authenticates the server and uses a TLS handshake to create an encrypted tunnel.
- In Phase 2, the server authenticates the user or machine credentials using an EAP authentication protocol. The EAP authentication is protected by the encrypted tunnel created in Phase 1. The authentication type negotiated during Phase 2 can be any valid EAP type, such as MS-CHAPv2.

Configuring EAP-PEAP as an EAP Authentication Method

NOTE: A valid server certificate must be in place on the SBR Carrier server before you configure the EAP-PEAP authentication protocol. For information about configuring certificates, see [“Certificates” on page 238](#).

To configure EAP-PEAP on a SBR Carrier server using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > EAP Methods**.

The **EAP Methods List** page ([Figure 92 on page 264](#)) appears.

2. Select **EAP-PEAP**.

The **Selected EAP Method: EAP-PEAP** pane ([Figure 112 on page 303](#)) appears.

Figure 112: Selected EAP Method: EAP-PEAP Pane

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. The main content area is divided into two panes. The top pane, titled "EAP Methods List", contains buttons for Apply, Reset, and Refresh, and a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

The bottom pane, titled "Selected EAP Method: EAP-PEAP", features a checkbox for "Enable EAP-PEAP Method" and a tabbed interface with the following tabs: Request Filters, Response Filters, Session Resumption, Inner Authentication, and Advanced Server Settings. The "Request Filters" tab is active, showing four rows of configuration options, each with a checkbox and a dropdown menu:

Request Filters	Response Filters	Session Resumption	Inner Authentication	Advanced Server Settings
Transfer Outer Attribs to New:	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Transfer Outer Attribs to continue:	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Edit New:	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Edit Continue:	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

At the bottom of the pane are buttons for Save, Reset, and Cancel.

3. Select the **Enable EAP-PEAP Method** check box to enable the EAP-PEAP method.

NOTE: You can also enable the EAP-PEAP method by using the **EAP Methods List** page. In the **EAP Methods List** page, click the **Status** column of the **EAP-PEAP** entry, select the appeared check box, and click **Apply**.

4. Configure request filters for the EAP-PEAP protocol. For more information about configuring request filters, see [“Configuring Request Filters—EAP-PEAP” on page 304](#).
5. Configure response filters for the EAP-PEAP protocol. For more information about configuring response filters, see [“Configuring Response Filters—EAP-PEAP” on page 306](#).

6. Configure session resumption for the EAP-PEAP protocol. For more information about configuring session resumption, see [“Configuring Session Resumption—EAP-PEAP” on page 308](#).
7. Configure inner authentication for the EAP-PEAP protocol. For more information about configuring inner authentication, see [“Configuring Inner Authentication Settings—EAP-PEAP” on page 310](#).
8. Configure advanced server settings for the EAP-PEAP protocol. For more information about configuring advanced server settings, see [“Configuring Advanced Server Settings—EAP-PEAP” on page 312](#).
9. Click **Save** to save the configuration.

Configuring Request Filters—EAP-PEAP

Request filters affect the attributes of inner authentication requests. By default, SBR Carrier does not use request filters.

NOTE: You must configure filters using the **Filters** page before you can associate them with the EAP-PEAP authentication method. For information about configuring filters, refer to [“Setting Up Filters” on page 217](#).

To configure request filtering for the EAP-PEAP protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-PEAP** pane, click the **Request Filters** tab (Figure 113 on page 305).

Figure 113: EAP-PEAP—Request Filters

The screenshot shows the Steel-Belted Radius Carrier Web GUI. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and a user profile for 'root'. The main content area is divided into two sections. The first section, 'EAP Methods List', contains a table with columns 'Name' and 'Status'. The second section, 'Selected EAP Method: EAP-PEAP', features a tabbed interface with 'Request Filters' selected. This tab contains four rows of configuration options, each with a checkbox and a dropdown menu. At the bottom of the configuration area are 'Save', 'Reset', and 'Cancel' buttons.

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Selected EAP Method: EAP-PEAP

☐ Enable EAP-PEAP Method

Request Filters	Response Filters	Session Resumption	Inner Authentication	Advanced Server Settings
Transfer Outer Attribs to New:	<input type="checkbox"/>			
Transfer Outer Attribs to continue:	<input type="checkbox"/>			
Edit New:	<input type="checkbox"/>			
Edit Continue:	<input type="checkbox"/>			

Buttons: Save, Reset, Cancel

2. Optionally, select the **Transfer Outer Attribs to New** check box and select the filter you want to use from the **Transfer Outer Attribs to New** list.

This filter affects only a new inner authentication request (rather than continuations of previous requests).

- If this filter is specified, all attributes from the outer request are transferred to the inner request and this filter is applied. The transfer occurs and the filter is applied before any attributes specified in the inner authentication are added to the request.
- If this filter is not specified, no attributes from the outer request are transferred to the inner request.

3. Optionally, select the **Transfer Outer Attrs to continue** check box and select the filter you want to use from the **Transfer Outer Attrs to continue** list.

This filter affects only a continued inner authentication request (rather than the first inner authentication request). If this filter is specified, all attributes from the outer request are transferred to the inner request and this filter is applied. The transfer occurs and the filter is applied before any attributes specified in the inner authentication are added to the request.

If this filter is not specified, no attributes from the outer request are transferred to the inner request.

4. Optionally, select the **Edit New** check box and select the filter you want to use from the **Edit New** list.

This filter affects only a new inner authentication request (rather than continuations of previous requests). If this filter is specified, it is applied to the inner request that is the cumulative result of attributes transferred from the outer request (by the filter specified in Step 2) and attributes included in the inner authentication request sent through the tunnel by the client.

If this filter is not specified, the request remains unaltered.

5. Optionally, select the **Edit Continue** check box and select the filter you want to use from the **Edit Continue** list.

This filter affects only a continued inner authentication request (rather than a new inner authentication request). If this filter is specified, it is applied to the inner request that is the cumulative result of attributes transferred from the outer request (by the filter specified in Step 3) and attributes included in the inner authentication request sent through the tunnel by the client.

If this filter is not specified, the request remains unaltered.

Configuring Response Filters—EAP-PEAP

Response filters affect the attributes in the final response (Access-Accept or Access-Reject) returned to the originating NAD. By default, SBR Carrier does not use response filters.

To configure response filtering for the EAP-PEAP protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-PEAP** pane, click the **Response Filters** tab ([Figure 114 on page 307](#)).

Figure 114: EAP-PEAP—Response Filters

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. The main content area is titled "EAP Methods List" and contains a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Below the table, the "Selected EAP Method: EAP-PEAP" configuration section is shown. It includes a checkbox for "Enable EAP-PEAP Method" and a tabbed interface with the following tabs: Request Filters, Response Filters, Session Resumption, Inner Authentication, and Advanced Server Settings. The "Response Filters" tab is currently selected, showing the following options:

- ☐ Transfer Inner Attribs To Accept: [Dropdown menu]
- ☐ Transfer Inner Attribs To Reject: [Dropdown menu]

At the bottom of the configuration section are buttons for Save, Reset, and Cancel.

2. Optionally, select the **Transfer Inner Attribs to Accept** check box and select the filter you want to use from the **Transfer Inner Attribs to Accept** list.

This filter affects only an outer Access-Accept response that is sent back to a NAD.

- If this filter is specified, the filter is applied to the inner authentication response and all resulting attributes are transferred to the outer authentication response.
- If this filter is not specified, no inner authentication response attributes are transferred to the outer authentication response.

3. Optionally, select the **Transfer Inner Attribs to Reject** check box and select the filter you want to use from the **Transfer Inner Attribs to Reject** list.

This filter affects only a continued inner authentication request (rather than the first inner authentication request). If this filter is specified, all attributes from the outer request are transferred to the inner request and this filter is applied. The transfer occurs and the filter is applied before any attributes specified in the inner authentication are added to the request.

If this filter is not specified, no attributes from the outer request are transferred to the inner request.

Configuring Session Resumption—EAP-PEAP

You use session resumption settings to specify whether session resumption is permitted and under what circumstances session resumption is performed.

NOTE: For session resumption to work, the NAD must be configured to handle the **Session-Timeout** return list attribute, so that the NAD can notify the client to reauthenticate after the session timer has expired.

To configure session resumption for the EAP-PEAP protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-PEAP** pane, click the **Session Resumption** tab (Figure 115 on page 309).

Figure 115: EAP-PEAP—Session Resumption

The screenshot shows the Steel-Belted Radius Carrier configuration interface. At the top, there is a header bar with the title "Steel-Belted Radius Carrier" and the Juniper Networks logo. Below the header is a navigation bar with links: Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root.

The main content area is divided into two sections. The first section is titled "EAP Methods List" and contains a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

The second section is titled "Selected EAP Method: EAP-PEAP". It contains a checkbox labeled "Enable EAP-PEAP Method" which is currently unchecked. Below this is a tabbed interface with five tabs: Request Filters, Response Filters, Session Resumption, Inner Authentication, and Advanced Server Settings. The "Session Resumption" tab is currently selected.

Under the "Session Resumption" tab, there are three input fields, each with a numeric value of 0 and a spin button:

- Session Timeout(In Seconds): 0
- Termination Action: 0
- Resumption Limit(In Seconds): 0

At the bottom of the configuration pane, there are three buttons: Save, Reset, and Cancel.

2. In the **Session Timeout(In Seconds)** field, enter the maximum number of seconds you want the client to remain connected to the NAD before having to reauthenticate.

If you enter a number greater than 0, the lesser of this value and the remaining resumption limit is sent in a **Session-Limit** attribute to the RADIUS client on the RADIUS Access-Accept response.

If you enter 0, no **Session-Limit** attribute is generated. This does not prevent the authentication methods performing secondary authorization from providing a value for this attribute.

Entering a value such as 600 seconds (10 minutes) does not necessarily cause a full reauthentication to occur every 10 minutes. You can configure the resumption limit to make most reauthentications fast and computationally efficient.

3. Enter the value to be returned in a **Termination-Action** attribute in the **Termination Action** field.

The **Termination-Action** attribute is a standard attribute supported by most access points and determines what happens when the session timeout is reached. Valid values are:

- -1: Do not send the attribute; the default value. This does not prevent any authentication method that performs secondary authorization from providing a value for this attribute.
 - 0: Send the **Termination-Action** attribute with a value of 0.
 - 1: Send the **Termination-Action** attribute with a value of 1.
4. Enter the maximum number of seconds you want the client to be able to reauthenticate using the TLS session resumption feature in the **Resumption Limit(In Seconds)** field.

This type of reauthentication is fast and computationally efficient. It does, however, depend on previous authentications and is not as secure as a complete (computationally expensive) authentication. Specifying a value of 0 disables the session resumption feature.

BEST PRACTICE: Using the Resumption Limit Option Effectively

Two scenarios where the resumption limit can be used effectively:

- In a wireless environment, the client is moving between access points. The resumption limit can be tuned to make the handover between access points smoother by not forcing a complete reauthorization that requires repeated verification of user information.

When the new access point queries SBR Carrier, the server replies that the session ID is already valid. Because it is known to be good, repeating the inner authentication is not required, which saves some time. The access point acknowledges the *reauthorization not required* message and the session continues.

- Another use for resumption limit occurs when the server ordinarily requires the client to reauthorize every 10 minutes or so, to ensure the client is still connected. Setting the resumption limit to 3600 seconds with a session timeout of 600 seconds means that the interval reauthorizations are fast and efficient, and a complete reauthorization is required just once an hour instead of every 10 minutes.

Configuring Inner Authentication Settings—EAP-PEAP

Inner authentication settings let you specify the manner in which the inner authentication step operates.

To configure inner authentication settings for the EAP-PEAP protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-PEAP** pane, click the **Inner Authentication** tab (Figure 116 on page 311).

Figure 116: EAP-PEAP—Inner Authentication

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and a user profile for 'root'. The main content area is titled 'EAP Methods List' and contains a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Below the table is the 'Selected EAP Method: EAP-PEAP' configuration pane. It includes a checkbox for 'Enable EAP-PEAP Method' and a tabbed interface with the following tabs: Request Filters, Response Filters, Session Resumption, Inner Authentication (selected), and Advanced Server Settings. The 'Inner Authentication' tab contains two input fields: 'Directed Realm:' and 'Realm Selection Script:'. At the bottom of the pane are buttons for Save, Reset, and Cancel.

2. Optionally, enter the name of a directed realm in the **Directed Realm** field.

Specifying the name of a directed realm causes the request to be routed based on the methods listed in the directed realm. Omitting this setting causes the inner authentication request to be handled like any other request received from a NAD.

3. Optionally, enter the name of a realm selection script in the **Realm Selection Script** field.

You must license the optional JavaScripting module to use realm selection scripts. For information about realm selection scripting, refer [“Creating Realm Selection Scripts” on page 895](#).

Configuring Advanced Server Settings—EAP-PEAP

You use advanced server settings to specify the manner in which the inner authentication step operates.

To configure advanced server settings for the EAP-PEAP protocol using the Web GUI:

1. In the **Selected EAP Method: EAP-PEAP** pane, click the **Advanced Server Settings** tab ([Figure 117 on page 312](#)).

Figure 117: EAP-PEAP—Advanced Server Settings

The screenshot displays the Steel-Belted Radius Carrier Web GUI. The top navigation bar includes links for Home, RADIUS Configuration, Tools, Help, Logout, and the current user (root). The main content area is titled "EAP Methods List" and shows a table with columns for Name and Status. The EAP-PEAP method is selected and highlighted in blue. Below this, the "Selected EAP Method: EAP-PEAP" pane is open, showing the "Advanced Server Settings" tab. This tab contains various configuration fields for EAP-PEAP, including TLS Message Fragment Length, Max Transaction Time, Challenge Timeout, Return MPPE Keys, TLS Protocol Version, DH Prime Bits, Cipher Suites, PEAP Minimum Version, and PEAP Maximum Version. The "Enable EAP-PEAP Method" checkbox is checked. At the bottom of the pane are buttons for Save, Reset, and Cancel.

Name	Status
EAP-PEAP	Disabled
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-TLS Helper	Disabled

Selected EAP Method: EAP-PEAP

☒ Enable EAP-PEAP Method

Request Filters	Response Filters	Session Resumption	Inner Authentication	Advanced Server Settings
TLS Message Fragment Length:	1020			
Max Transaction Time:	120			
Challenge Timeout:	30			
Return MPPE Keys:	<input checked="" type="checkbox"/>			
TLS Protocol Version:	TLSv1.1			
DH Prime Bits:	1024			
Cipher Suites:	0x003C,0x003D,0x0067,C			
PEAP Minimum Version:	0			
PEAP Maximum Version:	1			

Save Reset Cancel

2. In the **TLS Message Fragment Length** field, enter the maximum length of the TLS message that may be generated during each iteration of the TLS exchange.

Enter a number in the range 500 through 4096 bytes. This value affects the number of RADIUS challenge/response round-trips required to conclude the TLS exchange. A value of 1400 bytes may result in 6 round-trips, while a value of 500 bytes may result in 15 round-trips.

Some access points may have problems with RADIUS responses or EAP messages that exceed the size of one Ethernet frame (1500 bytes including IP/UDP headers).

The default length for TLS messages is 1020 bytes, which prevents the RADIUS challenge response (carried in a UDP packet) from exceeding one Ethernet frame.

3. In the **Max Transaction Time** field, enter the maximum number of seconds you want for the authentication sequence to complete.

If the authentication sequence takes longer than this setting, user authentication is terminated.

4. In the **Challenge Timeout** field, enter the number of seconds after which a challenge request times out.

You can enter a value greater than or equal to 1 second, but this value must not exceed the value specified in the **Max Transaction Time** field. The default value is 30 seconds.

5. Select the **Return MPPE Keys** check box to specify whether the EAP-PEAP module includes RADIUS **MS-MPPE-Send-Key** and **MS-MPPE-Recv-Key** attributes in the final RADIUS Access-Accept response sent to the access point.

Select this check box if the access point needs to key the WEP encryption. If the access point is authenticating only end users and WEP is not being used, you can clear this check box.

6. Use the **TLS Protocol Version** list to specify the TLS protocol version on which the server expects the client to initiate the handshake process.

Valid values are TLSv1, TLSv1.1, and TLSv1.2.

7. Use the **DH Prime Bits** list to specify the number of bits in the prime number that the module uses for Diffie-Hellman exponentiation.

Selecting a longer prime number makes the system less susceptible to certain types of attacks but requires more CPU processing to compute the Diffie-Hellman key agreement operation.

Valid values are 512, 1024, 1536, 2048, 3072, and 4096 bits.

8. In the **Cipher Suites** field, enter the TLS cipher suites (in order of preference) that the server is to use.

These cipher suites are documented in RFC 2246, *The TLS Protocol Version 1*, RFC 4346, *The TLS Protocol Version 1.1*, and RFC 5246, *The TLS Protocol Version 1.2*.

The default value is: 0x0067,0x006B,0xC030,0xC028,0xC014,0xC013.

See [Table 36 on page 273](#) for the list of tested cipher suites and their TLS protocol versions.

9. In the **PEAP Minimum Version** field, enter the minimum version of the PEAP protocol that you want the server to negotiate.

If you enter 0, the server negotiates version 0.

If you enter 1, the server negotiates version 1.

NOTE: The value entered in this setting must be less than or equal to the value entered for the **PEAP Maximum Version** field.

10. In the **PEAP Maximum Version** field, enter the maximum version of the PEAP protocol that you want the server to negotiate.

If you enter 0, the server negotiates version 0.

If you enter 1, the server negotiates version 1.

NOTE: The value entered in this setting must be equal to or greater than the value entered for the **PEAP Minimum Version** field.

EAP-MD5-Challenge Authentication Protocol

EAP-MD5-Challenge, which is described in RFC 2284, enables a RADIUS server to authenticate a connection request by verifying an MD5 hash of a user's password. The server sends the client a random challenge value, and the client proves its identity by hashing the challenge and its password with MD5.

EAP-MD5-Challenge is typically used on trusted networks where risk of packet sniffing or active attack are fairly low. Because of significant security vulnerabilities, EAP-MD5-Challenge is not usually used on public networks or wireless networks, because third parties can capture packets and apply dictionary attacks to identify password hashes. Because EAP-MD5-Challenge does not provide server authentication, it is vulnerable to spoofing (a third party advertising itself as an access point).

By default, the EAP-MD5-Challenge password protocol is available for use by the Native and UNIX authentication methods. Support for the EAP-MD5-Challenge protocol is discussed in **eap.ini** file in the *SBR Carrier Reference Guide*.

EAP-MS-CHAP-V2 Authentication Protocol

EAP-MS-CHAP-V2 (Microsoft Challenge-Handshake Authentication Protocol version 2) is a mutual authentication method that supports password-based user or computer authentication. During the EAP-MS-CHAP v2 authentication process, both the client and the RADIUS server must prove that they have knowledge of the user's password for authentication to succeed. Mutual authentication is provided by including an authenticator packet returned to the client after a successful server authentication.

EAP-MS-CHAP-V2 is typically used inside a TLS tunnel created by TTLS or PEAP.

By default, the EAP-MS-CHAP-V2 password protocol is available for use by the Native and UNIX authentication methods. Support for the EAP-MS-CHAP-V2 protocol is discussed in the **eap.ini** file in the *SBR Carrier Reference Guide*.

EAP-SIM and EAP-AKA Authentication Protocols

The EAP-SIM and EAP-AKA authentication protocols are supported with the optional SIM authentication module license. For details on these authentication protocols, see [“SIM Authentication Module” on page 536](#).

Configuring Replication

IN THIS CHAPTER

- Overview of Replication | 316
- Replication Requirements | 319
- Adding a Replica Server | 319
- Enabling a Replica Server | 323
- Editing a Replica Server | 325
- Deleting a Replica Server | 325
- Publishing Server Configuration Information | 326
- Notifying Replica RADIUS Servers | 326
- Designating a New Primary Server | 327
- Making a Standalone Server the Primary Server | 328
- Making a Standalone Server a Replica Server | 328
- Verifying the Primary and Replica Servers Are Enabled | 329
- Demote a Primary or Replica Server to a Standalone Server | 329
- Recovering a Replica After a Failed Configuration Package Download | 330
- Changing the Name or IP Address of a Server | 330
- Replication Error Messages | 331

This section describes how to configure and use the centralized configuration management (CCM) feature to coordinate SBR Carrier server settings in a replication environment. CCM is the process by which information is shared between a primary RADIUS server and one or more replica RADIUS servers in a multi-server environment. This chapter contains these topics:

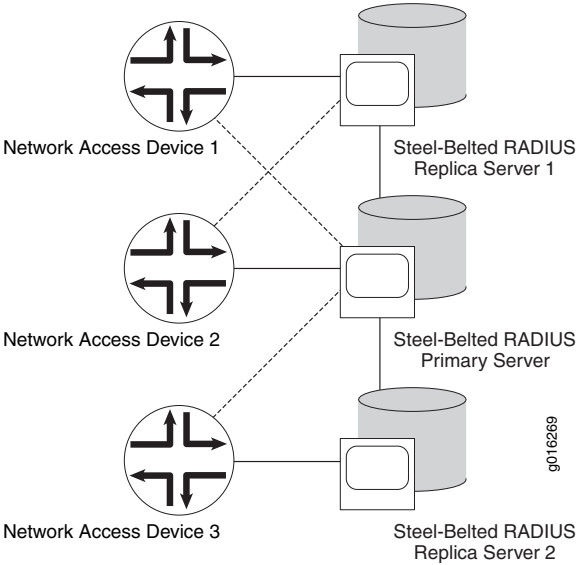
Overview of Replication

SBR Carrier supports the replication of RADIUS configuration data from a *primary* server to one or more *replica servers* within a replication realm. Replication provides administrators with an easy way to configure multiple servers that require the same information. Depending on network configuration, you can use

replication to increase AAA capacity, balance AAA traffic across RADIUS servers, or ensure that authentication services are not interrupted if access to a primary or replica server is interrupted (redundancy).

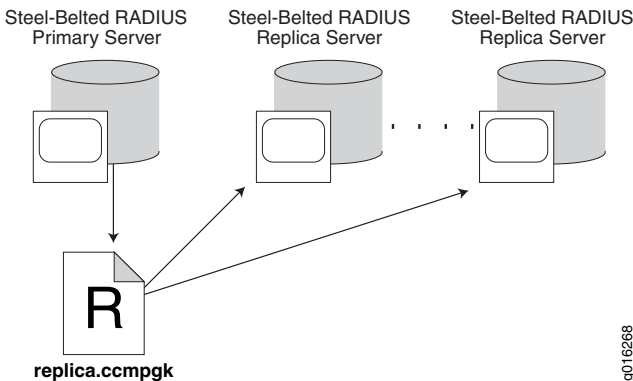
Figure 118 on page 317 illustrates an environment where RADIUS traffic is load-balanced by configuring each network access server to authenticate users through a different RADIUS server (solid line). If a RADIUS server becomes unavailable, the NAD can fail over to its backup RADIUS server (dotted line).

Figure 118: Using Replication for Load Balancing and Redundancy



All the servers within a realm reflect the current configuration specified by the network administrator: the network administrator modifies the configuration on the primary server, and the primary server propagates the new configuration to its replica servers. For example, after a network administrator configures a new RADIUS client or profile on the primary server, the network administrator tells the primary server to publish a date-stamped configuration package file that contains the updated configuration information. After publication, the primary server notifies each replica server that a new configuration package is ready. Each replica server then downloads and installs the configuration package to update its settings.

Figure 119: Publication and Distribution of Configuration Packages in a Replication Environment



The primary server maintains a list of the replica servers that have registered with it. The primary server uses this list to track which servers to notify after it publishes an updated configuration package to resynchronize the configuration of replica servers.

NOTE: By default, file permissions for configuration packages on Solaris servers are set to **rw-rw----**, which excludes users other than the file owner and the owner's group from displaying the contents of file packages.

If the primary server needs to be taken out of service for an extended period, the network administrator promotes one of the replica servers to be the new primary server. Thereafter, the other replica servers copy the configuration package from the promoted primary server.

The following types of information are included in a configuration package.

- | | |
|---|---|
| <ul style="list-style-type: none"> • Server information • RADIUS client information • User information • Profile information • Proxy target information • EAP method configuration • Filters | <ul style="list-style-type: none"> • RADIUS tunnel information • Name parsing information • Authentication method information • Authentication realm information • Rejection messages • JavaScript (.jsi) files |
|---|---|

You administer this information by launching the Web GUI for the primary server: the information is propagated to the replica servers in the domain. If you launch the Web GUI for a replica server, you can view this information, but you cannot modify it.

The following types of information are not included in a configuration package:

- Address pool information—You administer address pools for a server by launching the Web GUI for that server. Because an address must not be assigned to two users at the same time, each server in a realm must have its own address pools, and these pools must not overlap.
- Administrator information—Administrator information must be configured for each primary and replica server separately.
- Statistics information—Server statistics are not replicated. You can view statistics for replica servers when you launch the Web GUI for the primary server.
- Report information—Report information is not replicated. To obtain report information for a primary or replica server, launch the Web GUI for the applicable server.
- SBR Carrier configuration files—Configuration files *.ini files other than **filter.ini** and **eap.ini**, *.aut files other than **peapauth.aut**, **ttlsauth.aut**, **tlsauth.aut**, and **tlsauth.eap**, and *.dir files are not replicated.

When you change configuration files on the primary server, you must copy the modified files to the appropriate directory on each replica server.

NOTE: Configuration packages are retained until they are replaced. An old configuration package is automatically deleted 24 hours after a new configuration package is published.

Replication Requirements

Servers in a replication realm must comply with the following requirements.

- All servers in a replication realm must be running the same operating system version and patches.
- All servers in a replication realm must be running the same version of SBR Carrier.
- All servers in a replication realm must be configured to support the same types of users (host or UNIX).
- The system clocks on servers in a replication realm must be synchronized to within 10 minutes of one another and their time zones must be configured correctly. SBR Carrier uses the system clock value and time zone setting to convert local time to Universal Time Coordinated (also known as Greenwich Mean Time) when evaluating synchronization. If possible, use a Network Time Protocol (NTP) server to set the system clock on all servers automatically.
- All servers in a replication realm must use the same TCP port to exchange replication information. The default port for replication communication is TCP 1812, though you can specify another port for replication traffic by modifying the **radius.ini** file.
- If a firewall stands between servers in a replication realm, the firewall must be configured to pass traffic on the port used for replication communication.

Adding a Replica Server

In most situations, you add a replica server to a realm as follows:

1. Copy the **replica.ccmpkg** configuration package file from the primary server to a directory on the host you want to add as a replica server.

NOTE: The **replica.ccmpkg** file contains sensitive information. Do not store it in a publicly accessible location, such as a file server or shared directory.

2. Install the SBR Carrier server software on the host you want to add as a replica server.
3. When the configuration script asks what kind of server you are installing, choose **Replica** and, when prompted, enter the path to the **replica.ccmpkg** file.
4. Restart the host you want to add as a replica server.

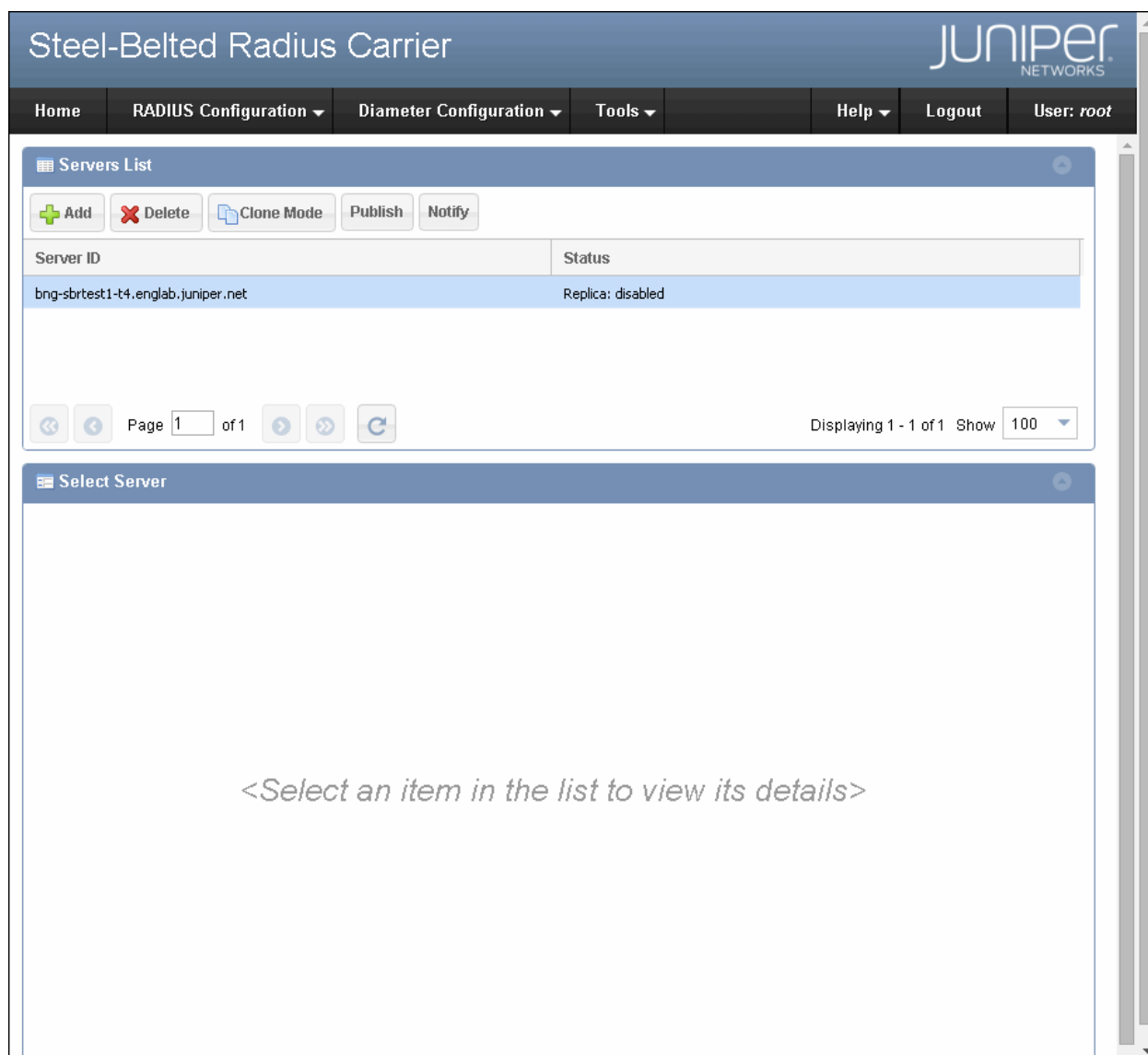
The replica server registers itself with the primary server automatically after it is restarted. Thereafter, the replica server automatically connects to the primary server once an hour to check whether an updated configuration package is available.

In some circumstances, however, you may want to add a replica server to the server list on the primary server manually so that it shows up immediately. To add a replica server manually using the Web GUI:

1. Run the Web GUI for the primary server.
2. Select **RADIUS Configuration > Replication**.

The **Server List** page ([Figure 120 on page 321](#)) appears.

Figure 120: Server List Page



3. Click **Add**.

The **Create Server** pane (Figure 121 on page 322) appears.

Figure 121: Create Server Pane

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration ▼ Diameter Configuration ▼ Tools ▼ Help ▼ Logout User: root

Servers List

Create Server

Name:

Secret: Show

Port: 1812

☐ Enabled

Address

Status:

Publication Path:

Last Published:

Current Publication:

Add Edit Delete

Save Clear Cancel

4. Enter a name for the RADIUS server in the **Name** field.

Although you can assign any name to a RADIUS server, use the device's hostname to avoid confusion.

5. Enter the replication secret for the RADIUS server in the **Secret** field.

For privacy, characters are masked. You can click **Show** to display the characters in the replication secret. After viewing the characters, you can click **Hide** to hide the characters.

6. Enter the port number to be used for replication communication in the **Port** field.

The default port number is 1812.

7. Click **Add** under the **Address** area.

The **Add Address** dialog box ([Figure 122 on page 323](#)) appears.

Figure 122: Add Address Dialog

A screenshot of a web-based dialog box titled "Add Address". The dialog has a blue header bar with the title and a close button (X) in the top right corner. Below the header, there is a label "Address:" followed by a text input field. At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

8. Enter an IP address you want to associate with the server in the **Address** field.

9. Click **OK**.

The **Address** area in the **Create Server** pane ([Figure 121 on page 322](#)) displays an updated list of IP addresses.

You can modify the entered IP address by using the **Edit** and **Delete** buttons.

10. Repeat steps [7](#) through [9](#) to add more IP addresses for the server.

11. Click **Save** to save the server configuration.

The **Server List** page ([Figure 120 on page 321](#)) displays an updated list of server entries.

Enabling a Replica Server

To enable a replica server using the Web GUI:


1. Select **RADIUS Configuration > Replication**.

The **Server List** page ([Figure 120 on page 321](#)) appears.

2. Select the RADIUS server entry you want to enable.

The **Selected Server** pane ([Figure 123 on page 324](#)) displays the settings configured for the server.

Figure 123: Selected Server Pane

Steel-Belted Radius Carrier


Home
RADIUS Configuration
Diameter Configuration
Tools
Help
Logout
User: root

Servers List

Add
Delete
Clone Mode
Publish
Notify

Server ID	Status
bng-sbrtest1-t4.englab.juniper.net	Replica: disabled

Page 1 of 1
Displaying 1 - 1 of 1
Show 100

Selected Server: bng-sbrtest1-t4.englab.juniper.net

Name: bng-sbrtest1-t4.englab.ju
Secret: ***** Show
Port: 1812
☐ Enabled

Address
10.212.11.48

Status: Replica: disabled
Publication Path:
Last Published:
Current Publication:

Add Edit Delete

Save Reset Cancel

3. Select the **Enabled** check box.
4. Click **Save** to enable the replica server.

Editing a Replica Server

To edit a replica server using the Web GUI:

1. Select **RADIUS Configuration > Replication**.

The **Server List** page ([Figure 120 on page 321](#)) appears.

2. Select the server entry you want to edit.

The **Selected Server** pane ([Figure 123 on page 324](#)) displays the settings configured for the server.

3. Edit the settings for the server entry as appropriate.

For information about the fields in the **Selected Server** pane, see [“Adding a Replica Server” on page 319](#).

NOTE: You cannot edit the name of the server.

4. Click **Save** to save the changes.

The **Server List** page ([Figure 120 on page 321](#)) displays an updated list of server entries.

Deleting a Replica Server

To delete a replica server from a realm using the Web GUI:

1. Select **RADIUS Configuration > Replication**.

The **Server List** page ([Figure 120 on page 321](#)) appears.

2. Select the server entry you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to confirm the delete request.

The **Server List** page ([Figure 120 on page 321](#)) displays an updated list of server entries.

Publishing Server Configuration Information

If you change the configuration of your primary server and want to push that data to your replicas, you must manually publish the modified configuration so that your replica servers can download the modified settings. The replication engine is not automated; it will always entail manual intervention to push the data to the replicas.

The following configurations are published to the replicas:

- Server information
- RADIUS client information
- User information
- Profile information
- Proxy target information
- EAP method configurations
- Filters
- RADIUS tunnel information
- Name parsing information
- Authentication method information
- Authentication realm information
- Rejection messages
- Javascript (.js) files

To publish the server configuration information using the Web GUI:

1. Select **RADIUS Configuration > Replication**.

The **Server List** page ([Figure 120 on page 321](#)) appears.

2. Click **Publish**.

A confirmation dialog box is displayed.

3. Click **Yes** to publish the server configuration information.

This creates a file called `/opt/JNPRsbr/radius/packages/timestamp_SBR.ccmpkg`, where timestamp reflects the date and time when the package was created.

Notifying Replica RADIUS Servers

Under normal circumstances, a replica server connects to its primary server every hour to check whether a new replication package **replica.ccmpkg** file has been published. If necessary, a network administrator can manually notify a replica server to download and install the current configuration package from the

primary server. Manual notification informs replica servers a new configuration package is available for download. It is useful when network issues prevent the automatic download and installation of a configuration package when it is first published, and the configuration on the replica no longer matches the configuration on the primary server.

NOTE: You can display the **Server List** page in Web GUI to determine the status of your replica servers.

Notification only informs replica servers a new configuration package is available for download. The configuration package must first be created by publishing the configuration package. See [“Publishing Server Configuration Information” on page 326](#).

To notify replica servers that new configuration information has been published, using the Web GUI:

1. Select **RADIUS Configuration > Replication**.

The **Server List** page ([Figure 120 on page 321](#)) appears.

2. Select the replica server entry you want to notify.

3. Click **Notify**.

The replica server is notified that a new configuration package is available for download. Once the replica server downloads and installs the new configuration package from the primary server, it becomes resynchronized with the primary server.

Designating a New Primary Server

You can change which server within a realm is designated as the primary server for that realm.

To designate a new primary server:

1. Stop the RADIUS service on the replica server.
2. Log in to the replica server as **root**.
3. Navigate to: `/opt/JNPRsbr/radius` or the directory where SBR is installed.
4. Run the **sbrsetuptool** utility with the **promote** option.

```
# sbrsetuptool -promote
```

The utility creates a configuration package to change this server to the primary server.

5. Restart the updated replica server to make it the new primary server.
6. Publish a new configuration package administratively to configure all replica servers to use the new primary server. For more information about how to publish the server configuration information, see [“Publishing Server Configuration Information” on page 326](#).

After you designate a new primary server for a realm, the old primary server becomes a replica server automatically.

Making a Standalone Server the Primary Server

To make a standalone server the primary server:

1. Stop the RADIUS service on the standalone server.
2. Log in to the server as **root**.
3. Navigate to: `/opt/JNPRsbr/radius` or the directory where SBR is installed.
4. Run the following command:


```
# sbrsetuptool -install -path /opt/JNPRsbr/radius -identity PRIMARY -secret <secret password>
```
5. Check the `sbrsetuptool.log` file for a line indicating that the server is now configured as a PRIMARY.
6. Restart the server.

The `replica.ccmpkg` is now present in the `/opt/JNPRsbr/radius` directory. This can be used to register replica servers.

Making a Standalone Server a Replica Server

1. Stop the RADIUS service on the standalone server.
2. Log in to the server as **root**.
3. Navigate to: `/opt/JNPRsbr/radius` or the directory where SBR is installed.
4. Run either of these commands:

```
# sbrsetuptool -install -path /opt/JNPRsbr/radius -identity REPLICA -radpath <path to the REPLICA.CCMPKG file>
```

or

```
# sbrsetuptool -install -path /opt/JNPRsbr/radius -identity REPLICA -primary <made up name of primary>
<IP address of PRIMARY> <secret password of primary>
```

5. Check the **sbrsetuptool.log** file for a line indicating that the server is now configured as a replica.
6. Restart the server.

Verifying the Primary and Replica Servers Are Enabled

To perform the verification using the Web GUI when the primary server is running and the replica server has been registered:

1. Select **RADIUS Configuration > Replication**.

The **Server List** page ([Figure 120 on page 321](#)) appears.

2. Verify that both the primary and replica servers are listed in the **Server List** page.

If the replica server is disabled, enable it. For more information about how to enable a replica server, see [“Enabling a Replica Server” on page 323](#).

3. Publish the configuration. For more information about how to publish the server configuration information, see [“Publishing Server Configuration Information” on page 326](#).

4. Click  a few times and ensure both servers are up to date.

Demote a Primary or Replica Server to a Standalone Server

To take a replica or primary server and demote it back to a standalone server:

1. Stop the RADIUS service on the standalone server.
2. Log in to the server as **root**.
3. Navigate to: /opt/JNPRsbr/radius or the directory where SBR is installed.
4. Run the following command:

```
# sbrsetuptool -install -path /opt/JNPRsbr/radius -identity SA
```

Recovering a Replica After a Failed Configuration Package Download

If a replica server fails during the download of a configuration package, its configuration may be corrupted or it may have a stale secret.

To recover after a failed download:

1. Stop the RADIUS service on the replica server.
2. Log in to the replica server as **root**.
3. Navigate to: `/opt/JNPRsbr/radius` or the directory where SBR is installed.
4. Run the **sbrsetuptool** utility with the **identity** option and information about where to download configuration information.

To obtain configuration from a configuration package, issue the following command:

```
# sbrsetuptool -identity REPLICA -radpath pathname
```

Where *pathname* specifies the path to a **replica.ccmpkg** package.

To obtain configuration from the primary server for the realm, issue the following command:

```
# sbrsetuptool -identity REPLICA -primary name address secret
```

Where *name* specifies the DNS name of the primary server, *address* specifies the IP address of the primary server, and *secret* specifies the shared secret used to authenticate configuration downloads.

5. Restart the updated replica server so that it can load the latest published configuration from the primary server.

After the replica server is restarted, it re-synchronizes with the current primary server.

Changing the Name or IP Address of a Server

You may need to change the DNS name or IP address assigned to a primary or replica server if your network changes.

To change the DNS name or IP address of a primary or replica server using the Web GUI:

1. Stop the RADIUS service on the RADIUS server you want to change.
2. Log in to the RADIUS server as **root**.
3. Navigate to: `/opt/JNPRsbr/radius` or the directory where SBR is installed.
4. Run the **sbrsetuptool** utility with the **identity** option.

To rename a primary server, enter the following command:

```
# sbrsetuptool -identity PRIMARY
```

To rename a replica server, enter the following command:

```
# sbrsetuptool -identity REPLICA
```

- 5. Restart the updated server so that it can load its new configuration.
- 6. In the **Server List** page ([Figure 120 on page 321](#)), modify the DNS name or IP address of the server you want to rename and verify that the shared secret on the renamed server is correct.

You may need to use the **Server List** page to delete the old server name from the list of servers in the realm.
- 7. Publish the modified configuration to propagate the name change to the replica servers. For more information about how to publish the server configuration information, see [“Publishing Server Configuration Information” on page 326](#).

Replication Error Messages

The following tables list possible causes for error messages caused by replication issues.

Error Messages on Replica Servers

[Table 37 on page 331](#) lists possible causes for error messages on replica servers in a replication realm.

Table 37: Error Messages on Replica Servers

Error Type	Error Message	Possible Cause
Post Errors (Errors with Notification from Primary)	CRadManagedServerNotifyPost::ExecutePost invalid signature!	Mismatched replication secret.
	CRadManagedServerNotifyPost::ExecutePost invalid sequence number	Two posts have the same sequence number. The clocks on the primary and replica are more than 10 minutes apart.

Table 37: Error Messages on Replica Servers (continued)

Error Type	Error Message	Possible Cause
	CRadManagedServerNotifyPost::ExecutePost decrypt failed	Shared secret failed to decrypt. Bad Replication Shared secret.
	CRadManagedServerNotifyPost::ExecutePost invalid <body> missing parameters	Post had an invalid xml request.
Update Errors (Errors with Published package from Primary)	CRadManagedServerUpdate::DoStart Failed to open 'file_name' for writing	Temp directory does not exist. Temp directory or file have incorrect permissions.
	CRadManagedServerUpdate::StartUpdates has already started update	Update is already in progress.
	CRadManagedServerUpdate::DownloadPackage HTTP POST error:errCode Primary ID	Error in transmitting request to Primary (timeout during transmit).
	CRadManagedServerUpdate::DownloadPackage HTTP headers parsing error	Error in receiving package. Typically caused by a timeout during receive resulting from an invalid package.
	CRadManagedServerUpdate::DownloadPackage connection primary IP Addr error: errCode Primary ID	Replica failed to connect with Primary. Primary not running.
	CRadManagedServerUpdate::DownloadPackage exceeded iterations limit while communicating with CCM	Update failed after three attempts.

Table 37: Error Messages on Replica Servers (continued)

Error Type	Error Message	Possible Cause
	CRadManagedServerUpdate:: ProcessPackage signature mismatch	Secrets on Replica and Primary do not match.
	CRadManagedServerUpdate:: ProcessPackage CCM error: 'Error String' 'Parameter'	Error parsing downloaded packages.
	CRadManagedServerUpdate:: ProcessPackage Failed to open \" << file_name << \" for writing	Temp directory does not exist. Temp directory or file has incorrect permissions.
	CRadManagedServerUpdate:: ProcessPackage thumbprint mismatch	Invalid package. Republish the package.
Proxy Errors (Statistics retrieve errors)	CRadProxyPost:: ExecutePost invalid signature!	Mismatched replication secret.
	CRadProxyPost:: ExecutePost invalid sequence number	Two posts have the same sequence number. The clocks on the primary and replica are more than 10 minutes apart.
	CRadProxyPost:: ExecutePost decrypt failed	Shared secret failed to decrypt. Bad Replication Shared secret.
	CRadProxyPost:: ExecutePost invalid <body> missing parameters	Post had an invalid XML request.

Error Messages on Primary Servers

Table 38 on page 334 lists possible causes for error messages on primary servers in a replication realm.

Table 38: Error Messages on Primary Servers

Error Type	Error Message	Possible Cause
Notify Target (Both Notify and Publish send a notification)	CRadConfigManagedServerHTTP Notification::NotifyTarget failed to fetch	Replica does not exist in database. This can occur if two administrators are running instances of Web GUI, one administrator deletes a replica, then the other administrator tries to publish to that replica.
	CRadConfigManagedServerHTTP Notification::NotifyTarget failed <i>replicald</i>	Notify failed to communicate with replica, Replica is not running, or check Replica DCF log for more information.
Publication Provider (requests from GUI to Notify or Publish)	CRadConfigPublicationProvider::UpdateResource notify invoked when not Primary	Attempted to Notify as a Replica, this is only allowed from a Primary.
	CRadConfigPublicationProvider::UpdateResource publish invoked when not Primary	Replica attempted to publish; publication is permitted only from a Primary.
Publication Post (parsing of Post from replica to get data)	CRadConfigServerProviderPost::ExecutePost signature mismatch with server:	Replica and Primary have different replication secrets.
	CRadConfigServerProvider::GetResource invoked when not Primary	Another Replica is requesting a download from this server which is a replica. Replica that is requesting a download needs to be reconfigured.

Table 38: Error Messages on Primary Servers *(continued)*

Error Type	Error Message	Possible Cause
Proxy Errors (Statistics retrieve errors)	CRadProxyClient:: Send failed to fetch	<p>Replica does not exist in database.</p> <p>This can occur if two administrators are running instances of Web GUI, one administrator deletes a replica, then the other administrator tries to publish to that replica.</p>
	CRadProxyClient:: SendData HTTP POST error:	Connection error with replica.

3GPP Support

IN THIS CHAPTER

- Overview | 336
- 3GPP Configuration | 337

This chapter describes the 3GPP support in the Steel-Belted Radius Carrier server. This chapter contains these topics:

Overview

Steel-Belted Radius Carrier extends RADIUS functionality to 3GPP implementation on the scale required by Internet Service Providers and carriers. 3GPP support facilitates the management of mobile sessions and their associated resources through communication with a Gateway GPRS Support Node (GGSN). The 3GPP support in Steel-Belted Radius Carrier is based on the specifications given in the *Interworking between the Public Land Mobile Network (PLMN) supporting Packet Based Services and Packet Data Networks (PDN)* documentation (TS 29.061), which is available at www.3gpp.org.

Within the 3G network, the AAA server (Steel-Belted Radius Carrier) interfaces with only one type of RADIUS client device, called a Gateway GPRS Support Node (GGSN). The GPRS network consists of a GGSN and a Serving GPRS Support Node (SGSN). The SGSN is responsible for detecting mobile stations (MS) in its serving area, data packet transmissions to and from the MS, and monitoring the location of MS in its serving area. The GGSN serves as a gateway between the GPRS network and external Packet Data Networks (PDN) such as the Internet or private data networks.

Data Connection Process

Establishing a data connection between an MS and PDN is a two-phase process:

1. GPRS attach procedure
2. PDP context activation

The GPRS attach procedure creates a logical link between the MS and the SGSN. This procedure includes user authentication (identity verification) and authorization to access GPRS services. After the attach procedure is complete, the MS is ready to establish the data session.

To transmit or receive GPRS data, the MS must activate a Packet Data Protocol (PDP) context. The PDP context is a set of parameters of all the information required for establishing an end-to-end connection such as PDP type (for example, IP), PDP address (IPv4 or IPv6), Quality of Service (QoS) parameters, and Network Service Access Point Identifier (NSAPI). After the PDP context is activated, a point-to-point tunnel is established between the SGSN and the GGSN allowing for data transmission between the MS and the requested application.

A single MS might require multiple sessions, using multiple PDP contexts on a single GGSN. Multiple PDP contexts enable the MS to simultaneously access multiple applications. However, each application may require a different set of transmission parameters. For example, an e-mail application might require different QoS parameters than a Multimedia Messaging Service (MMS) application, which uses images, audio, video, and rich text.

Steel-Belted Radius Carrier supports the ability to differentiate between these sessions by using a concatenation of the value of the **User-Name** attribute with the one-character NSAPI (network service access point identifier) value.

Accounting Process

The 3GPP specification enables GGSN to generate multiple Accounting Start/Stop pairs for a given session. When 3GPP is enabled, the Steel-Belted Radius Carrier server expects the GGSN to send the 3GPP-Session-Stop-Indicator attribute as needed in Accounting Stop requests.

- When a GGSN sends an Accounting Stop request that includes a 3GPP-Session-Stop-Indicator attribute (regardless of its value), the GGSN notifies the AAA server to delete the session.
- When a GGSN sends an Accounting Stop request that does not include a 3GPP-Session-Stop-Indicator attribute in the request, the Steel-Belted Radius Carrier server marks the session as dormant and does not free any of its allocated resources (for example, IP address).

This type of Accounting Stop request is typically followed by an Accounting Start that marks the session as being active again.

3GPP Configuration

Support for 3GPP in the Steel-Belted Radius Carrier is configured in the **3gpp.ini** file, which contains the 3GPP settings.

For details on configuring the **3gpp.ini** file, see the *SBR Carrier Reference Guide*.

4

PART

Diameter Operations

Diameter Basics | **339**

Administering the Local Network Element | **349**

Administering Diameter Remote Network Elements | **358**

Administering the Diameter Policy | **376**

Administering Request Routing Rules | **417**

Displaying Diameter Statistics | **429**

Diameter Basics

IN THIS CHAPTER

- Diameter Overview | 339
- Communication between SBR Carrier Server and the Elements in LTE Network | 341
- Diameter Authentication Process | 347
- Diameter Authorization Process | 347
- RADIUS to Diameter Translation | 347

This chapter describes a conceptual overview of Diameter AAA services. This chapter contains the following sections:

Diameter Overview

Diameter is an industry-standard protocol for providing AAA services for applications such as network access or IP mobility in both local and roaming situations. The Diameter protocol is widely used in the IP Multimedia Subsystem (IMS) and Long Term Evolution (LTE) architectures for LTE or IMS entities (such as Home Subscriber Server (HSS)) to exchange AAA-related information.

The Diameter protocol has the following benefits:

- Acts as a reliable transport protocol (Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP))
- Satisfies network or transport layer security (IPsec or TLS)
- Provides larger address space than RADIUS for attribute-value pairs (AVPs) and identifiers
- Provides better roaming support
- Enables dynamic discovery of peers
- Supports additional extensions to add new command codes and AVPs to create new Diameter applications without modifying the existing code

- Provides basic support for user-sessions and accounting
- Supports server-initiated messages, such as a request to terminate service to a particular user

NOTE: SBR Carrier does not implement network or transport layer security, user-session management, or accounting. Also, the addition of new command codes and/or AVPs to create new Diameter applications is not supported.

A Diameter-based remote access environment typically involves the following types of components:

- A *Diameter node* is a machine that performs the hosting process to implement the Diameter protocol. The Diameter node may act as a client, an agent, or a server.
- The *Diameter agent* is a Diameter node that provides relay, proxy, redirect, or translation services.
- The *Diameter client* is a Diameter node that supports Diameter client applications as well as the base protocol. Diameter clients are often implemented in devices situated at the edge of a network and provide access control services for the network. Typical examples of Diameter clients include the Network Access Device (NAD) and the mobile IP foreign agent.
- *Diameter peers* represent two Diameter nodes sharing a direct TCP or SCTP transport connection.
- The *Diameter server* is a Diameter node that handles AAA requests for a particular realm. The Diameter server must support Diameter server applications in addition to the base protocol.

Diameter Application

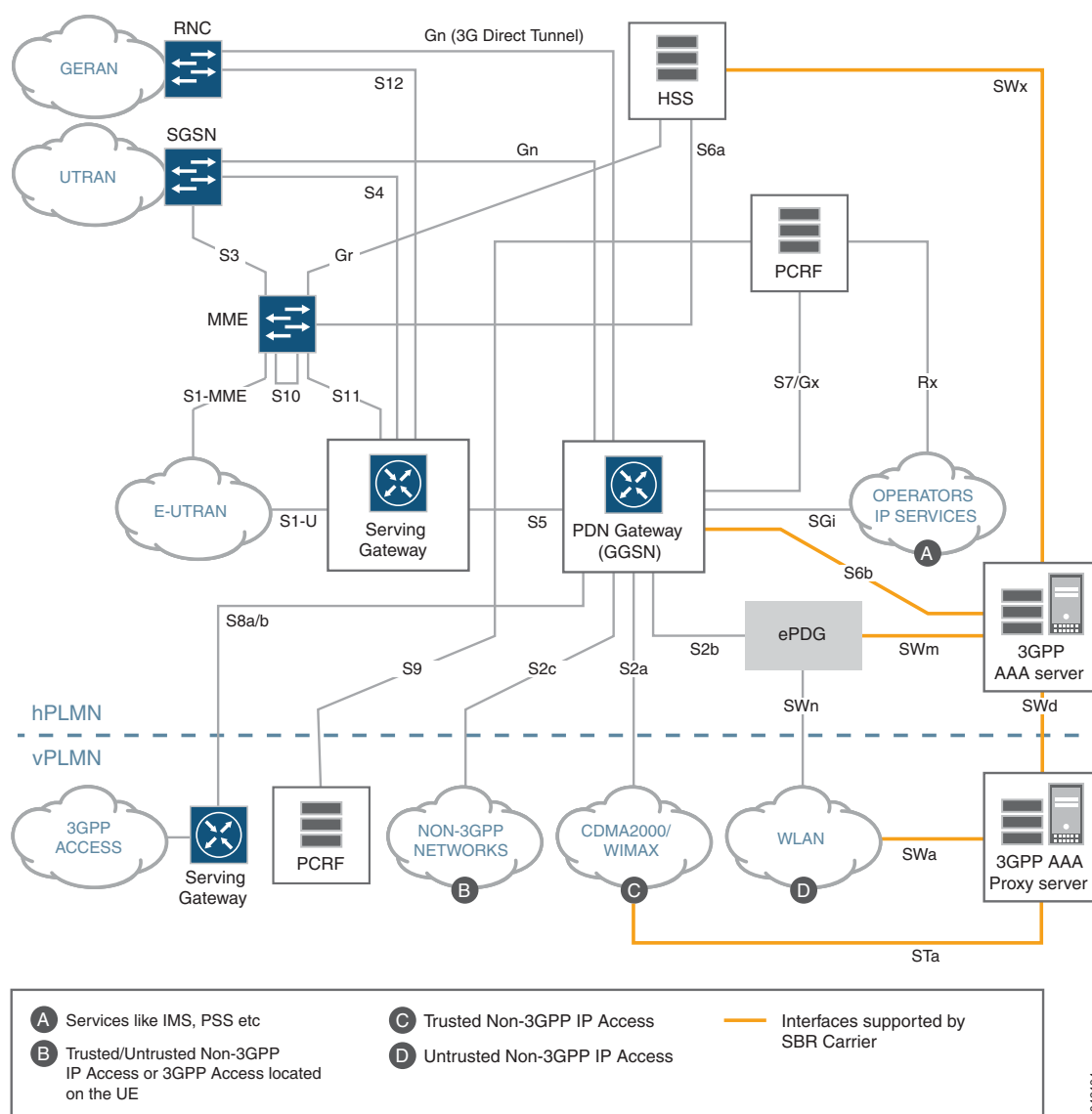
A Diameter application is a protocol (not a software application) based on the Diameter base protocol. Each application is defined by an application identifier. You can add new command codes or new mandatory AVPs to the Diameter application. Adding a new optional AVP does not require a new application. Examples of Diameter application are:

- Diameter Mobile IPv4 Application
- Diameter Network Access Server Application
- Diameter Session Initiation Protocol Application
- Diameter Credit-Control Application

Communication between SBR Carrier Server and the Elements in LTE Network

The network devices that are used to setup an LTE network are called network elements. Each network element performs a specific function. The network elements communicate with each other over reference points, which can also be referred to the interface. [Figure 124 on page 341](#) illustrates the usage of SBR Carrier in a LTE network environment.

Figure 124: SBR Carrier in a LTE Network Environment

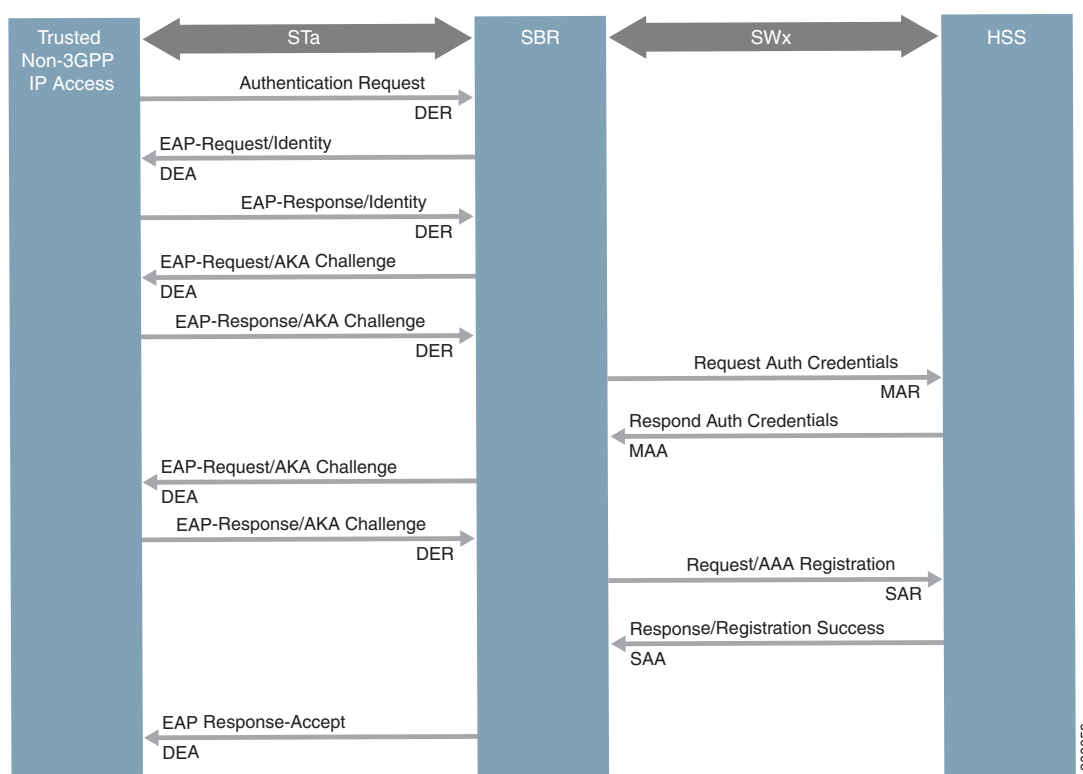


Communication with Trusted Non-3GPP Network

The STa reference point connects the trusted non-3GPP access network with the SBR Carrier server or proxy server (that is, 3GPP AAA server or proxy server) and transports access authentication, authorization, mobility parameters, and charging-related information in a secure manner. The SWa or STa reference point determines whether the non-3GPP access network is trusted or not during the authentication and authorization procedures executed between the non-3GPP access network and the 3GPP AAA server.

The STa and SWa reference points use the same Diameter application and partly share the same authentication and authorization procedure. The other procedures are specific to the STa and SWa reference points. [Figure 125 on page 342](#) illustrates the EAP authentication message flow between the trusted non-3GPP network and HSS.

Figure 125: EAP Authentication Message Flow Between Trusted Non-3GPP Network and HSS



Communication with Untrusted Non-3GPP Network

The SWa reference point connects the untrusted non-3GPP access network with the SBR Carrier server or proxy server (that is, 3GPP AAA server or proxy server) and transports access authentication, authorization and charging-related information in a secure manner. The SWa or STa reference point

determines whether the non-3GPP access network is trusted or not during the authentication and authorization procedures executed between the non-3GPP access network and the 3GPP AAA server.

The authentication and authorization procedures defined for the STa reference point are also used in the SWa reference point, but with the following differences:

- Information about the user's service request and the access network may not be included in the authentication and authorization request.
- Information that describes the user's subscription profile is not downloaded to the non-3GPP access network.

Communication with HSS

SBR Carrier uses Diameter to communicate with an HSS through the SWx reference point to obtain authentication, subscription and PDN connection-related data. HSS contains subscriber information and authentication credentials such as user identity keys and subscription information (for example, International Mobile Subscriber Identity (IMSI), mobile station ISDN (MSISDN), and user profile information), including service subscription states and QoS parameters specific to the user.

The SWx reference point is used to perform non-3GPP access location management procedure for the following purposes:

- To register the current SBR Carrier server address in the HSS for a 3GPP user. SBR Carrier initiates the registration procedure after authenticating a new subscriber (either during attach or handover). As part of the response, HSS returns the subscriber's user profile data (QoS profile, user capabilities, and so on.) to SBR Carrier.
- To de-register the currently registered SBR Carrier server address in the HSS for the 3GPP user and purge any related non-3GPP user status data in the HSS. SBR Carrier de-registers its address and purges user status data when the user is not within the non-3GPP access coverage area, another evolved packet core (EPC) network entity (for example, charging system) has initiated a disconnection, or a re-authentication failure occurs.
- To purge the user equipment from SBR Carrier. HSS initiates the purging process when the user's subscription has been cancelled or for other operator-determined reasons.

Communication with Proxy Servers

The SWd reference point connects the proxy servers, possibly through intermediate networks, to the SBR Carrier server. Some specific characteristics of this reference point are:

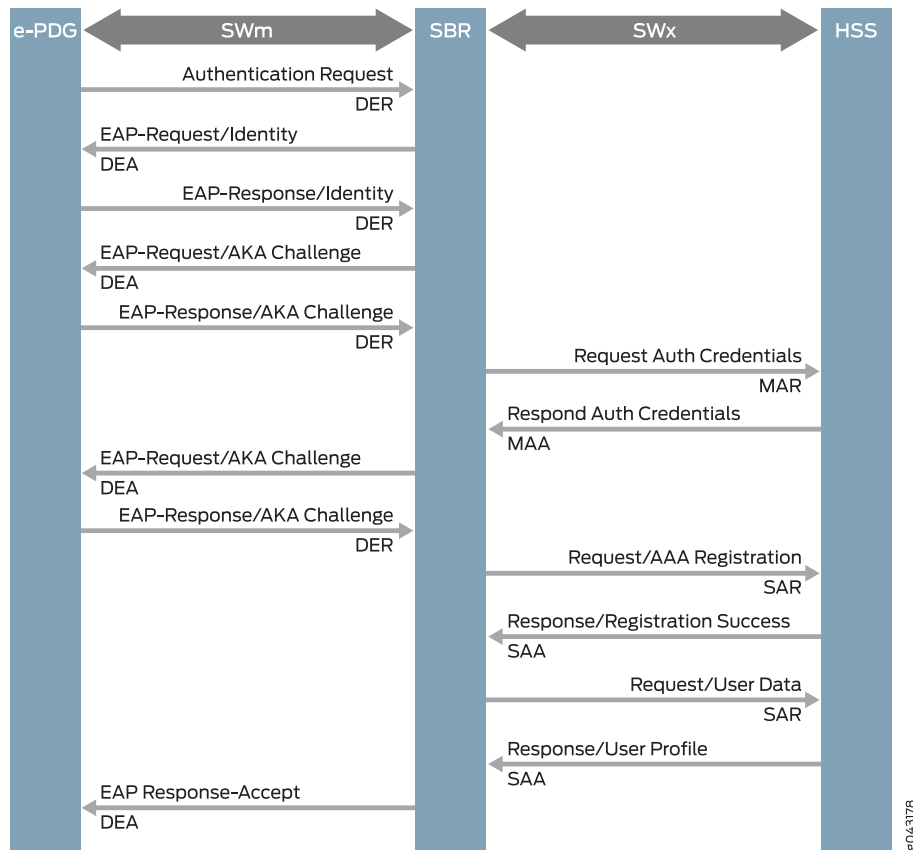
- Carries data for authentication and authorization signaling between the proxy server and the SBR Carrier server.
- Carries keying data for the purpose of radio interface integrity protection and encryption.

- Purges a user from the access network for immediate service termination.

Communication with ePDG

The SWm reference point connects the Evolved Packet Data Gateway (ePDG) with the SBR Carrier server or proxy server (that is, 3GPP AAA server or proxy server) and transports access authentication, authorization, and subscription profile data from the SBR Carrier server or proxy server to the ePDG. The subscription profile information is fetched from the HSS by the SBR Carrier server. The SWm reference point is also used to transport session termination indications and requests initiated from both the SBR Carrier server and ePDG. [Figure 126 on page 344](#) illustrates the EAP authentication message flow between the ePDG and HSS.

Figure 126: EAP Authentication - Message Flow



The SWm reference point supports both pseudonym authentication and fast re-authentication. SBR Carrier makes access restriction decisions based on the values in the following AVPs that are transmitted from the HSS as part of the Non-3GPP-User-Data AVP, which is a Grouped AVP:

- Non-3GPP-IP-Access
- Non-3GPP-IP-Access-APN
- Service-Selection
- Visited-Network-Identifier
- VPLMN-Dynamic-Address-Allowed

NOTE: SBR Carrier supports all the mandatory SWm AVPs specified in 3GPP TS 29.273.

Communication with PDG or PGW

The S6b reference point connects the packet data gateway (PDG) (that is, Packet Data Network Gateway (PGW)) with the SBR Carrier server or proxy server (that is, 3GPP AAA server or proxy server). The S6b reference point is used to authenticate and authorize the user equipment and update the PDG address to the SBR Carrier server or proxy server and HSS. The S6b reference point is also used to download subscriber information to the PDG. [Figure 127 on page 346](#) and [Figure 128 on page 346](#) respectively illustrate the EAP authentication message flow and authorization message flow between the PDG and HSS.

NOTE: SBR Carrier supports all the mandatory S6b AVPs specified in 3GPP TS 29.273.

Figure 127: EAP Authentication Message Flow Between PDG and HSS

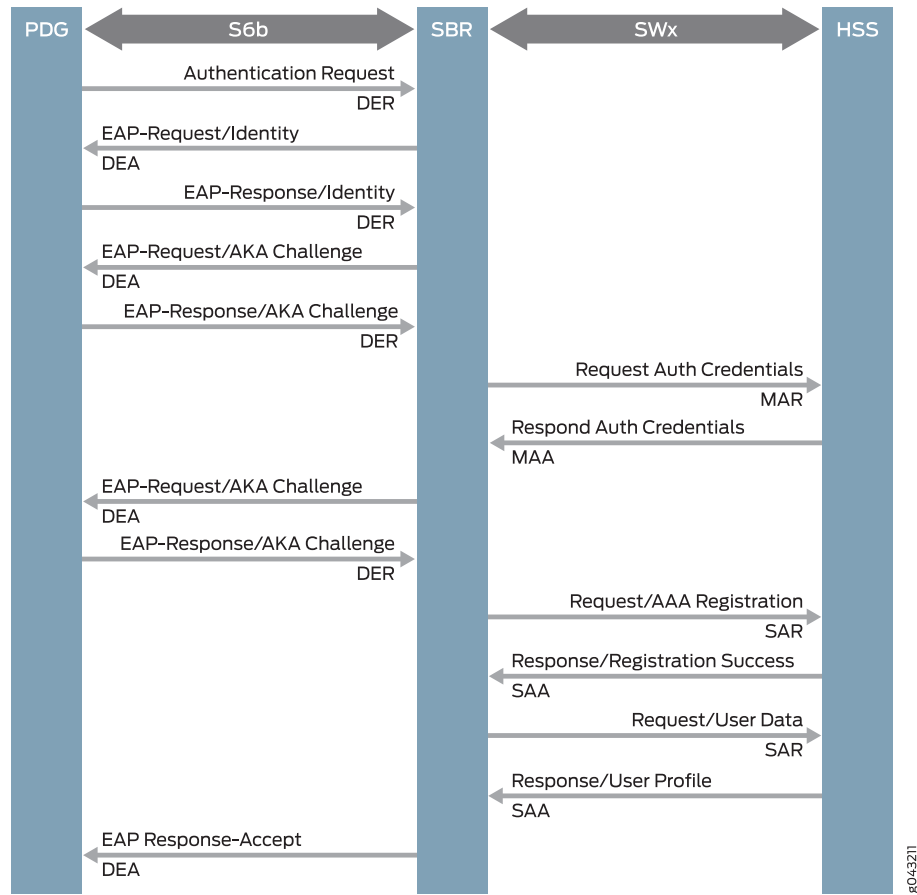
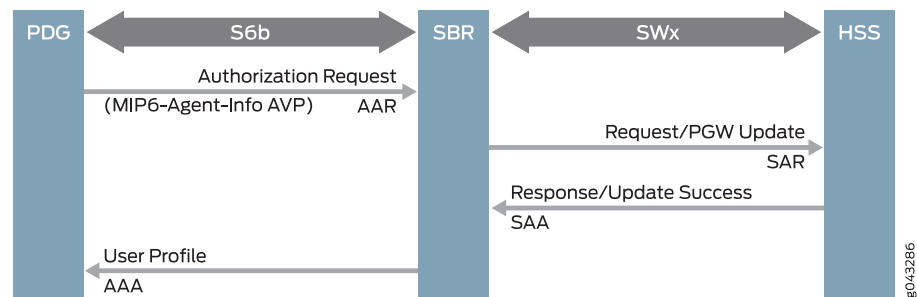


Figure 128: Authorization Message Flow Between PDG and HSS



Diameter Authentication Process

SBR Carrier uses EAP-AKA and EAP-AKA' protocols to perform Diameter authentication. SBR Carrier receives Access-Requests from non-3GPP network elements and forwards them to HSS for validation. The HSS returns the authentication vectors for the particular subscriber to the 3GPP AAA module, which in turn initiates a challenge handshake. Once the challenge handshake is successful, authentication is successful.

Diameter Authorization Process

SBR Carrier performs Diameter authorization based on subscriber data provided by HSS. SBR Carrier can enforce fine-grained authorization policy by using local profiles. SBR Carrier uses local profiles in combination with the subscriber profile stored in the HSS to provide additional security for authorization. During the authorization process, SBR Carrier uses user-defined profile selection rule sets to select the local profile used to authorize the request.

SBR Carrier determines what service scenario is being requested based on the contents and the origin of the request (non-3GPP network access device). The service scenario is relevant to authorization (the user's subscriptions).

SBR Carrier downloads the primary non-3GPP network profile data used for policy decisions from the HSS during authentication. SBR Carrier downloads whatever information is available from the HSS, and tolerates any (or all) missing values because it is assumed that the Web GUI configures the local profile to compliment the HSS profile. The protocols at the SWx reference point ensure that the downloaded profile remains synchronized with HSS while the SBR Carrier server is registered with the HSS. In particular, HSS invokes the policy update procedure if the policy is changed on the HSS while the SBR Carrier server is the registered AAA server.

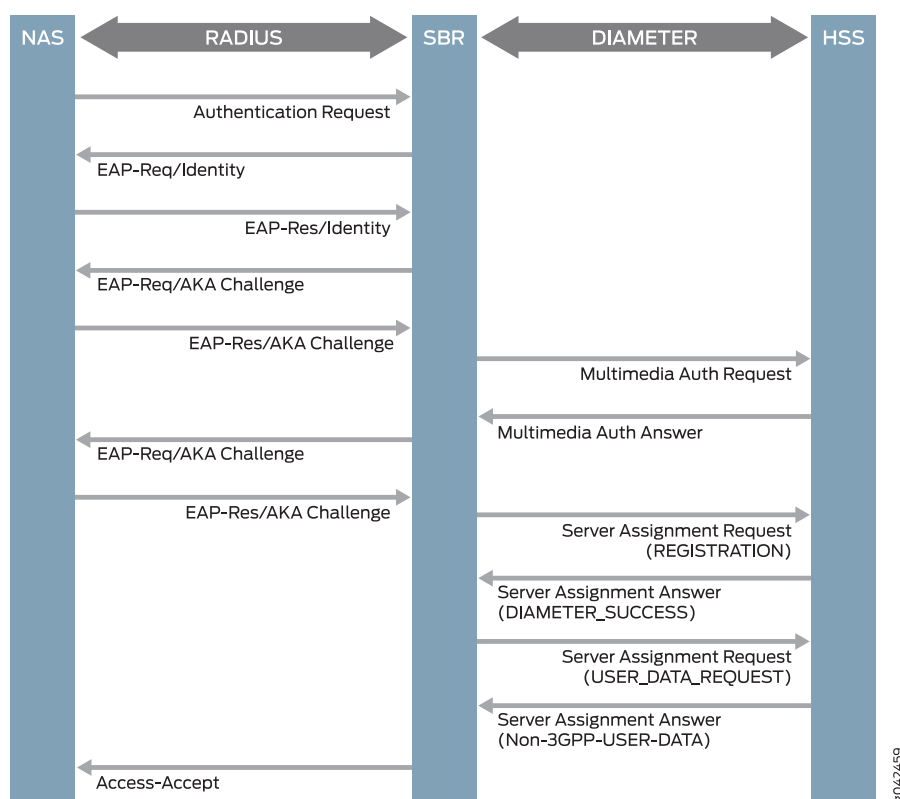
RADIUS to Diameter Translation

SBR Carrier supports RADIUS to Diameter translation to convert RADIUS authentication or authorization requests to Diameter authentication or authorization requests.

[Figure 129 on page 348](#) illustrates the RADIUS to Diameter translation function.

NOTE: In the RADIUS to Diameter translation scenario, the default length of the reauthentication identity is set to 72 bytes and the length of the pseudonym identity is hard-coded to 33 bytes. You can use the **Temp_Identity_Length** parameter in the **radius_listener.xml** file (available in the **/opt/JNPRsbr/radius/system** path) to modify the length of the reauthentication identity to a value ranging from 63 through 253 bytes. But, we recommend that you not modify the parameter names or values in the **radius_listener.xml** file. We also recommend that you use the default length for the reauthentication identity to support NADs that handle only the minimum length for the **User-Name** attribute.

Figure 129: RADIUS to Diameter Translation for Requests



NOTE: SBR Carrier does not support generating CoA/DM messages when a request is sent for RADIUS to Diameter conversion.

Administering the Local Network Element

IN THIS CHAPTER

- Local Network Element Overview | 349
- Configuring SBR Carrier Server Identification | 349
- Configuring the Diameter Message Transport | 353

This chapter describes how to configure the local network element, which includes the server identification and Diameter message transport configuration. This chapter contains the following sections:

Local Network Element Overview

The SBR Carrier server acts as a local network element, which communicates with Diameter remote network elements over bidirectional Diameter connections.

Configuring SBR Carrier Server Identification

You can configure SBR Carrier server identification by configuring the local identity, the local addresses advertised by the server during capability exchange request authentication, and the self names or realm names used by remote network elements to refer the local network element.

Setting Up the Local Identity for SBR Carrier

The local identity of SBR Carrier is specified by the server *Origin-Host* and *Origin-Realm*.

NOTE: The local identity of SBR Carrier is pre-configured with the *Origin-Host=your-host.your-realm.net* and the *Origin-Realm=your-realm.net*. Reconfigure these settings for your network environment.

To configure the local identity of the SBR Carrier server using the Web GUI:

1. Select **Diameter Configuration > Local Network Element > Identification**. The **Identification** page (Figure 130 on page 350) appears.

Figure 130: Identification Page

The screenshot displays the 'Identification' page of the Steel-Belted Radius Carrier Web GUI. The page is titled 'Steel-Belted Radius Carrier' and features the Juniper Networks logo. The navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user 'root'.

The main content area is divided into three sections:

- Local Identity:** Contains two text input fields: 'Host Name (Origin-Host):' with a placeholder '<<host name>>' and 'Realm Name (Origin-Realm):' with a placeholder '<realm name >>'.
- Local Address:** Contains a large text area for 'Addresses this server advertises during CER:' and two buttons: 'Add' and 'Delete'.
- Self Names:** Contains a large text area for 'Realm names that others use to refer to the local server:' and two buttons: 'Add' and 'Delete'.

At the bottom of the page, there are three buttons: 'Save', 'Reset', and 'Refresh'.

2. Enter the Origin-Host name of the server in the **Host Name (Origin-Host)** field.
SBR Carrier uses this name in Origin-Host AVPs.
3. Enter the name of the realm or network in which the server resides in the **Realm Name (Origin-Realm)** field.

SBR Carrier uses this realm name in Origin-Realm AVPs.

4. Click **Save** to save the server identification configuration.

Configuring Local Addresses for the SBR Carrier Server

The local addresses specify IP addresses that the server uses for sending Diameter Capabilities Exchange messages (CER/CEA messages) between Diameter peers. The first Diameter message exchanged between two Diameter peers, after establishing the transport connection, is the Capabilities Exchange message. The Capabilities Exchange message carries a peer's identity and its capabilities (such as the protocol version number and the supported Diameter applications). A Diameter node only transmits commands to peers that have advertised support for the appropriate Diameter application.

To add a local address for the server using the Web GUI:

1. Select **Diameter Configuration > Local Network Element > Identification**. The **Identification** page (Figure 130 on page 350) appears.
2. Click **Add** in the **Local Address** area.

The **Add Local Address** dialog box (Figure 131 on page 351) appears.

Figure 131: Add Local Address Dialog

The image shows a dialog box titled "Add Local Address" with a close button in the top right corner. Inside the dialog, there is a label "IP Address:" followed by a text input field. To the right of the input field is a checkbox labeled "Use IPv6". At the bottom right of the dialog are two buttons: "OK" and "Cancel".

3. Enter the IP address of the server in the **IP Address** field.
4. Optionally, select the **Use IPv6** check box to use IPv6 addressing.

NOTE: SBR Carrier does not support an embedded IPv4 address as an IPv6 address.

5. Click **OK**.

The **Local Address** area displays an updated list of local address entries.

To delete a local address using the Web GUI:

1. Select **Diameter Configuration > Local Network Element > Identification**. The **Identification** page ([Figure 130 on page 350](#)) appears.
2. Select the local address entry you want to delete.
3. Click **Delete** in the **Local Address** area.
A confirmation dialog box is displayed.
4. Click **Yes** to delete the address.

The **Local Address** area displays an updated list of local address entries.

Configuring Local Realm for the SBR Carrier Server

Self names specify the realm names. The Network Access Identifier (NAI) in the request identifies the intended realm name for servers. In order to properly interpret requests received from intermediate servers, SBR Carrier must know which realms it is responsible for servicing locally.

When a request is received, the server examines the NAI to determine the realm to which the request should be routed. If the request does not contain an NAI, the Destination-Realm in the request is used. If the realm to which the request is to be routed is listed in the **Self Names** field, the realm is ignored and the request is treated as if it contained no realm. If no realm is present in either the NAI or the Destination-Realm, the request is considered to be local.

NOTE: If SBR Carrier receives a request for a realm not listed in the Self Names field, it will attempt to find a matching routing rule to determine how to route the request. For more information about request routing rules, see [“Administering Request Routing Rules” on page 417](#).

To add a local realm for the server using the Web GUI:

1. Select **Diameter Configuration > Local Network Element > Identification**. The **Identification** page ([Figure 130 on page 350](#)) appears.
2. Click **Add** in the **Self Names** area.

The **Add Self Name** dialog box ([Figure 132 on page 353](#)) appears.

Figure 132: Add Self Name Dialog



3. Enter the realm name in the **Name** field.

4. Click **OK**.

The **Self Names** area displays an updated list of realm entries.

To delete an existing local realm using the Web GUI:

1. Select **Diameter Configuration > Local Network Element > Identification**. The **Identification** page ([Figure 130 on page 350](#)) appears.

2. Select the realm name entry that you want to delete.

3. Click **Delete** in the **Self Names** area.

A confirmation dialog box is displayed.

4. Click **Yes** to delete the realm name.

The **Self Names** area displays an updated list of realm entries.

Configuring the Diameter Message Transport

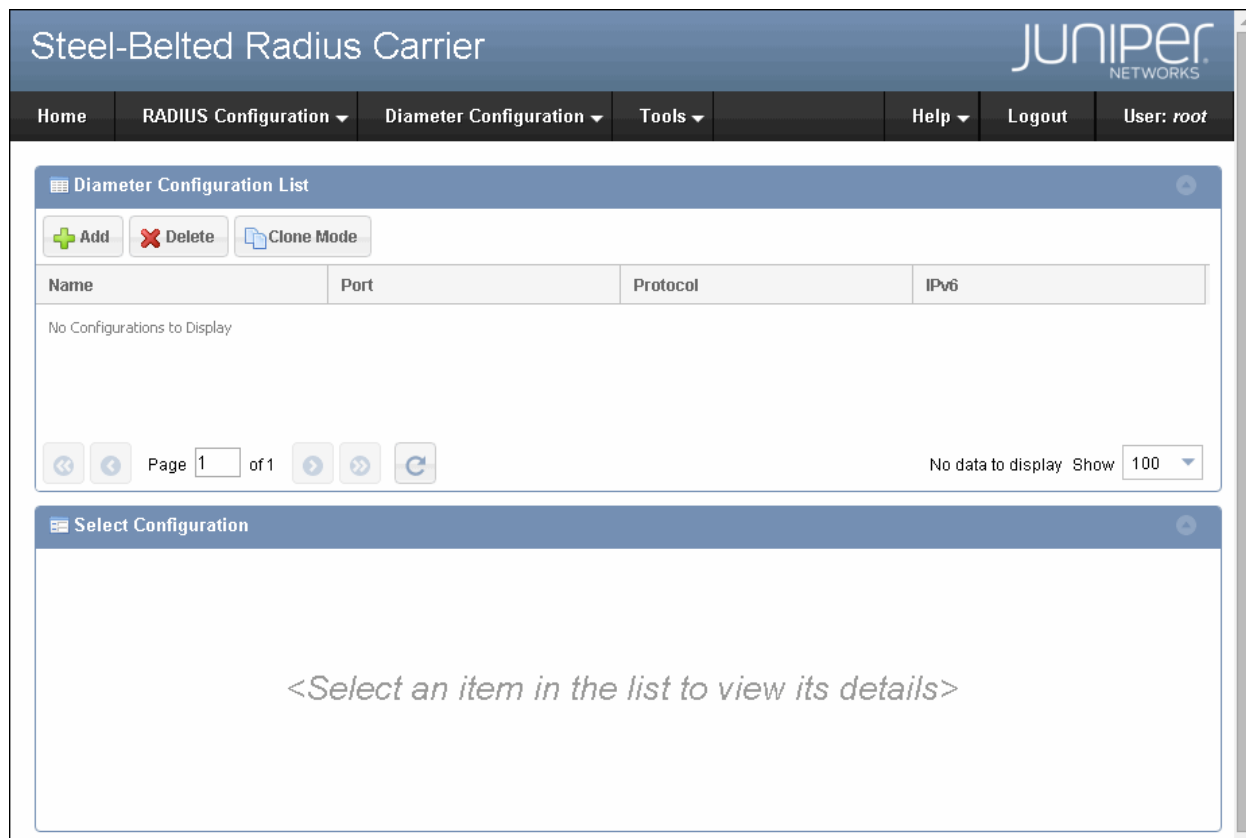
The Diameter message transport configuration includes configuring the transport protocol (either TCP or SCTP), the port number, and the IP address format. SBR Carrier can listen for Diameter messages (authentication messages, authorization messages, and accounting messages) on any port or multiple ports. You can add, edit, or delete Diameter message transports for SBR Carrier.

Adding a Diameter Message Transport Entry

To add a Diameter message transport entry using the Web GUI:

1. Select **Diameter Configuration > Local Network Element > Diameter Configuration**. The **Diameter Configuration List** page ([Figure 133 on page 354](#)) appears.

Figure 133: Diameter Configuration List Page



2. Click **Add**.

The **Create Diameter Configuration** pane ([Figure 134 on page 355](#)) appears.

Figure 134: Create Diameter Configuration Pane

The screenshot shows the 'Steel-Belted Radius Carrier' web interface. The top navigation bar includes 'Home', 'RADIUS Configuration', 'Diameter Configuration', 'Tools', 'Help', 'Logout', and 'User: root'. Below this is a 'Diameter Configuration List' section. The main area is titled 'Create Diameter Configuration' and contains the following fields and controls:

- Name:** A text input field.
- Port:** A numeric input field with up and down arrows.
- Protocol:** A dropdown menu.
- Use IPv6 Networking:** A checkbox.
- Buttons:** 'Save' (with a floppy disk icon), 'Reset' (with a circular arrow icon), and 'Cancel' (with a red X icon).

3. Enter a name for the transport entry in the **Name** field.
4. Enter the port number that the server uses for Diameter messages in the **Port** field.
5. Select either **sctp** or **tcp** from the **Protocol** list to set the transport protocol for Diameter messages.
6. Optionally, select the **Use IPv6 Networking** check box to use IPv6 addressing.

NOTE: SBR Carrier does not support an embedded IPv4 address as an IPv6 address.

7. Click **Save** to save the new transport entry.

The **Diameter Configuration List** page (Figure 133 on page 354) displays an updated list of Diameter message transport entries.

Editing a Diameter Message Transport Entry

To edit an existing Diameter message transport entry using the Web GUI:

1. Select **Diameter Configuration > Local Network Element > Diameter Configuration**. The **Diameter Configuration List** page (Figure 133 on page 354) appears.
2. Select the Diameter message transport entry that you want to edit.

The **Selected Configuration** pane (Figure 135 on page 356) displays the settings configured for the Diameter message transport entry.

Figure 135: Selected Configuration Pane

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and a User: root indicator. The main content area is divided into two panes. The top pane, titled 'Diameter Configuration List', contains buttons for Add, Delete, and Clone Mode, followed by a table with columns for Name, Port, Protocol, and IPv6. The table lists a single entry named 'tcp' with port 1787, protocol tcp, and IPv6 set to false. Below the table are pagination controls showing 'Page 1 of 1' and a 'Show 100' dropdown. The bottom pane, titled 'Selected Configuration: tcp', contains input fields for Name (tcp), Port (1787), and Protocol (tcp), along with a checkbox for 'Use IPv6 Networking' which is unchecked. At the bottom of this pane are buttons for Save, Reset, and Cancel.

Name	Port	Protocol	IPv6
tcp	1787	tcp	false

3. Edit the settings for the Diameter message transport entry as appropriate.

For information about the fields in the **Selected Configuration** pane, see [“Adding a Diameter Message Transport Entry” on page 354](#).

NOTE: You cannot edit the name of the Diameter message transport.

4. Click **Save**.

The **Diameter Configuration List** page (Figure 133 on page 354) displays an updated list of Diameter message transport entries.

Deleting a Diameter Message Transport Entry

To delete an existing Diameter message transport entry using the Web GUI:

1. Select **Diameter Configuration > Local Network Element > Diameter Configuration**. The **Diameter Configuration List** page ([Figure 133 on page 354](#)) appears.
2. Select the transport entry that you want to delete.
3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to delete the transport entry.

The **Diameter Configuration List** page ([Figure 133 on page 354](#)) displays an updated list of Diameter message transport entries.

Administering Diameter Remote Network Elements

IN THIS CHAPTER

- [Remote Network Element Overview | 358](#)
- [Creating and Configuring a New Diameter Remote Network Element | 359](#)
- [Editing a Diameter Remote Network Element | 372](#)
- [Deleting a Diameter Remote Network Element | 375](#)

This chapter describes how to administer Diameter remote network elements. It includes the following sections:

Remote Network Element Overview

A remote network element is a multi-addressable, logical network entity which may host functions. A function refers to an IP Multimedia Subsystem (IMS) function, which means any one of the identified IMS components. An IMS function communicates with other IMS functions exclusively using reference points. Multiple functions can coexist in the same network element.

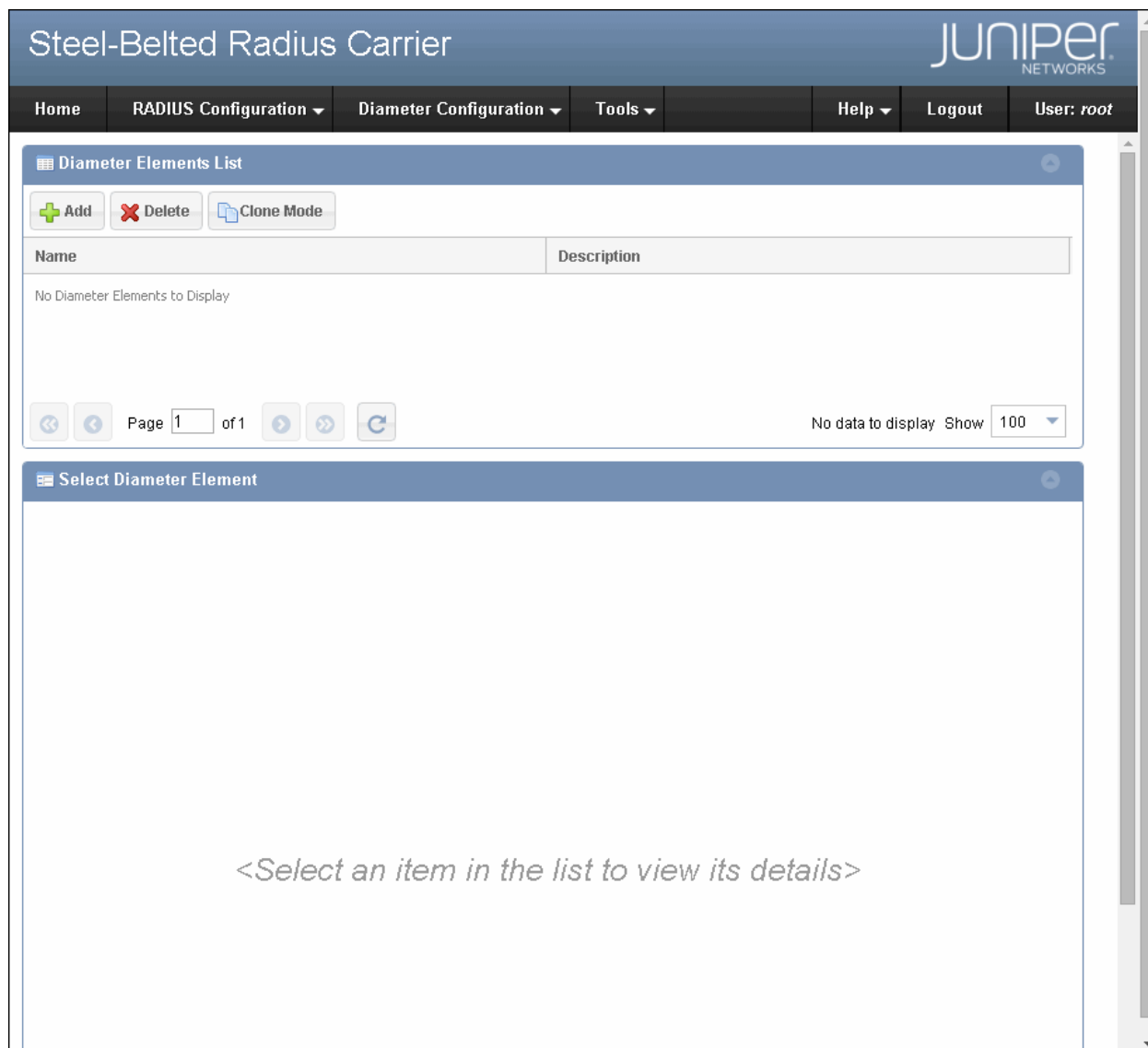
Creating and Configuring a New Diameter Remote Network Element

To create and configure a new Diameter remote network element using the Web GUI:

1. Select **Diameter Configuration > Remote Network Elements > Diameter Elements**.

The **Diameter Elements List** page ([Figure 136 on page 359](#)) appears.

Figure 136: Diameter Elements List Page



2. Click **Add**.

The **Create Network Element** pane ([Figure 137 on page 360](#)) appears.

Figure 137: Create Network Element Pane

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

Diameter Network Elements

Create Network Element

Name:

Description:

Add Diameter Network Elements

Connections

Name	Protocol	Address	Port
No Connections to Display			

Functions

Name	Description
No Functions to Display	

☒ Round Robin ☐ Primary / Backup Add Edit Delete

Save Clear Cancel

3. Enter the name by which the other network elements refer to this Diameter remote network element in the **Name** field.
4. Optionally, enter a description for the Diameter remote network element in the **Description** field. The description you associate with the Diameter remote network element is not used during processing.
5. Add Diameter connections for the new Diameter remote network element. For more information about adding Diameter connections, see [“Adding Diameter Connections to the Diameter Remote Network Element” on page 361](#).
You can modify the Diameter connection entries by using the **Edit** and **Delete** buttons.
6. Define the order of the Diameter connections by selecting each connection and using the **Up** or **Down** arrow. The order specifies the order in which SBR Carrier uses the Diameter connections to send messages to the Diameter remote network element.
7. Select the mode to be used for sending messages over multiple Diameter connections to the Diameter remote network element. The available options are:

- **Round Robin**—SBR Carrier uses Diameter connections in round robin manner as per the ordered list to send messages. If a Diameter connection is not operating then it is excluded from the round robin logic till it starts to work properly.
- **Primary / Backup**—SBR Carrier sends all messages over the first Diameter connection defined in the ordered list. If the first Diameter connection fails, all messages are sent over the next Diameter connection in the ordered list. When the first Diameter connection becomes operational again, then SBR Carrier sends all messages over the first Diameter connection.

NOTE: To use either of these modes, you must have multiple Diameter connections configured for the Diameter remote network element.

8. Assign functions to the new Diameter remote network element. For more information about assigning functions, see [“Assigning Functions to the Diameter Remote Network Element” on page 366](#).

You can modify the function entries by using the **Edit** and **Delete** buttons.

9. Click **Save** to save the new Diameter remote network element.

The **Diameter Elements List** page ([Figure 136 on page 359](#)) displays an updated list of Diameter remote network elements.

Adding Diameter Connections to the Diameter Remote Network Element

SBR Carrier communicates with the remote network element over Diameter connections. You may configure multiple Diameter connections between the server and a remote network element.

Because the Diameter protocol is peer-to-peer, a single, bidirectional Diameter connection carries both incoming and outgoing messages. The Web GUI lets you configure an ordered collection of connections for the remote peers that form the remote network element. This list of remote peer connections is part of the network element. For example, a multi-homed server that presents a different Origin-Host for each of its network connections could be represented as a network element consisting of multiple connections to Diameter peers.

[Table 39 on page 361](#) summarizes the parameters for configuring Diameter connections.

Table 39: Diameter Connection Parameters

Field	Description
Name	The name of the Diameter connection in SBR Carrier.

Table 39: Diameter Connection Parameters (*continued*)

Field	Description
Description	A description of the connection (optional). The description you associate with a connection is not used during processing.
Host Name	The Origin-Host name of the remote peer.
Realm Name	The Origin-Realm name of the remote peer.
Require Source IP Match for Connection	This option determines if the source IP address of a connection attempt must match one of the configured IP addresses used to connect to this peer. If this is not selected, traffic will be accepted from any IP address as long as the client presents the correct hostname during the capabilities exchange. This functionality allows other peers to exist behind Network Address Translation (NAT) devices.
IP Address	The IP address used for the connection.
Use IPv6 Networking	Enable this option if you are using IPv6 addressing. Leave it disabled if you are using IPv4 addressing. NOTE: SBR Carrier does not support an embedded IPv4 address as an IPv6 address.
Port	The port number used by the remote peer for Diameter messages.
Active	If this option is selected, the server periodically attempts to connect (or reconnect after a connection has failed) to the remote peer. If this option is not selected, a connection is established only after the remote peer actively connects to the server.

Table 39: Diameter Connection Parameters (*continued*)

Field	Description
TCP / SCTP	<p>Selects the protocol used to carry Diameter messages to the remote peer.</p> <p>If set to TCP, Diameter messages are sent using TCP.</p> <p>If set to SCTP, Diameter messages are sent using Stream Control Transmission Protocol (SCTP). This allows you to use multiple IP addresses for a Diameter connection. When the remote peer is running SCTP, SBR Carrier accepts packets from any of its IP addresses (that are defined in the Additional IP addresses field).</p>
Additional IP addresses	<p>An ordered set of IP addresses to use for a multi-link connection, in addition to the IP address specified in the IP Address field. This feature can only be enabled when using SCTP.</p>

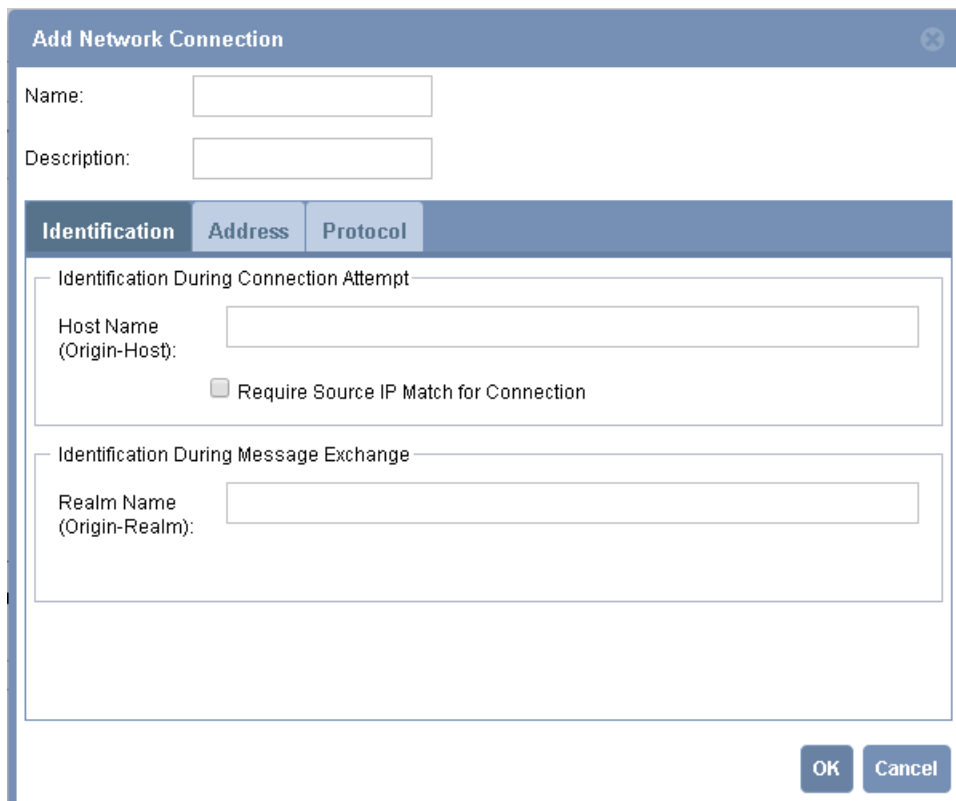
To add a Diameter connection for a Diameter remote network element using the Web GUI:

1. In the **Create Network Element** pane ([Figure 137 on page 360](#)), click **Add** in the **Connections** area.

The **Add Network Connection** dialog box appears ([Figure 138 on page 364](#)) with the **Identification** tab selected.

For more information about the Diameter connection parameters, see [Table 39 on page 361](#).

Figure 138: Add Network Connection Dialog—Identification



The image shows a web-based dialog box titled "Add Network Connection". It has a blue header bar with the title and a close button. Below the header, there are two text input fields: "Name:" and "Description:". Below these fields is a tabbed interface with three tabs: "Identification" (which is selected and highlighted in blue), "Address", and "Protocol". Under the "Identification" tab, there are two sections. The first section is titled "Identification During Connection Attempt" and contains a text input field for "Host Name (Origin-Host):" and a checkbox labeled "Require Source IP Match for Connection". The second section is titled "Identification During Message Exchange" and contains a text input field for "Realm Name (Origin-Realm):". At the bottom right of the dialog box, there are two buttons: "OK" and "Cancel".

2. Enter a name for the connection in the **Name** field.
3. Optionally, enter a description for the connection in the **Description** field. The description is not used during processing.
4. Enter the Origin-Host name of the remote peer in the **Host Name (Origin-Host)** field.
5. Optionally, select the **Require Source IP match for Connection** check box if you want to require the source IP address of a connection attempt to match one of the configured IP addresses used to connect to the peer.

6. Enter the Origin-Realm name of the remote peer in the **Realm Name (Origin-Realm)** field.
7. Click the **Address** tab (Figure 139 on page 365) and enter the IP address of the remote peer in the **IP Address** field.

Figure 139: Add Network Connection Dialog—Address

The screenshot shows a dialog box titled "Add Network Connection". At the top, there are two text input fields labeled "Name:" and "Description:". Below these is a tabbed interface with three tabs: "Identification", "Address", and "Protocol". The "Address" tab is currently selected. Inside the "Address" tab, there is a section titled "Address Used for Connection Attempt". This section contains three fields: "IP Address:" with a text input field, "Port:" with a spinner box showing "3868", and a "Use IPv6" checkbox which is unchecked. There is also an "Active" checkbox which is checked. At the bottom right of the dialog box are "OK" and "Cancel" buttons.

8. Optionally, select the **Use IPv6** check box to use IPv6 addressing.

NOTE: SBR Carrier does not support an embedded IPv4 address as an IPv6 address.

9. Enter the port number that the remote peer uses for Diameter messages in the **Port** field.
10. Optionally, select the **Active** check box if you want the server to actively try to connect to the remote peer.
11. Click the **Protocol** tab (Figure 140 on page 366) to specify the transport protocol used to carry Diameter messages.

- Select the **TCP** check box to use the TCP protocol for Diameter messages.
- Select the **SCTP (Default)** check box to use the SCTP protocol for Diameter messages and click **Add** to add an ordered set of IP addresses used for the multi-link connection (in addition to the IP address specified in the **IP Address** field under the **Address** tab).

You can modify the ordered set of IP addresses by using the **Edit** and **Delete** buttons.

Figure 140: Add Network Connection Dialog—Protocol

The screenshot shows the 'Add Network Connection' dialog box with the 'Protocol' tab selected. At the top, there are input fields for 'Name:' and 'Description:'. Below these are three tabs: 'Identification', 'Address', and 'Protocol'. The 'Protocol' tab contains a section titled 'Protocol Used to Carry Diameter Messages' with two radio buttons: 'SCTP (Default)' (which is selected) and 'TCP'. Below this is a section titled 'Additional Addresses for Multi-Link' containing a list box with the header 'Address' and the text 'No Addresses to Display'. To the right of the list box are three buttons: 'Add', 'Edit', and 'Delete'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

12. Click **OK** to save the connection.

The new Diameter connection entry is now displayed in the **Connections** area in the **Create Network Element** pane ([Figure 137 on page 360](#)).

Repeat these steps to add multiple Diameter connections for a Diameter remote network element.

Assigning Functions to the Diameter Remote Network Element

The function refers to an IMS function, which means any one of the identified IMS components. An IMS function communicates with other IMS functions exclusively using reference points. Multiple functions can coexist in the same network element.

For example, you might configure a remote network element and assign only a single function to it, such as HSS. On the other hand, you could create a single remote network element and assign multiple functions to it such as the Downstream server and non-3GPP network access functions.

[Table 40 on page 367](#) provides a description of each of the functions supported by SBR Carrier.

Table 40: Diameter Functions Support in SBR Carrier

Function	Description
Non-3GPP Network	The non-3GPP network function is an AAA source for the direct IP service provided by the non-3GPP network infrastructure.
Downstream	<p>This function is assigned to other 3GPP servers to which this server may forward (proxy) requests.</p> <p>For example, the HSS may indicate that the request needs to be redirected to another 3GPP server; in which case the local 3GPP server must proxy the request to the remote 3GPP server. The remote 3GPP server would be assigned the Downstream function. Another example would be when a subscriber's identity or realm decoration indicates roaming, the 3GPP server may need to proxy the request to another 3GPP server outside the home public land mobile network (HPLMN).</p>
HSS	The HSS is a Diameter-based subscriber and policy database used in 3GPP networks that implement IMS R6 or later. SBR Carrier downloads and caches both user credentials and service subscription data (profile data) from the HSS. In addition, SBR Carrier coordinates with other SBR Carrier servers through a registration mechanism in the HSS. The first SBR Carrier server that authenticates a particular subscriber is registered as the responsible server in the HSS. Any subsequent authentications for that subscriber will be redirected to the responsible server, as long as it remains registered. After the subscriber has left the network, the registration may be purged by the SBR Carrier server or the HSS, meaning it is no longer responsible and the next server to authenticate this subscriber may take over. The reference point between SBR Carrier and the HSS is SWx.
ePDG	The ePDG function performs authentication and authorization of tunnel requests, when tunneled IP service (3GPP IP service) is being provided to an untrusted non-3GPP IP access network.
PDG	The PDG function performs authentication and authorization of tunnel requests, when tunneled IP service (3GPP IP Service) is being provided to a non-3GPP IP access network.

Some functions require you to configure routing rules based on either the subscriber identity, known as the IMSI or the realm name. These routing rules are called implicit routing rules. For instance, when you

assign the HSS function to a Diameter remote network element, you need to specify which subscribers are served by the HSS.

To assign a function to a Diameter remote network element using the Web GUI:

1. In the **Create Network Element** pane (Figure 137 on page 360), click **Add** in the **Functions** area

The **Add Network Function** dialog box (Figure 141 on page 368) appears with the **Non-3GPP Network** option selected.

Figure 141: Add Network Function Dialog

The dialog box is titled "Add Network Function". It is divided into two main sections. On the left, under the heading "Select Network Function", there is a list of radio buttons: "Non-3GPP Network" (which is selected), "HSS", "ePDG", "PDG", and "Downstream". On the right, under the heading "New Function - Non-3GPP", there is a text area containing the following text: "The Non-3GPP Network function is a source for authentication, authorization, and accounting for Direct IP service provided by the Non-3GPP Network infrastructure." Below this text, it says "You may supply a description of this particular WLAN element below." followed by a "Description:" label and a text input field containing the text "Non-3GPP Network". At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

2. Select the function that you want to assign to the Diameter remote network element.

The right-side pane displays a brief description of the corresponding function and a **Description** field. If the function supports implicit routing, the respective configuration tabs will also be displayed in the right-side pane. Figure 142 on page 369 shows an example dialog for the HSS function.

Figure 142: Add Network Function Dialog—HSS

The screenshot shows the 'Add Network Function' dialog box. On the left, under 'Select Network Function', the 'HSS' option is selected. The main area is titled 'New Function - HSS' and contains the following text: 'The Home Subscriber Server is a Diameter-based user and policy database supporting IMS R6 or greater.' and 'You may supply a description of this particular HSS element below.' Below this is a 'Description:' field with the text 'Home Subscriber Server'. There are two tabs: 'IMSI Routing' and 'Realm Routing', with 'Realm Routing' currently selected. Under 'IMSI Prefixes Routed to the Remote HSS:', there is a table with one header 'IMSI Prefix' and one row containing the text 'No IMSI Prefixes to Display'. At the bottom of the dialog, there is a checkbox labeled 'Default Routing Rule for Local Users' which is checked, and three buttons: 'Add', 'Edit', and 'Delete'. At the very bottom right of the dialog are 'OK' and 'Cancel' buttons.

3. Optionally, enter a description for the function in the **Description** field.
4. Configure implicit routing rules, if the function supports implicit routing. For more information about implicit routing rules configuration, see [“Configuring Implicit Routing Rules” on page 369](#).
You can modify the prefix and realm routing entries by using the **Edit** and **Delete** buttons.
5. Click **OK**.

The new function entry is now displayed in the **Functions** area in the **Create Network Element** pane ([Figure 137 on page 360](#)).

Repeat these steps to assign multiple functions to a Diameter remote network element.

Configuring Implicit Routing Rules

Implicit routing is based on subscriber identity or realm. In non-3GPP networks, the IMSI of the mobile device is used as the subscriber identity.

Table 41 on page 370 shows the functions that use implicit routing rules, and the type of routing rules used by each function.

Table 41: Functions that Use Implicit Routing Rules

Function	IMSI Prefix Routing	Realm Routing
HSS	✓	✓
Downstream	-	✓

SBR Carrier uses the following priorities to process implicit routing rules:

1. IMSI
2. Realm
3. Default IMSI route (default route for local users)
4. Default Realm route (default route for all users from all realms)

For example, if both IMSI and realm routing rules are defined for the function, IMSI routing rules take priority over realm routing rules.

NOTE: Any explicit routing rules defined for a function, take priority over implicit routing rules. For more information about explicit routing rules, see [“Configuring Request Routing Rules” on page 419](#).

Configuring IMSI Routing Rules

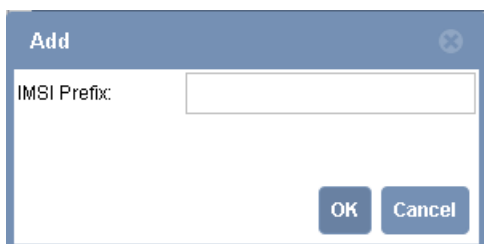
IMSI prefix routing allows you to specify which IMSI prefix numbers are associated with the remote peer (function). For example, entering 3000 for the IMSI prefix routing for an HSS instructs SBR Carrier to use this HSS for all requests that have an IMSI beginning with 3000.

To configure IMSI routing rules using the Web GUI:

1. Click the **IMSI Routing** tab in the **Add Network Function** dialog box. For a sample **Add Network Function** dialog box, see [Figure 142 on page 369](#).
2. Define IMSI routing rule.
 - Select the **Default Routing Rule for Local Users** check box to assign this function to all local subscribers.
 - or
 - a. Click **Add**.

The **Add** dialog box ([Figure 143 on page 371](#)) is displayed.

Figure 143: Add Dialog



- b. Enter a number in the **IMSI Prefix** field and click **OK**. The **IMSI Routing** tab of the **Add Network Function** dialog box (for a sample **Add Network Function** dialog box, see [Figure 142 on page 369](#)) displays the updated list of IMSI prefix routing entries.

Repeat these steps to create multiple prefix routing entries.

Configuring Realm Routing Rules

The realm routing allows you to specify which realms are routed to the remote peer (function). When a request is received, the server examines the NAI decoration to determine the realm to which the request should be routed. If the request does not contain an NAI decoration, the Destination-Realm in the request is used. For example, if you entered *XYZ.com* under the realm routing rule for an HSS function, SBR Carrier would retrieve subscriber credentials from this HSS for processing any requests with an NAI decoration that includes *XYZ.com*.

To configure the realm routing rules using the Web GUI:

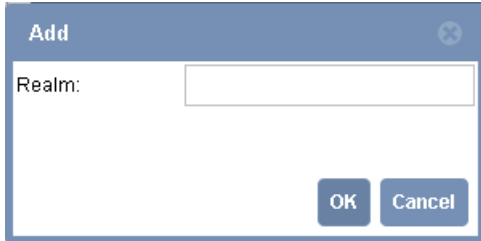
1. Click the **Realm Routing** tab in the **Add Network Function** dialog box. For a sample **Add Network Function** dialog box, see [Figure 142 on page 369](#).
2. Define realm routing rule.
 - Select the **Default Routing Rule for All Users from All Realms** check box to route the requests from all realms to the remote peer.

or

 - a. Click **Add**.

The **Add** dialog box ([Figure 144 on page 372](#)) is displayed.

Figure 144: Add Dialog

A small dialog box titled "Add" with a close button in the top right corner. It contains a label "Realm:" followed by a text input field. At the bottom right, there are two buttons: "OK" and "Cancel".

- b. Enter the realm name in the **Realm** field and click **OK**. The server routes only requests containing the provided real name to the remote peer.

The **Realm Routing** tab of the **Add Network Function** dialog box (for a sample **Add Network Function** dialog box, see [Figure 142 on page 369](#)) displays the updated list of realm routing entries.

Repeat these steps to create multiple realm routing entries.

Editing a Diameter Remote Network Element

To edit an existing Diameter remote network element using the Web GUI:

1. Select **Diameter Configuration > Remote Network Elements > Diameter Elements**.

The **Diameter Elements List** page ([Figure 136 on page 359](#)) appears.

2. Select the Diameter remote network element entry that you want to edit.

The **Selected Network Element** pane ([Figure 145 on page 373](#)) displays the settings configured for the Diameter remote network element.

Figure 145: Selected Network Element Pane

Steel-Belted Radius Carrier

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

Diameter Elements List

+ Add - Delete Clone Mode

Name	Description
Diameter Element 1	Example configuration

Page 1 of 1 Displaying 1 - 1 of 1 Show 100

Selected Diameter Element: Diameter Element 1

Name: Diameter Element 1

Description: Example configuration

Diameter Network Elements

Connections

Name	Protocol	Address	Port
No Connections to Display			

Round Robin Primary / Backup Add Edit Delete

Functions

Name	Description
No Functions to Display	

Add Edit Delete

Save Reset Cancel

3. Edit the existing description in the **Description** field.

NOTE: You cannot edit the name of the Diameter remote network element.

4. In the **Connections** area, you can:
 - Click **Add** to configure and add a new Diameter connection. The **Add Network Connection** dialog box (Figure 138 on page 364) appears. For more information about the fields in the **Add Network Connection** dialog box, see “Adding Diameter Connections to the Diameter Remote Network Element” on page 361.
 - Select a Diameter connection entry and click **Edit** to edit the existing connection. The **Edit Network Connection** dialog box similar to the **Add Network Connection** dialog box (Figure 138 on page 364) appears. For more information about the fields in the **Edit Network Connection** dialogs, see “Adding Diameter Connections to the Diameter Remote Network Element” on page 361.

NOTE: You cannot edit the name of the Diameter connection.

- Select a Diameter connection entry and click **Delete** to delete the connection.
- Change the mode to be used for sending messages over multiple Diameter connections to the Diameter remote network element. The available options are:
 - **Round Robin**
 - **Primary / Backup**

NOTE: To use either of these modes, you must have multiple Diameter connections configured for the Diameter remote network element.

- Reorder the Diameter connections by selecting each connection and using the **Up** or **Down** arrow.
- The **Connections** area displays an updated list of Diameter connections entries.

5. In the **Functions** area, you can:

- Click **Add** to configure and add a new function. The **Add Network Function** dialog box (Figure 141 on page 368) appears. For more information about the fields in the **Add Network Function** dialog box, see [“Assigning Functions to the Diameter Remote Network Element” on page 366](#).
- Select a function entry and click **Edit** to edit the existing function. The **Edit Network Function** dialog box similar to the **Add Network Function** dialog box Figure 141 on page 368 appears. For more information about the fields in the **Edit Network Function** dialog box, see [“Assigning Functions to the Diameter Remote Network Element” on page 366](#).
- Select a function entry and click **Delete** to delete the function.

The **Functions** area displays an updated list of function entries.

6. Click **Save** to save the changes.

The **Diameter Elements List** page (Figure 136 on page 359) displays an updated list of Diameter remote network elements.

Deleting a Diameter Remote Network Element

To delete a Diameter remote network element using the Web GUI:

1. Select **Diameter Configuration > Remote Network Elements > Diameter Elements**.

The **Diameter Elements List** page ([Figure 136 on page 359](#)) appears.

2. Select the remote network element entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to delete the remote network element.

The **Diameter Elements List** page ([Figure 136 on page 359](#)) displays an updated list of Diameter remote network elements.

Administering the Diameter Policy

IN THIS CHAPTER

- [Policy Overview | 376](#)
- [Configuring a Local Profile | 376](#)
- [Configuring Local Profile Selection | 408](#)

This chapter provides a high level discussion of the SBR Carrier policy capabilities which include defining local profiles and profile selection. It also presents how to administer a Diameter policy in SBR Carrier.

This chapter covers the following topics:

Policy Overview

In non-3GPP networks, authorization is based on subscriber data provided by the HSS. SBR Carrier can enforce fine-grained authorization policy by using local profiles. The local profiles are used in combination with the subscriber profile stored in the HSS to provide additional security for authorization. During the authorization process, SBR Carrier uses user-defined profile selection rule sets to select the local profile used to authorize the request.

Configuring a Local Profile

SBR Carrier can enforce fine-grained authorization policy by using local profiles. Local profiles are used in combination with the subscriber profile stored in the HSS to provide additional security for authorization. A local profile is configured in the Web GUI by defining ordered lists of attributes and a non-3GPP interworking policy. You can define as many local profiles as you require.

NOTE: Each server in your network needs to be configured with at least one local profile that includes the policy configuration for non-3GPP IP access.

A mobile operator can use profiles to create a wireless plan for subscribers. For example, you might offer a voice only plan as well as a voice and data plan. A profile could be defined for each plan and customized for the specific requirements of the SWa, STa, SWm, and S6b reference points supported in the plan.

Creating a Local Profile

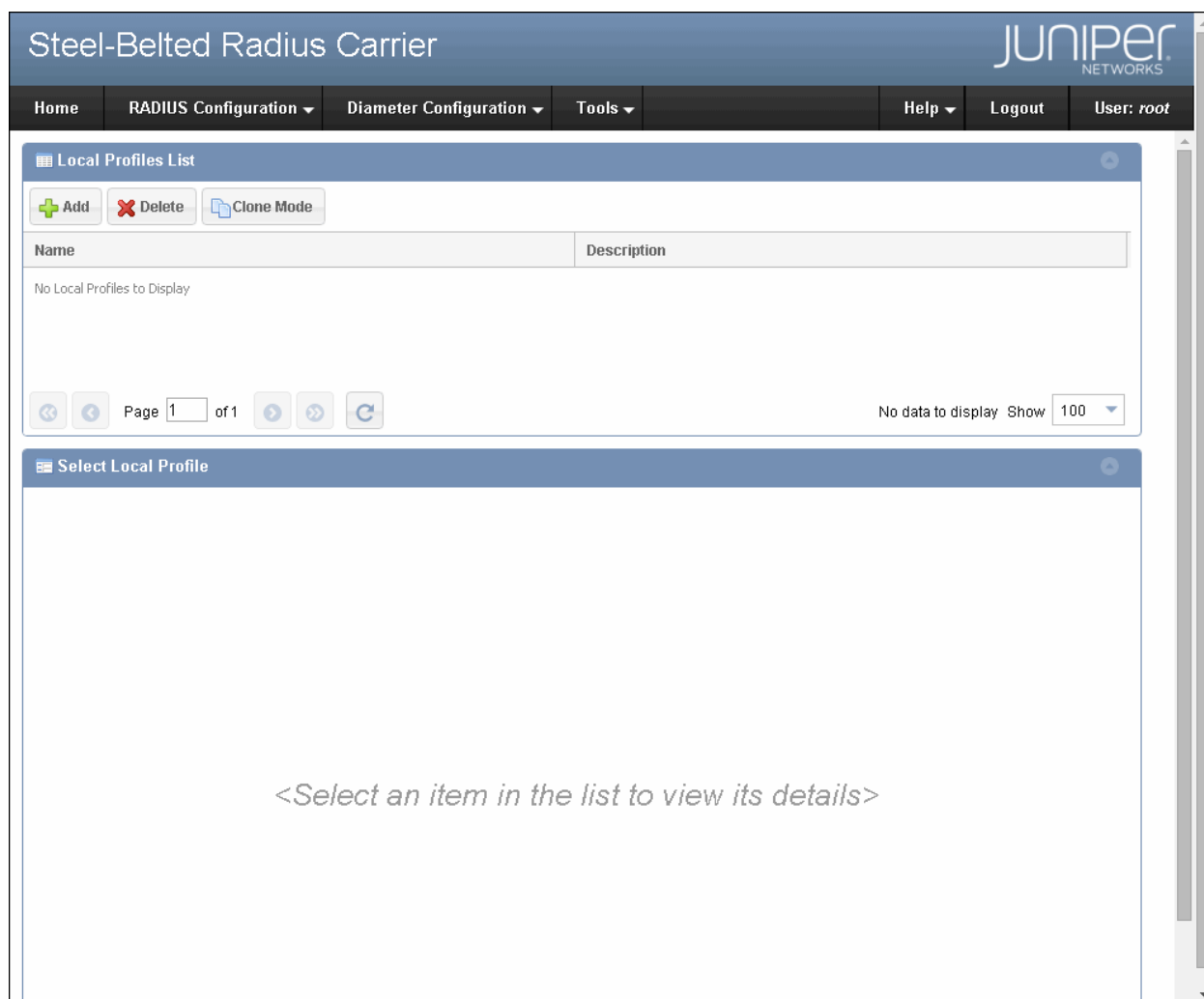
To create a local profile, you configure attribute lists and a non-3GPP interworking policy for the SWa, STa, SWm, and S6b reference points. Attribute lists specify attribute values that are placed in the response, and are typically used to provide additional parameters needed to complete the authorization. The order in which the attributes are placed in the response is user-defined. The non-3GPP interworking policy is a list of rules governing the authorization.

To create a local profile using the Web GUI:

1. Select **Diameter Configuration > Policy > Local Profiles**.

The **Local Profiles List** page ([Figure 146 on page 378](#)) appears.

Figure 146: Local Profiles List Page



2. Click **Add**.

The **Create Local Profile** pane ([Figure 147 on page 379](#)) appears.

Figure 147: Create Local Profile Pane

Steel-Belted Radius Carrier

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

Local Profiles List

Add Delete Clone Mode

Name	Description
No Local Profiles to Display	

Page 1 of 1 No data to display Show 100

Create Local Profile

Name:

Description:

SWa/STa Interface SWm Interface S6b Interface

Attributes Non-3GPP Interworking Policy

Name	Description
No Ordered SWa/STa Attributes to Display	

Add Edit Delete

Save Reset Cancel

3. Enter a name for the local profile in the **Name** field.
4. Optionally, enter a description for the local profile in the **Description** field.
The description is not used during processing.
5. Configure authorization attributes for the SWa, STa, SWm, and S6b reference points. For more information about configuring authorization attributes, see [“Configuring Authorization Attributes” on page 380](#).
You can modify authorization attribute entries by using the **Edit** and **Delete** buttons.
6. Define the order of authorization attribute entries by selecting each entry and using the **Up** or **Down** arrow.
7. Configure a non-3GPP interworking policy for the SWa, STa, SWm, and S6b reference points. For more information about configuring the non-3GPP interworking policy for the SWa or STa reference point,

see [“Configuring a Non-3GPP Interworking Policy for SWa or STa Reference Point” on page 386](#). For more information about configuring the non-3GPP interworking policy for the SWm reference point, see [“Configuring a Non-3GPP Interworking Policy for SWm Reference Point” on page 389](#). For more information about configuring the non-3GPP interworking policy for the S6b reference point, see [“Configuring a Non-3GPP Interworking Policy for S6b Reference Point” on page 398](#).

You can modify non-3GPP interworking policy entries by using the **Edit** and **Delete** buttons.

8. Define the order of non-3GPP interworking policy entries by selecting each entry and using the **Up** or **Down** arrow.
9. Click **Save** to save the profile configuration.

The **Local Profiles List** page ([Figure 146 on page 378](#)) displays an updated list of local profiles.

Configuring Authorization Attributes

SBR Carrier supports the following authorization attribute lists:

- Return list
- Copy list
- Periodic reauthorization

You can create any combination of the above attribute authorization lists you require. For example, you can configure the profile with both return lists and copy lists. This allows the visited SBR Carrier server to add its own level of authorization by copying some attributes from the server response, as well as setting some attributes in the response.

The value of each attribute has a well-defined data type: numeric, string, IP address, time, or hexadecimal. For example, Callback-Number is of type string and contains a telephone number, while NAS-Port-Type is an item from a list, and can be Sync, Async, and so forth.

NOTE: SBR Carrier supports signed integers (negative numbers) for attributes received in packets. However, the Web GUI does not support signed integers, and treats both signed and unsigned integers as unsigned integers.

Attributes can be single- or multi-valued. Single-valued attributes appear at most once in the response; multi-valued attributes may appear several times.

If an attribute appears more than once in the response list, each value of the attribute is sent as part of the response packet. For example, to enable both IP and IPX header compression for a subscriber, the Framed-Compression attribute should appear twice in the response list: once with the value VJTCPIPheadercompression and once with the value IPXheadercompression.

The order in which attributes appear in the response is user defined. In addition, an attribute can appear more than once in the response, and the order in which the attributes appear may be significant.

For Diameter, if the request from the client included a request for authorization, a successful response must include the authorization AVPs that are relevant to the service being provided.

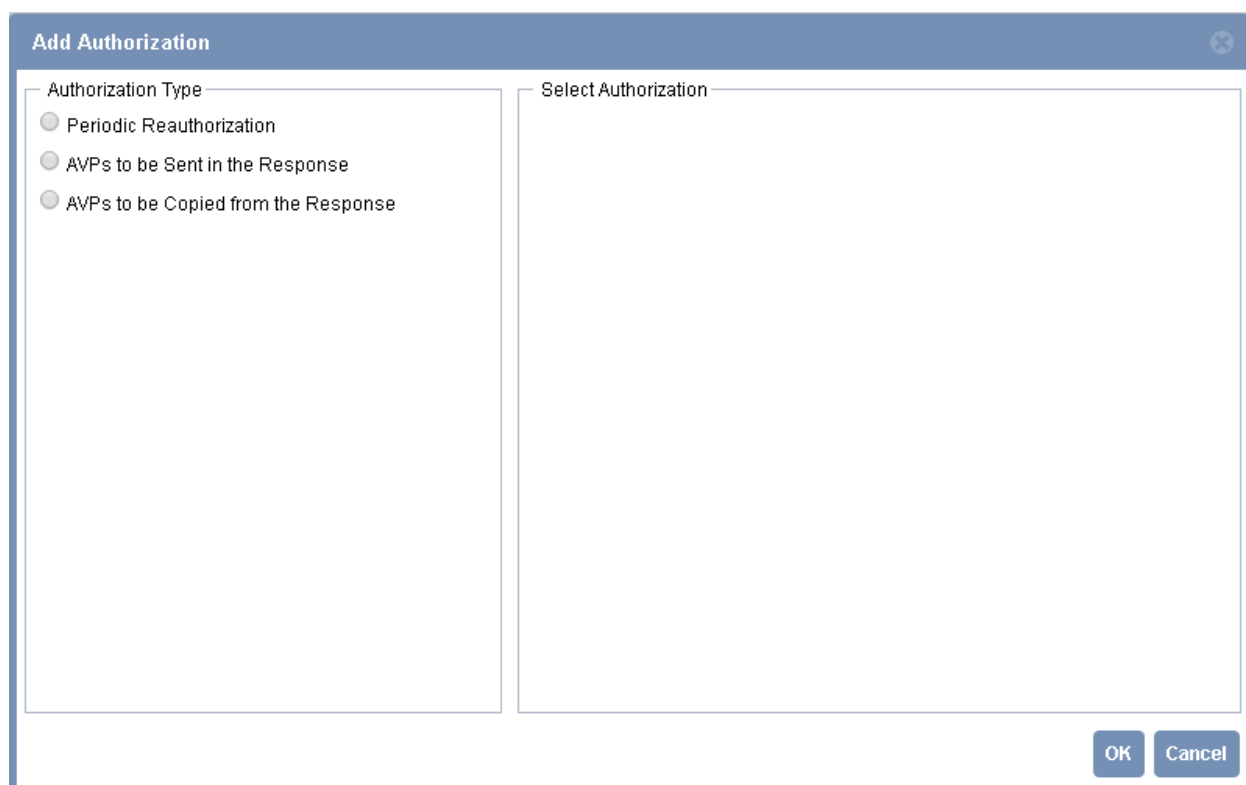
Attributes are always returned to the client (non-3GPP network) in the authorization response only when authorization is granted.

To configure authorization attributes for the SWa, STa, SWm, or S6b reference point using the Web GUI:

1. In the **Create Local Profile** pane ([Figure 147 on page 379](#)), click **Attributes** under the respective tab.
2. Click **Add**.

The **Add Authorization** dialog box for authorization attributes ([Figure 148 on page 381](#)) appears.

Figure 148: Add Authorization Dialog for Authorization Attributes



The image shows a dialog box titled "Add Authorization". It has a blue header bar with the title and a close button. The dialog is divided into two main sections. The left section, titled "Authorization Type", contains three radio button options: "Periodic Reauthorization", "AVPs to be Sent in the Response", and "AVPs to be Copied from the Response". The right section, titled "Select Authorization", is a large empty rectangular area. At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

In the **Add Authorization** dialog box, you can:

- Select the **Periodic Reauthorization** option to add periodic reauthorization attributes. For more information about adding periodic reauthorization attributes, see [“Adding Periodic Reauthorization Attributes” on page 382](#).

- Select the **AVPs to be Sent in the Response** option or the **AVPs to be Copied from the Response** option to add a return or copy list. For more information about adding return or copy list, see [“Adding a Return or Copy List” on page 383](#).

Adding Periodic Reauthorization Attributes

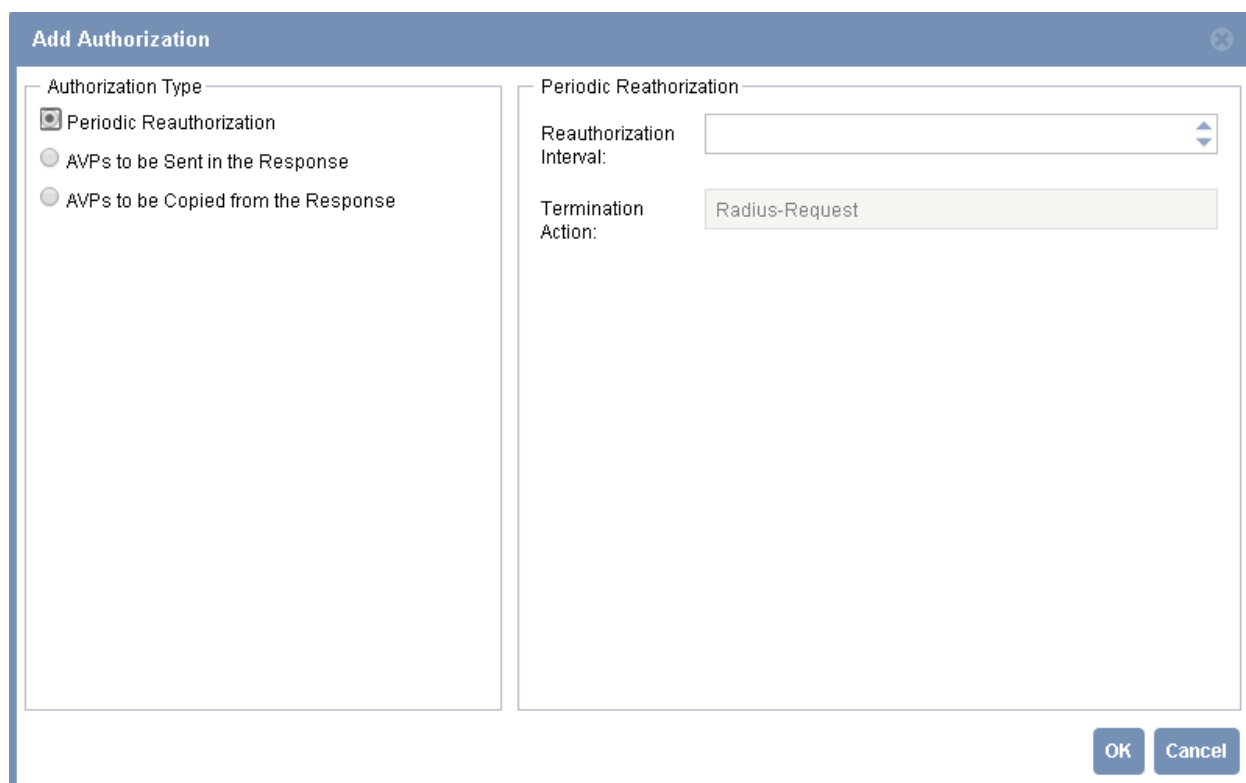
The periodic reauthorization feature provides a set of attributes that instruct the non-3GPP network to initiate reauthorization requests automatically. SBR Carrier uses two attributes for this instruction: Session-Timeout and Termination-Action. The Session-Timeout attribute value is configurable (as the Reauthorization Interval) in seconds. This attribute sets the maximum time of service to be provided to the subscriber. The server sends this attribute to the client in the response. The value of the Termination-Action attribute instructs the non-3GPP network to perform what is necessary after Session-Timeout elapses.

To add periodic reauthorization attributes for the selected reference point using the Web GUI:

1. In the **Add Authorization** dialog box for authorization attributes ([Figure 148 on page 381](#)), select the **Periodic Reauthorization** option.

The **Periodic Reauthorization** pane ([Figure 149 on page 382](#)) appears.

Figure 149: Add Authorization Dialog—Periodic Reauthorization



The screenshot shows a web-based dialog box titled "Add Authorization". It has a blue header bar with a close button in the top right corner. The dialog is divided into two main sections. The left section, titled "Authorization Type", contains three radio button options: "Periodic Reauthorization" (which is selected), "AVPs to be Sent in the Response", and "AVPs to be Copied from the Response". The right section, titled "Periodic Reauthorization", contains two fields: "Reauthorization Interval:" with a text input field and a vertical spinner, and "Termination Action:" with a dropdown menu currently showing "Radius-Request". At the bottom right of the dialog are "OK" and "Cancel" buttons.

2. Enter a value (in seconds) in the **Reauthorization Interval** field.

This value equates to the **Session-Timeout** attribute and specifies the maximum time of service to be provided to the subscriber.

NOTE: The **Termination Action** field displays the value of the **Termination-Action** attribute that instructs the non-3GPP network to perform what is necessary after the session timeout elapses. This field is a read-only field.

3. Click **OK**.

The **Attributes** tab in the **Create Local Profile** pane ([Figure 147 on page 379](#)) displays the updated attributes list.

Adding a Return or Copy List

The attributes to be set in the response feature allows you to specify which attributes you want to set in the response sent to the client. This feature always replaces all instances of a particular attribute with its locally configured attribute. Only one locally configured value of any attribute can exist.

To create an attribute return list, you select the attributes you want to return in the response from a predefined list. For each attribute, the Web GUI prompts you to enter values using familiar data types such as string, integer, or network address. You then place the attributes in the order in which you want them to be sent in the response.

By including appropriate attributes in the response list, you can create a variety of connection policies. For example, particular IP addresses can be assigned to specific subscribers, IP header compression can be turned on or off, or a time limit can be assigned to the connection.

The attributes to be copied from downstream feature allows you to specify which attributes you want to copy from a downstream server response. This feature always adds all instances of a particular attribute from the downstream server response to the response sent to the client.

For example, if SBR Carrier in a visited public land mobile network (VPLMN) is accessed by a roaming subscriber, the subscriber's home SBR Carrier server is responsible for performing session authorization. When SBR Carrier in the VPLMN receives the request from the roaming subscriber, it proxies the request to the subscriber's home server (HPLMN). The home server performs the authorization and returns the response to the SBR Carrier server in the VPLMN. SBR Carrier uses the list of attributes to be copied from downstream to select which attributes are copied from the HSS or home server response, and in what order they are to be placed in the final response it sends to the non-3GPP network.

To create an attribute copy list, you select the attributes you want to copy from the downstream server response from a predefined list. You do not enter values for the attributes; the values are copied from the downstream server response. You then place the attributes in the order in which you want them to be sent in the response.

NOTE: Only attributes that are in the original response from the downstream server may be copied. If the ordered list includes an attribute that is not in the response from the downstream server, the attribute is not copied in the final response to the client.

To add a return or copy list to the SWa, STa, SWm, or S6b reference point using the Web GUI:

1. In the **Add Authorization** dialog box for authorization attributes ([Figure 148 on page 381](#)), select either the **AVPs to be Sent in the Response** or **AVPs to be Copied from the Response** option.

The **AVPs to be Sent in the Response** or **AVPs to be Copied from the Response** pane appears.

[Figure 150 on page 384](#) shows a sample **AVPs to be Sent in the Response** pane.

Figure 150: Add Authorization Dialog—AVPs to be Set in the Response

Add Authorization

Authorization Type

- ☐ Periodic Reauthorization
- ☒ AVPs to be Sent in the Response
- ☐ AVPs to be Copied from the Response

AVPs to be Sent in the Response

Name	Value
------	-------

Add Edit Delete

OK Cancel

2. Click **Add**.

The **Add AVP** dialog box ([Figure 151 on page 385](#)) appears.

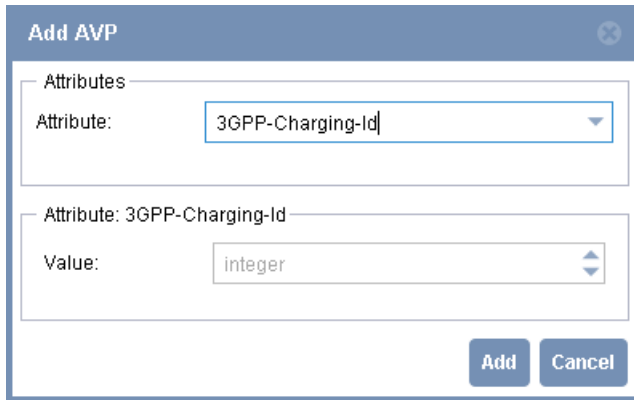
Figure 151: Add AVP Dialog



The 'Add AVP' dialog box has a title bar with a close button. It contains two sections: 'Attributes' and 'Select Attribute'. The 'Attributes' section has a label 'Attribute:' followed by a dropdown menu. The 'Select Attribute' section is an empty rectangular box. At the bottom right are 'Add' and 'Cancel' buttons.

3. Select the attribute that you want to add from the **Attributes** list. [Figure 152 on page 385](#) shows a sample **Add AVP** dialog box for the **3GPP-Charging-Id** attribute.

Figure 152: Sample Add AVP Dialog



This is a sample 'Add AVP' dialog box. The 'Attributes' section shows the 'Attribute:' dropdown menu with '3GPP-Charging-Id' selected. The 'Select Attribute' section now contains a sub-section titled 'Attribute: 3GPP-Charging-Id' with a 'Value:' dropdown menu showing 'integer'. The 'Add' and 'Cancel' buttons are at the bottom right.

4. Select or enter a value for the selected attribute in the **Value** field.

The **Add AVP** dialog box changes according to the type of attribute list you are creating. The **AVPs to be Sent in the Response** option requires that you enter or select a value, string, or IP address. For the **AVPs to be Copied from the Response** option, you just select the attribute to be copied from the attributes list.

5. Click **Add**.
6. Repeat steps 3 through 5 to assign more attributes to the return or copy list.
7. When you are finished adding AVPs, click **Cancel**.

The **AVPs to be Sent in the Response** pane displays the updated list of selected attributes. For a sample **AVPs to be Sent in the Response** dialog box, see [Figure 150 on page 384](#). You can modify the return or copy list by using the **Edit** and **Delete** buttons.

8. Click **OK**.

The **Attributes** tab in the **Create Local Profile** pane ([Figure 147 on page 379](#)) displays the updated attributes list.

Configuring a Non-3GPP Interworking Policy for SWa or STa Reference Point

Non-3GPP interworking policies are lists of rules governing authorization. SBR Carrier supports the following non-3GPP interworking policies for the SWa or STa reference point:

- Global Non-3GPP Access
- Global WLAN Direct IP Access

To configure a non-3GPP interworking policy for the SWa or STa reference point using the Web GUI:

1. In the **Create Local Profile** pane ([Figure 147 on page 379](#)), select **SWa/STa Interface > Non-3GPP Interworking Policy**.
2. Click **Add**.

The **Add Authorization** dialog box for non-3GPP interworking policy of SWa or STa reference point ([Figure 153 on page 387](#)) appears.

Figure 153: Add Authorization Dialog for Non-3GPP Interworking Policy-SWa/STa Interface

In the **Add Authorization** dialog box for non-3GPP interworking policy of SWa or STa reference point, you can:

- Select the **Global Non-3GPP Access** option to configure global non-3GPP network access policy. For more information about configuring global non-3GPP network access policy, see [“Configuring Global Non-3GPP Network Access Policy” on page 387](#).
- Select the **Global WLAN Direct IP Access** option to configure global WLAN direct IP access policy. For more information about configuring global WLAN direct IP access policy, see [“Configuring Global WLAN Direct IP Access Policy” on page 388](#).

Configuring Global Non-3GPP Network Access Policy

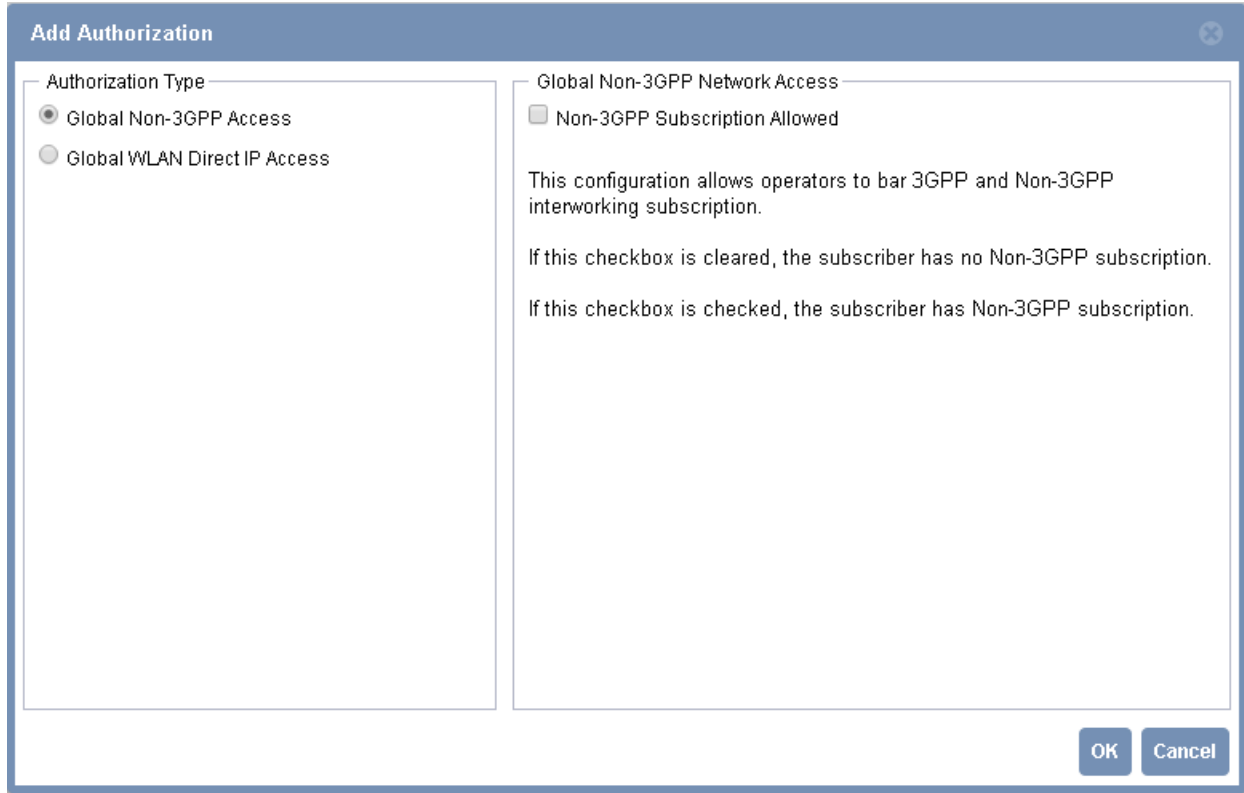
This policy allows the mobile operators to bar non-3GPP interworking subscription.

To configure the global non-3GPP network access policy using the Web GUI:

1. In the **Add Authorization** dialog box for non-3GPP interworking policy of the respective reference point ([Figure 153 on page 387](#), [Figure 156 on page 390](#), or [Figure 164 on page 399](#)), select the **Global Non-3GPP Network Access** option.

The **Global Non-3GPP Network Access** pane ([Figure 154 on page 388](#)) appears.

Figure 154: Add Authorization Dialog—Global Non-3GPP Network Access



The dialog box is titled "Add Authorization" and contains two main sections. The left section, titled "Authorization Type", has two radio buttons: "Global Non-3GPP Access" (which is selected) and "Global WLAN Direct IP Access". The right section, titled "Global Non-3GPP Network Access", contains a checkbox labeled "Non-3GPP Subscription Allowed" which is currently unchecked. Below the checkbox, there is explanatory text: "This configuration allows operators to bar 3GPP and Non-3GPP interworking subscription. If this checkbox is cleared, the subscriber has no Non-3GPP subscription. If this checkbox is checked, the subscriber has Non-3GPP subscription." At the bottom right of the dialog are "OK" and "Cancel" buttons.

2. Optionally, select the **Non-3GPP Subscription Allowed** check box to provide non-3GPP subscription for the subscriber.

Clear the **Non-3GPP Subscription Allowed** check box to block non-3GPP subscription for the subscriber.

3. Click **OK** to save the configuration.

The **Non-3GPP Interworking Policy** tab of the respective reference point in the **Create Local Profile** pane (Figure 147 on page 379) displays the updated interworking policy list.

Configuring Global WLAN Direct IP Access Policy

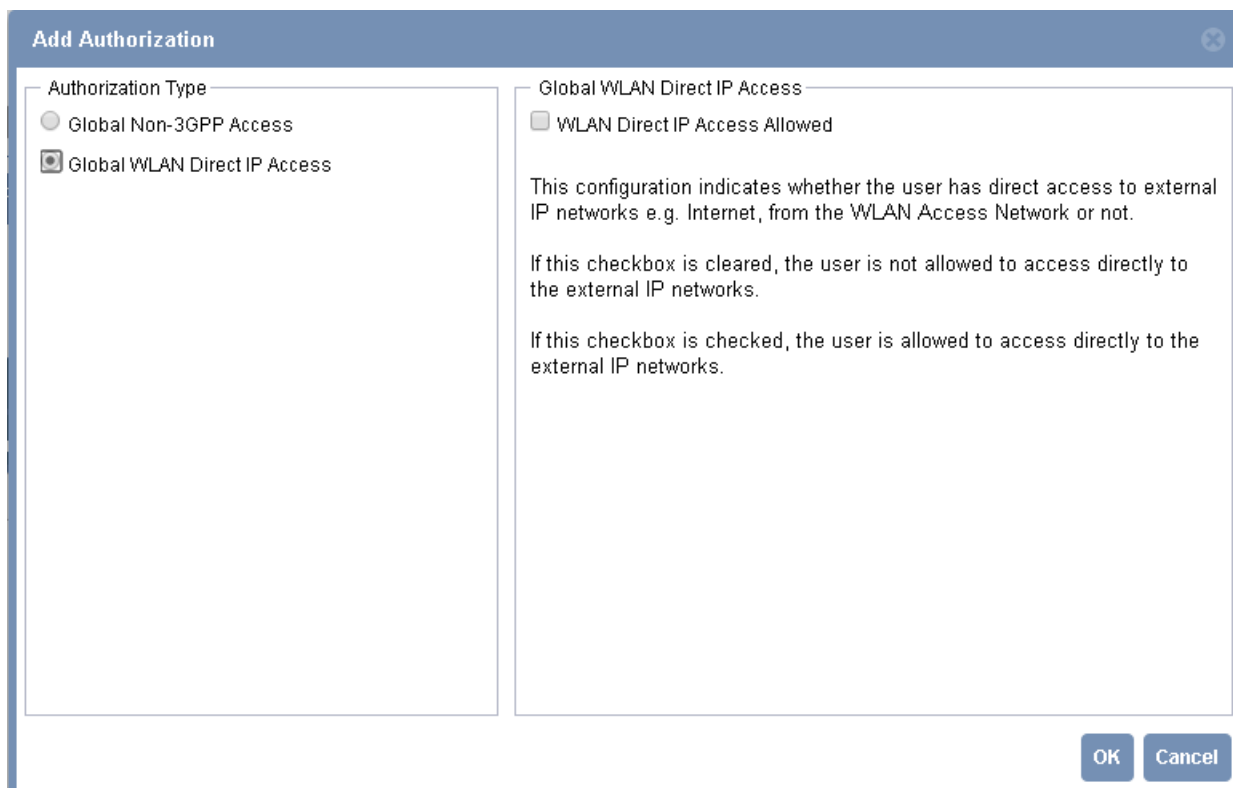
This policy allows the mobile operator to enable or disable the direct access to external networks (for example, Internet) from the WLAN access network for a subscriber.

To configure global WLAN direct IP access policy using the Web GUI:

1. In the **Add Authorization** dialog box for non-3GPP interworking policy of SWa or STa reference point (Figure 153 on page 387), select the **Global WLAN Direct IP Access** option.

The **Global WLAN Direct IP Access** pane (Figure 155 on page 389) appears.

Figure 155: Add Authorization Dialog—Global WLAN Direct IP Access



The dialog box is titled "Add Authorization" and contains two main sections. On the left, under "Authorization Type", there are two radio buttons: "Global Non-3GPP Access" (unselected) and "Global WLAN Direct IP Access" (selected). On the right, under "Global WLAN Direct IP Access", there is a checkbox labeled "WLAN Direct IP Access Allowed" which is currently unchecked. Below this checkbox, there is explanatory text: "This configuration indicates whether the user has direct access to external IP networks e.g. Internet, from the WLAN Access Network or not." followed by two paragraphs: "If this checkbox is cleared, the user is not allowed to access directly to the external IP networks." and "If this checkbox is checked, the user is allowed to access directly to the external IP networks." At the bottom right of the dialog are "OK" and "Cancel" buttons.

2. Optionally, select the **WLAN Direct IP Access Allowed** check box to allow the subscriber to access external networks.

Clear the **WLAN Direct IP Access Allowed** check box to block access to external networks for the subscriber.

3. Click **OK** to save the configuration.

The **Non-3GPP Interworking Policy** tab of the SWa/STa reference point in the **Create Local Profile** pane ([Figure 147 on page 379](#)) displays the updated interworking policy list.

Configuring a Non-3GPP Interworking Policy for SWm Reference Point

Non-3GPP interworking policies are lists of rules governing authorization. SBR Carrier supports the following non-3GPP interworking policies for the SWm reference point:

- Non-3GPP Interworking Allowed Visited Networks
- Global Non-3GPP Network Access
- Global Non-3GPP Network IP Access

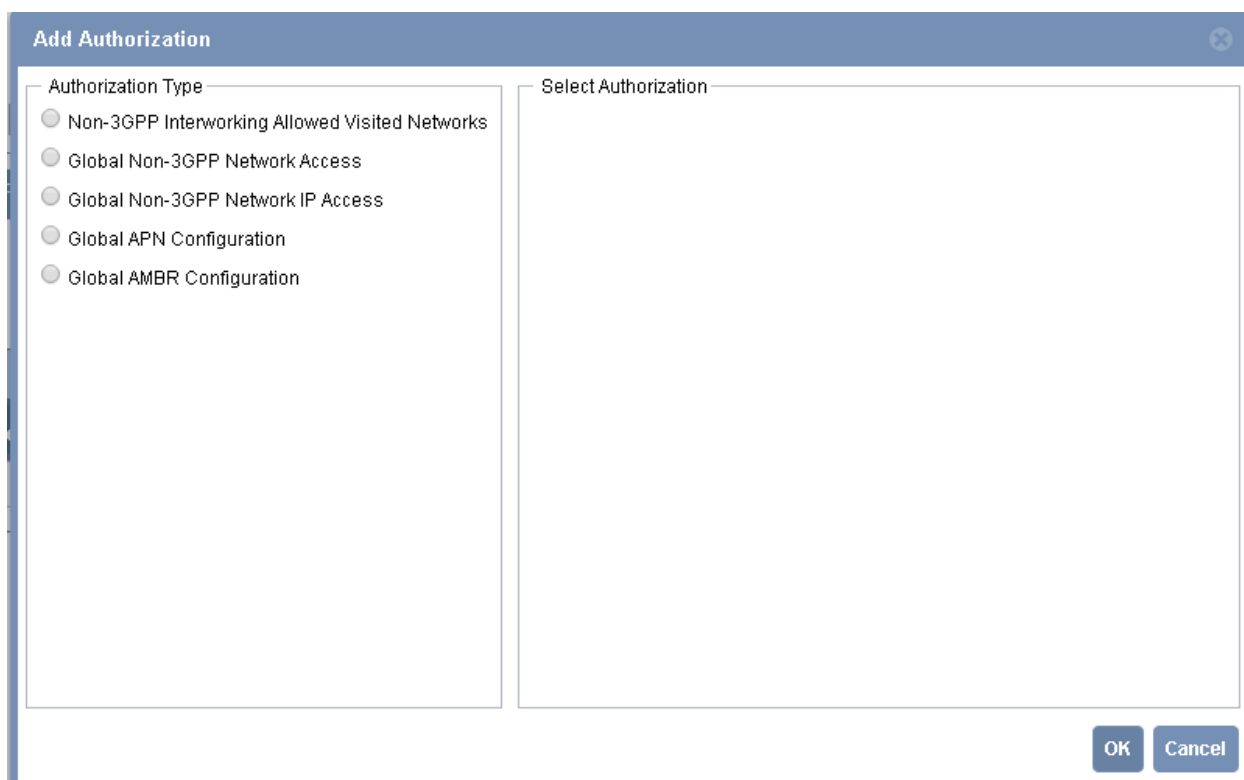
- Global APN Configuration
- Global AMBR Configuration

To configure a non-3GPP interworking policy for the SWm reference point using the Web GUI:

1. In the **Create Local Profile** pane ([Figure 147 on page 379](#)), select **SWm Interface > Non-3GPP Interworking Policy**.
2. Click **Add**.

The **Add Authorization** dialog box for non-3GPP interworking policy of SWm reference point ([Figure 156 on page 390](#)) appears.

Figure 156: Add Authorization Dialog for Non-3GPP Interworking Policy-SWm Interface



The image shows a web-based dialog box titled "Add Authorization". It has a blue header bar with the title and a close button. The dialog is divided into two main sections. The left section, titled "Authorization Type", contains a list of five radio button options: "Non-3GPP Interworking Allowed Visited Networks", "Global Non-3GPP Network Access", "Global Non-3GPP Network IP Access", "Global APN Configuration", and "Global AMBR Configuration". The right section, titled "Select Authorization", is a large empty rectangular area. At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

In the **Add Authorization** dialog box for non-3GPP interworking policy of SWm reference point, you can:

- Select the **Non-3GPP Interworking Allowed Visited Networks** option to configure non-3GPP interworking allowed visited networks policy. For more information about configuring non-3GPP interworking allowed visited networks policy, see [“Configuring Non-3GPP Interworking Allowed Visited Networks Policy” on page 391](#).
- Select the **Global Non-3GPP Network Access** option to configure global non-3GPP network access policy. For more information about configuring global non-3GPP network access policy, see [“Configuring Global Non-3GPP Network Access Policy” on page 387](#).
- Select the **Global Non-3GPP Network IP Access** option to configure global non-3GPP network IP access policy. For more information about configuring global non-3GPP network IP access policy, see [“Configuring Global Non-3GPP Network IP Access Policy” on page 393](#).
- Select the **Global APN Configuration** option to configure global access point name (APN) configuration policy. For more information about configuring global APN configuration policy, see [“Configuring Global APN Configuration Policy-SWm Interface” on page 394](#).
- Select the **Global AMBR Configuration** option to configure global aggregate maximum bit rate configuration policy. For more information about configuring global aggregate maximum bit rate configuration policy, see [“Configuring Global AMBR Configuration Policy” on page 397](#).

Configuring Non-3GPP Interworking Allowed Visited Networks Policy

The non-3GPP interworking allowed visited networks policy defines the VPLMNs that the subscriber can access while roaming.

To configure the non-3GPP interworking allowed visited networks policy using the Web GUI:

1. In the **Add Authorization** dialog box for non-3GPP interworking policy of the respective reference point ([Figure 156 on page 390](#) or [Figure 164 on page 399](#)), select the **Non-3GPP Interworking Allowed Visited Networks** option.

The **Allowed Visited Networks** pane ([Figure 157 on page 392](#)) appears.

Figure 157: Add Authorization Dialog—Allowed Visited Networks

The dialog box is titled "Add Authorization". It is divided into two main sections. The left section, titled "Authorization Type", contains five radio button options: "Non-3GPP Interworking Allowed Visited Networks" (which is selected), "Global Non-3GPP Network Access", "Global Non-3GPP Network IP Access", "Global APN Configuration", and "Global AMBR Configuration". The right section, titled "Allowed Visited Networks", contains a table with a single header row labeled "Name". Below the table is a checkbox labeled "Allow All". To the right of the checkbox are two buttons: "Add" and "Delete". At the bottom right of the dialog are two buttons: "OK" and "Cancel".

2. Add the visited networks.

- Select the **Allow All** check box to allow the subscriber to access any visited network.

Or

- Click **Add**.

The **Add VPLMN** dialog box ([Figure 158 on page 392](#)) appears.

Figure 158: Add VPLMN Dialog

The dialog box is titled "Add VPLMN". It contains a single text input field labeled "Name:". Below the input field are two buttons: "OK" and "Cancel".

- Enter the identifier of the visited network which a subscriber can access in the **Name** field and click **OK**. The **Allowed Visited Networks** pane ([Figure 157 on page 392](#)) displays the updated list of allowed visited networks.

Repeat these steps to add multiple allowed visited network entries. You can remove the allowed visited networks entry by using the **Delete** button.

3. Click **OK** to save the configuration.

The **Non-3GPP Interworking Policy** tab of the respective reference point in the **Create Local Profile** pane (Figure 147 on page 379) displays the updated interworking policy list.

Configuring Global Non-3GPP Network IP Access Policy

This policy allows the mobile operator to disable all non-3GPP access point names (APNs) for the subscriber at one time.

To configure global non-3GPP network IP access policy using the Web GUI:

1. In the **Add Authorization** dialog box for non-3GPP interworking policy of the respective reference point (Figure 156 on page 390 or Figure 164 on page 399), select the **Global Non-3GPP Network IP Access** option.

The **Global Non-3GPP Network IP Access** pane (Figure 159 on page 393) appears.

Figure 159: Add Authorization Dialog—Global Non-3GPP Network IP Access

The screenshot shows a dialog box titled "Add Authorization". On the left, under "Authorization Type", there are five radio button options: "Non-3GPP Interworking Allowed Visited Networks", "Global Non-3GPP Network Access", "Global Non-3GPP Network IP Access" (which is selected), "Global APN Configuration", and "Global AMBR Configuration". On the right, under "Global Non-3GPP Network IP Access", there is a checkbox labeled "Non-3GPP APN's Enabled" which is currently unchecked. Below the checkbox, there is explanatory text: "This configuration allows operator to disable all APN's for a subscriber at one time." followed by "If this checkbox is cleared, all APN's for this subscriber are disabled." and "If this checkbox is checked, all APN's for this subscriber are enabled." At the bottom right of the dialog are "OK" and "Cancel" buttons.

2. Optionally, select the **Non-3GPP APNs Enabled** check box to allow the subscriber to access all non-3GPP APNs.

Clear the **Non-3GPP APNs Enabled** check box to block access to all APNs for the subscriber.

3. Click **OK** to save the configuration.

The **Non-3GPP Interworking Policy** tab of the respective reference point in the **Create Local Profile** pane ([Figure 147 on page 379](#)) displays the updated interworking policy list.

Configuring Global APN Configuration Policy-SWm Interface

The global APN configuration policy defines the values that are used in the APN-Configuration AVP, which is sent in every successful Diameter-EAP-Answer (DEA) message. The values defined by the global APN configuration policy is used in the APN-Configuration AVP only if there is a match between the Service-Selection AVP in Diameter-EAP-Request (DER) message and the global APN configuration policy.

NOTE: The Service-Selection AVP in DER message and the global APN configuration policy are matched only if there is no match between the Service-Selection AVP in DER message and the Service-Selection AVP in Server-Assignment-Answer (SAA) message. If an APN-Configuration name match already exists between DER and SAA messages, the values in the SAA message are used in the APN-Configuration AVP.

To configure the global APN configuration policy using the Web GUI:

1. In the **Add Authorization** dialog box for non-3GPP interworking policy of SWm reference point (Figure 156 on page 390), select the **Global APN Configuration** option.

The **Global APN Configuration** pane (Figure 160 on page 395) appears with the **Basic Configuration** tab selected.

Figure 160: Add Authorization Dialog—Global APN Configuration (SWm interface)

The screenshot shows the 'Add Authorization' dialog box. On the left, under 'Authorization Type', the 'Global APN Configuration' option is selected. The main area is titled 'Global APN Configuration' and contains three tabs: 'Basic Configuration' (selected), 'AMBR', and 'EPS-Subscribed-QoS-Profile'. The 'Basic Configuration' tab has the following fields: 'Name' (text input), 'Context Identifier' (dropdown menu), 'PDN Type' (dropdown menu), a checkbox for 'VPLMN-Dynamic-Address-Allowed', and '3GPP-Charging-Characteristics' (text input). At the bottom right are 'OK' and 'Cancel' buttons.

2. Enter the name of the non-3GPP access point in the **Name** field.

NOTE: This name is used when performing a match between the Service-Selection AVP in DER message and the global APN configuration policy.

3. Enter the Context-Identifier AVP in the **Context Identifier** field. The specified Context-Identifier AVP is used as a grouped AVP in the APN-Configuration AVP.
4. Use the **PDN Type** list to specify the address type of PDN to be accessed.
 - Select the **IPv4** option to use IPv4 addressing.
 - Select the **IPv6** option to use IPv6 addressing.
 - Select the **IPv4v6** option to use both IPv4 and IPv6 addressing.
 - Select the **IPv4_OR_IPv6** option to use either IPv4 addressing or IPv6 addressing.

5. Use the **VPLMN-Dynamic-Address-Allowed** check box to specify whether the user in a VPLMN is allowed to activate a non-3GPP access point that accesses a PDG in the VPLMN.
6. Enter the charging characteristics value to be applied to all user sessions in the **3GPP-Charging-Characteristics** field.
7. Click the **AMBR** tab (Figure 161 on page 396) to specify the values that are used in the APN-AMBR configuration AVP.

Figure 161: Global APN Configuration Pane (SWm Interface)—AMBR

The screenshot shows a software interface titled "Add Authorization". On the left, under "Authorization Type", there are five radio button options: "Non-3GPP Interworking Allowed Visited Networks", "Global Non-3GPP Network Access", "Global Non-3GPP Network IP Access", "Global APN Configuration" (which is selected), and "Global AMBR Configuration". On the right, the "Global APN Configuration" pane is active, showing three tabs: "Basic Configuration", "AMBR" (which is selected), and "EPS-Subscribed-QoS-Profile". The "AMBR" tab contains two input fields: "Max-Requested-Bandwidth-UL:" and "Max-Requested-Bandwidth-DL:", each with a numeric spinner control. At the bottom right of the pane are "OK" and "Cancel" buttons.

8. Enter the maximum allowed uplink bandwidth in the **Max-Requested-Bandwidth-UL** field.
9. Enter the maximum allowed downlink bandwidth in the **Max-Requested-Bandwidth-DL** field.
10. Click the **EPS-Subscribed-QoS-Profile** tab (Figure 162 on page 397) to specify the QoS profile details.

Figure 162: Global APN Configuration Pane (SWm Interface)—EPS-Subscribed-QoS-Profile

The screenshot shows a web interface titled "Add Authorization". On the left, under "Authorization Type", there are five radio buttons: "Non-3GPP Interworking Allowed Visited Networks", "Global Non-3GPP Network Access", "Global Non-3GPP Network IP Access", "Global APN Configuration" (which is selected), and "Global AMBR Configuration". On the right, the "Global APN Configuration" pane is open, showing three tabs: "Basic Configuration", "AMBR", and "EPS-Subscribed-QoS-Profile" (which is active). The "EPS-Subscribed-QoS-Profile" tab contains the following fields and options:

- QoS-Class-Identifier:** A text input field.
- Allocation-Retention-Priority:** A section containing:
 - Priority Level:** A text input field.
 - ☐ Pre-emption-Capability-Enabled
 - ☒ Pre-emption-Vulnerability-Enabled

At the bottom right of the pane are "OK" and "Cancel" buttons.

11. Enter a value for the QoS class in the **QoS-Class-Identifier** field to identify QoS parameters that define the authorized QoS. You can enter the values 0, 10–64, 67, 68, and 71–255.
12. Enter the priority level of the QoS parameters in the **Priority Level** field. You can enter a value ranging from 1 through 15.
13. Use the **Pre-emption-Capability-Enabled** check box to enable or disable the pre-emption capability.
14. Use the **Pre-emption-Vulnerability-Enabled** check box to enable or disable the pre-emption vulnerability.
15. Click **OK** to save the configuration.

The **Non-3GPP Interworking Policy** tab of the SWm reference point in the **Create Local Profile** pane (Figure 147 on page 379) displays the updated interworking policy list.

Configuring Global AMBR Configuration Policy

The global aggregate maximum bit rate configuration policy defines the values that are used in the APN-AMBR configuration AVP, which is sent from the SBR Carrier server to the mobile home agent in the APN-Configuration AVP. The APN-AMBR configuration AVP is used to limit the aggregate bit rate that is expected to be provided across all non-guaranteed bit rate bearers and PGW connections of the same APN. The APN-AMBR configuration AVP is stored per APN in the HSS.

To configure the global aggregate maximum bit rate configuration policy using the Web GUI:

1. In the **Add Authorization** dialog box for non-3GPP interworking policy of the respective reference point (Figure 156 on page 390 or Figure 164 on page 399), select the **Global AMBR Configuration** option.
The **Global AMBR Configuration** pane (Figure 163 on page 398) appears.

Figure 163: Add Authorization Dialog—Global AMBR Configuration

The screenshot shows a dialog box titled "Add Authorization". On the left, under "Authorization Type", there are five radio button options: "Non-3GPP Interworking Allowed Visited Networks", "Global Non-3GPP Network Access", "Global Non-3GPP Network IP Access", "Global APN Configuration", and "Global AMBR Configuration". The "Global AMBR Configuration" option is selected. On the right, under "Global AMBR Configuration", there are two input fields: "Max-requested-Bandwidth-UL:" and "Max-requested-Bandwidth-DL:". Both fields are currently empty. At the bottom right of the dialog box are "OK" and "Cancel" buttons.

2. Enter the maximum allowed uplink bandwidth in the **Max-Requested-Bandwidth-UL** field.
3. Enter the maximum allowed downlink bandwidth in the **Max-Requested-Bandwidth-DL** field.
4. Click **OK** to save the configuration.

The **Non-3GPP Interworking Policy** tab of the respective reference point in the **Create Local Profile** pane (Figure 147 on page 379) displays the updated interworking policy list.

Configuring a Non-3GPP Interworking Policy for S6b Reference Point

Non-3GPP interworking policies are lists of rules governing authorization. SBR Carrier supports the following non-3GPP interworking policies for the S6b reference point:

- Non-3GPP Interworking Allowed Visited Networks
- Global Non-3GPP Network Access
- Global Non-3GPP Network IP Access
- Global APN Configuration

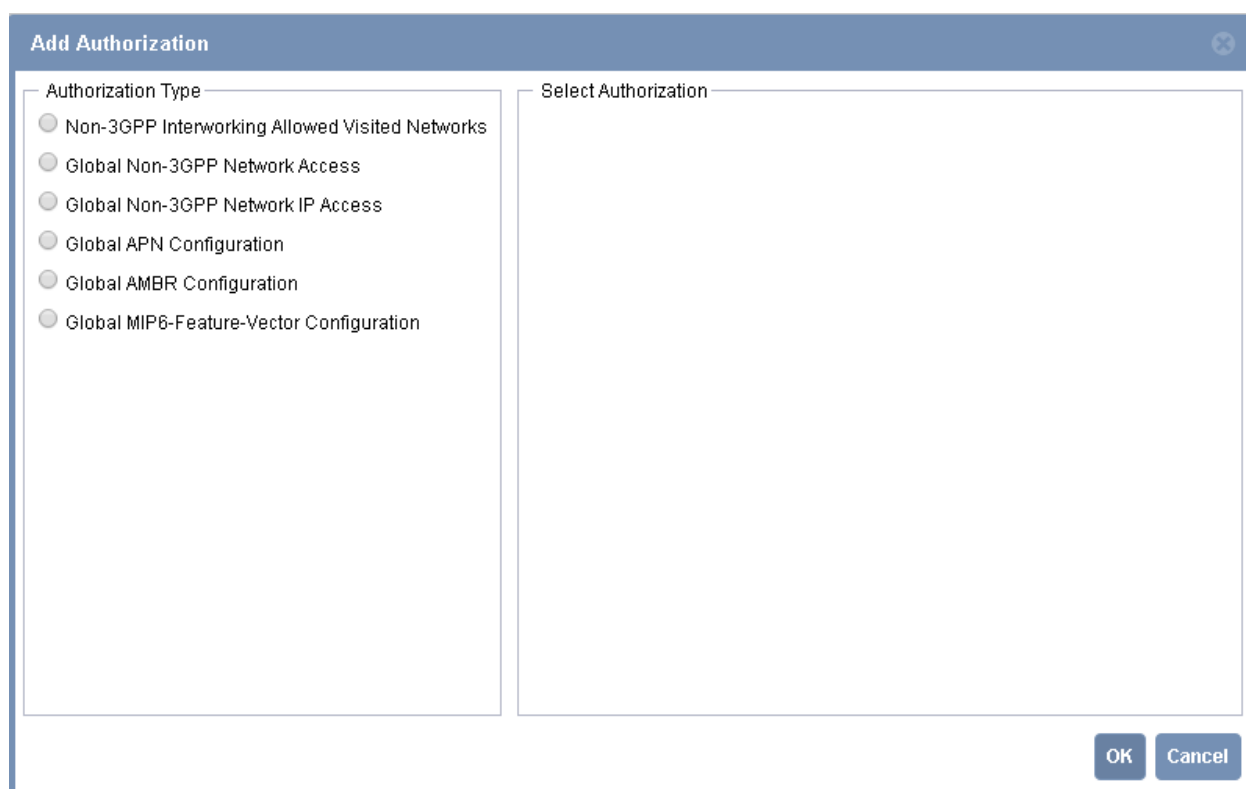
- Global AMBR Configuration
- Global MIP6-Feature-Vector Configuration

To configure a non-3GPP interworking policy for the S6b reference point using the Web GUI:

1. In the **Create Local Profile** pane ([Figure 147 on page 379](#)), select **S6b Interface > Non-3GPP Interworking Policy**.
2. Click **Add**.

The **Add Authorization** dialog box for non-3GPP interworking policy of S6b reference point ([Figure 164 on page 399](#)) appears.

Figure 164: Add Authorization Dialog for Non-3GPP Interworking Policy-S6b Interface



In the **Add Authorization** dialog box for non-3GPP interworking policy of S6b reference point, you can:

- Select the **Non-3GPP Interworking Allowed Visited Networks** option to configure non-3GPP interworking allowed visited networks policy. For more information about configuring non-3GPP interworking allowed visited networks policy, see [“Configuring Non-3GPP Interworking Allowed Visited Networks Policy” on page 391](#).
- Select the **Global Non-3GPP Network Access** option to configure global non-3GPP network access policy. For more information about configuring global non-3GPP network access policy, see [“Configuring Global Non-3GPP Network Access Policy” on page 387](#).

- Select the **Global Non-3GPP Network IP Access** option to configure global non-3GPP network IP access policy. For more information about configuring global non-3GPP network IP access policy, see [“Configuring Global Non-3GPP Network IP Access Policy” on page 393](#).
- Select the **Global APN Configuration** option to configure global APN configuration policy. For more information about configuring global APN configuration policy, see [“Configuring Global APN Configuration Policy-S6b Interface” on page 400](#).
- Select the **Global AMBR Configuration** option to configure global aggregate maximum bit rate configuration policy. For more information about configuring global aggregate maximum bit rate configuration policy, see [“Configuring Global AMBR Configuration Policy” on page 397](#).
- Select the **Global MIP6-Feature-Vector Configuration** option to configure global Mobile IP 6 (MIP6) feature vector configuration policy. For more information about configuring global MIP6 feature vector configuration policy, see [“Configuring Global MIP6 Feature Vector Configuration Policy” on page 404](#).

Configuring Global APN Configuration Policy-S6b Interface

The global APN configuration policy defines the values that are used in the APN-Configuration AVP, which is sent in every successful Diameter-EAP-Answer (DEA) message. The values defined by the global APN configuration policy are used in the APN-Configuration AVP only if there is a match between the Service-Selection AVP in the Diameter-EAP-Request (DER) message and the global APN configuration policy.

NOTE: The Service-Selection AVP in the DER message and the global APN configuration policy are matched only if there is no match between the Service-Selection AVP in the DER message and the Service-Selection AVP in the Server-Assignment-Answer (SAA) message. If an APN-Configuration name match already exists between the DER and SAA messages, the values in the SAA message are used in the APN-Configuration AVP.

To configure the global APN configuration policy using the Web GUI:

1. In the **Add Authorization** dialog box for non-3GPP interworking policy of S6b reference point (Figure 164 on page 399), select the **Global APN Configuration** option.

The **Global APN Configuration** pane (Figure 165 on page 401) appears with the **Basic Configuration** tab selected.

Figure 165: Add Authorization Dialog—Global APN Configuration (S6b interface)

2. Enter the name of the non-3GPP access point in the **Name** field.

NOTE: This name is used when performing a match between the Service-Selection AVP in DER message and the global APN configuration policy.

3. Enter the Context-Identifier AVP in the **Context Identifier** field. The specified Context-Identifier AVP is used as a grouped AVP in the APN-Configuration AVP.
4. Use the **PDN Type** list to specify the address type of PDN to be accessed.
 - Select the **IPv4** option to use IPv4 addressing.
 - Select the **IPv6** option to use IPv6 addressing.
 - Select the **IPv4v6** option to use both IPv4 and IPv6 addressing.
 - Select the **IPv4_OR_IPv6** option to use either IPv4 addressing or IPv6 addressing.
5. Use the **VPLMN-Dynamic-Address-Allowed** check box to specify whether the user in a VPLMN is allowed to activate a non-3GPP access point that accesses a PDG in the VPLMN.

6. Enter the charging characteristics value to be applied to all user sessions in the **3GPP-Charging-Characteristics** field.
7. Click the **AMBR** tab (Figure 166 on page 402) to specify the values that are used in the APN-AMBR configuration AVP.

Figure 166: Global APN Configuration Pane (S6b Interface)—AMBR

The screenshot shows a software interface titled "Add Authorization". On the left, under "Authorization Type", there are several radio button options: "Non-3GPP Interworking Allowed Visited Networks", "Global Non-3GPP Network Access", "Global Non-3GPP Network IP Access", "Global APN Configuration" (which is selected), "Global AMBR Configuration", and "Global MIP6-Feature-Vector Configuration". On the right, the "Global APN Configuration" pane is active, showing four tabs: "Basic Configuration", "AMBR" (selected), "MIP6-Agent-Info", and "EPS-Subscribed-QoS-Profile". The "AMBR" tab contains two input fields: "Max-Requested-Bandwidth-UL:" and "Max-Requested-Bandwidth-DL:", each with a small up/down arrow icon to its right. At the bottom right of the pane are "OK" and "Cancel" buttons.

8. Enter the maximum allowed uplink bandwidth in the **Max-Requested-Bandwidth-UL** field.
9. Enter the maximum allowed downlink bandwidth in the **Max-Requested-Bandwidth-DL** field.
10. Click the **MIP6-Agent-Info** tab (Figure 167 on page 403) to specify the MIP6 agent details.

Figure 167: Global APN Configuration Pane (S6b Interface)—MIP6-Agent-Info

11. Enter the IP address of the mobile home agent in the **MIP-Home-Agent-Address** field.

12. Optionally, select the **Use IPv6** check box to use IPv6 addressing.

NOTE: SBR Carrier does not support an embedded IPv4 address as an IPv6 address.

13. Enter the host name of the mobile home agent in the **Destination Host** field.

14. Enter the realm name in which the mobile home agent is located in the **Destination Realm** field.

The values provided in the **Destination Host** and **Destination Realm** fields are used to form the fully qualified domain name (FQDN).

15. Click the **EPS-Subscribed-QoS-Profile** tab (Figure 168 on page 404) to specify QoS profile details.

Figure 168: Global APN Configuration Pane (S6b Interface)—EPS-Subscribed-QoS-Profile

The screenshot shows a web interface titled "Add Authorization". On the left, under "Authorization Type", several radio buttons are listed, with "Global APN Configuration" selected. On the right, the "Global APN Configuration" pane is open, showing four tabs: "Basic Configuration", "AMBR", "MIP6-Agent-Info", and "EPS-Subscribed-QoS-Profile". The "EPS-Subscribed-QoS-Profile" tab is active, displaying the following fields:

- QoS-Class-Identifier:** A text input field.
- Allocation-Retention-Priority:** A section containing:
 - Priority Level:** A dropdown menu.
 - Pre-emption-Capability-Enabled:** An unchecked checkbox.
 - Pre-emption-Vulnerability-Enabled:** A checked checkbox.

At the bottom right of the pane are "OK" and "Cancel" buttons.

16. Enter a value for the QoS class in the **QoS-Class-Identifier** field to identify QoS parameters that define the authorized QoS. You can enter the values 0, 10–64, 67, 68, and 71–255.
17. Enter the priority level of the QoS parameters in the **Priority Level** field. You can enter a value ranging from 1 through 15.
18. Use the **Pre-emption-Capability-Enabled** check box to enable or disable the pre-emption capability.
19. Use the **Pre-emption-Vulnerability-Enabled** check box to enable or disable the pre-emption vulnerability.
20. Click **OK** to save the configuration.

The **Non-3GPP Interworking Policy** tab of the S6b reference point in the **Create Local Profile** pane (Figure 147 on page 379) displays the updated interworking policy list.

Configuring Global MIP6 Feature Vector Configuration Policy

The global MIP6 feature vector configuration policy defines NAD indicated capabilities that are supported or authorized by the 3GPP AAA server (that is, Diameter server). The server supported or authorized capabilities are sent to the NAD in the MIP6-Feature-Vector AVP.

To configure the global MIP6 feature vector configuration policy using the Web GUI:

1. In the **Add Authorization** dialog box for non-3GPP interworking policy of S6b reference point (Figure 164 on page 399), select the **Global MIP6-Feature-Vector Configuration** option.

The **Global MIP6-Feature-Vector Configuration** pane (Figure 169 on page 405) appears.

Figure 169: Global MIP6-Feature-Vector Configuration Pane

The screenshot shows a web-based configuration interface. On the left, under the heading 'Add Authorization', there is a list of 'Authorization Type' options with radio buttons. The selected option is 'Global MIP6-Feature-Vector Configuration'. To the right, a pane titled 'Global MIP6-Feature-Vector Configuration' is displayed. It contains a table with two columns: 'Value' and a checkbox. The table has two rows: one for 'MIP6_INTEGRATED' and one for 'LOCAL_HOME_AGENT_ASSIGNMENT', both with their checkboxes selected. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

Value	
MIP6_INTEGRATED	<input checked="" type="checkbox"/>
LOCAL_HOME_AGENT_ASSIGNMENT	<input checked="" type="checkbox"/>

2. Select the **MIP6_INTEGRATED** option to enable the support for MIP6 integrated scenario bootstrapping.
3. Select the **LOCAL_HOME_AGENT_ASSIGNMENT** option to enable the authorization for assigning local home agent to the mobile network.
4. Click **OK** to save the configuration.

The **Non-3GPP Interworking Policy** tab of the S6b reference point in the **Create Local Profile** pane (Figure 147 on page 379) displays the updated interworking policy list.

Editing a Local Profile

To edit an existing local profile in SBR Carrier using the Web GUI:

1. Select **Diameter Configuration > Policy > Local Profiles**.

The **Local Profiles List** page (Figure 146 on page 378) appears.

2. Select the local profile entry that you want to edit.

The **Selected Profile** pane (Figure 170 on page 406) displays the settings configured for the local policy.

Figure 170: Selected Profile Pane

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the current user (root). The main content area is divided into two panes. The top pane, titled 'Local Profiles List', contains a table with two rows: 'Profile 1' and 'Profile 2', both with the description 'Example configuration'. Below the table are pagination controls showing 'Page 1 of 1' and 'Displaying 1 - 2 of 2'. The bottom pane, titled 'Selected Profile: Profile 1', contains a form for editing the profile. It has fields for 'Name' (Profile 1) and 'Description' (Example configuration). Below these are tabs for 'SWa/STa Interface', 'SWm Interface', and 'S6b Interface'. The 'Attributes' tab is selected, showing a table with columns 'Name' and 'Description'. The table is currently empty, with a message 'No Ordered SWa/STa Attributes to Display'. At the bottom of the pane are buttons for 'Save', 'Reset', and 'Cancel'.

3. Edit the existing description in the **Description** field.

NOTE: You cannot edit the name of the local profile.

4. Select either the **Attributes** tab or the **Non-3GPP Interworking Policy** tab.

From the **Attributes** tab, you can:

- Click **Add** to add a new authorization attribute for the selected reference point. The **Add Authorization** dialog box for authorization attributes (Figure 148 on page 381) appears. For more information about adding authorization attributes, see “Configuring Authorization Attributes” on page 380.
- Select an authorization attribute entry and click **Edit** to edit the existing authorization attribute. For more information about the fields in the **Add Authorization** dialog box for authorization attributes, see “Configuring Authorization Attributes” on page 380
- Select an authorization attribute entry and click **Delete** to delete the attribute.
- Reorder the attributes by selecting each attribute and using the **Up** or **Down** arrow.

The **Attributes** tab displays an updated list of authorization attribute entries.

From the **Non-3GPP Interworking Policy** tab, you can:

- Click **Add** to configure a new non-3GPP interworking policy for the selected reference point. The **Add Authorization** dialog box for non-3GPP interworking policy (Figure 153 on page 387, Figure 156 on page 390, or Figure 164 on page 399) appears. For more information about configuring non-3GPP interworking policies, see “Configuring a Non-3GPP Interworking Policy for SWa or STa Reference Point” on page 386, “Configuring a Non-3GPP Interworking Policy for SWm Reference Point” on page 389, and “Configuring a Non-3GPP Interworking Policy for S6b Reference Point” on page 398.
- Select a non-3GPP interworking policy entry and click **Edit** to edit the existing non-3GPP interworking policy. For more information about the fields in the **Add Authorization** dialog box for non-3GPP interworking policy, see “Configuring a Non-3GPP Interworking Policy for SWa or STa Reference Point” on page 386, “Configuring a Non-3GPP Interworking Policy for SWm Reference Point” on page 389, and “Configuring a Non-3GPP Interworking Policy for S6b Reference Point” on page 398.
- Select a non-3GPP interworking policy entry and click **Delete** to delete the policy.
- Reorder the non-3GPP interworking policies by selecting each policy and using the **Up** or **Down** arrow.

The **Non-3GPP Interworking Policy** tab displays an updated list of non-3GPP interworking policy entries.

5. Click **Save** to save the changes.

The **Local Profiles List** page (Figure 146 on page 378) displays an updated list of local profile entries.

Deleting a Local Profile

To delete a local profile using the Web GUI:

1. Select **Diameter Configuration > Policy > Local Profiles**.

The **Local Profiles List** page ([Figure 146 on page 378](#)) appears.

2. Select the local profile entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to delete the local profile.

The **Local Profiles List** page ([Figure 146 on page 378](#)) displays an updated list of local profile entries.

Configuring Local Profile Selection

During the authorization process, SBR Carrier uses user-defined profile selection rule sets to select the local profile used to authorize the request. To select the correct profile for authorization, the server compares the HSS or downstream server response with the profile selection rule sets in the order in which they are defined. A match is declared when everything in a profile selection rule set is included in the HSS or downstream server response. If a match is found, the local profile associated with the profile selection rule set is merged with the HSS or downstream server response to form a complete profile which is used to authorize subscriber's requests.

The merged profile is cached in SBR Carrier and used to authorize all future requests from the subscriber. The cached profile in SBR Carrier and the HSS are synchronized through the Push Profile Request (PPR) process, which notifies the SBR Carrier server about the profile modification through a PPR message. If there is a conflict between the two databases, the HSS profile data takes precedence over the SBR Carrier profile data.

A specific profile can be configured for processing requests with no matching rules, or you can configure the server to reject all requests for which no match is found.

SBR Carrier can also provide backup support for the HSS by continuing to process requests when the HSS is unavailable. A specific profile can be defined for processing requests in this situation, or you have the option to reject all requests when the HSS is unavailable.

NOTE: SBR carrier selects the local profile based on the individual AVPs and not based on the AVPs within the grouped AVPs.

To configure the local profile selection using the Web GUI:

1. Select **Diameter Configuration > Policy > Local Profile Selection**.

The **Local Profile Selection** page ([Figure 171 on page 409](#)) appears.

Figure 171: Local Profile Selection Page

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration ▾ Diameter Configuration ▾ Tools ▾ Help ▾ Logout User: root

Local Profile Selection

Rulesets

Profile	Description
No Rulesets to Display	

Add Edit Delete

When No Ruleset Matches

☐ Reject All ☐ Use Selected Profile ▾

When HSS is Not Available

☐ Reject All ☐ Use Selected Profile ▾

Save Reset Refresh

2. Configure a local profiles selection rule set. For more information about configuring a profile selection rule set, see [“Creating a New Profile Selection Rule Set” on page 410](#), [“Editing Profile Selection Rule Sets” on page 415](#), and [“Deleting Profile Selection Rule Sets” on page 416](#).
3. Define the order of the rule sets by selecting each rule set and using the **Up** or **Down** arrow.
4. Specify whether to reject all requests or use a specific profile when there is no match found during the profile selection process in the **When No Ruleset Matches** area.
 - Select the **Reject All** option to reject all requests that do not have a matching rule.

- Select the **Use Selected Profile** option and select a profile that you want to use for requests that do not have a matching rule.
5. Specify whether to reject all requests or use a specific profile to continue to process the authorization requests when HSS is unavailable. In the **When HSS is Not Available** area, you can:
 - Select the **Reject All** option to reject all requests when HSS is unavailable.
 - Select the **Use Selected Profile** option and select a profile that you want to use for all requests when HSS is unavailable.
 6. Click **Save** to save the configuration.

Creating a New Profile Selection Rule Set

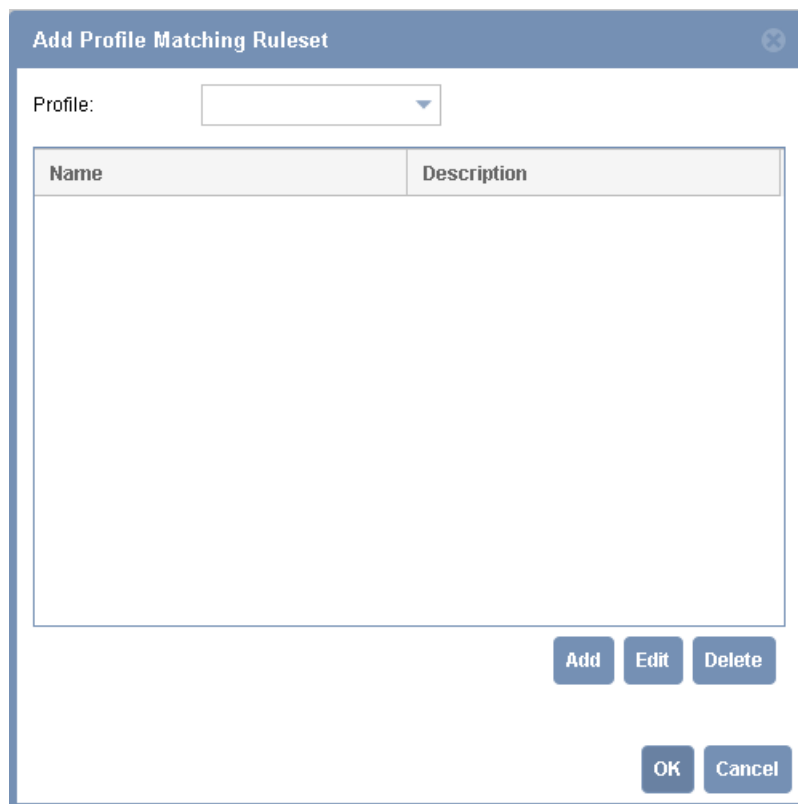
A profile selection rule set consists of ordered lists of user-defined attributes and/or non-3GPP interworking policies. To create a profile selection rule set, first you select the profile for which you want to create the selection rule set. You then add the user-defined attributes and non-3GPP interworking policies.

To create a profile selection rule set using the Web GUI:

1. In the **Local Profile Selection** page ([Figure 171 on page 409](#)), click **Add**.

The **Add Profile Matching Ruleset** dialog box ([Figure 172 on page 411](#)) appears.

Figure 172: Add Profile Matching Ruleset Dialog



The dialog box is titled "Add Profile Matching Ruleset". It features a "Profile:" label followed by a dropdown menu. Below this is a table with two columns: "Name" and "Description". The table is currently empty. At the bottom right of the table area are three buttons: "Add", "Edit", and "Delete". At the very bottom of the dialog are two buttons: "OK" and "Cancel".

Name	Description
------	-------------

2. Select the profile for which you want to create the profile selection rule set from the **Profile** list.
3. Create matching rules for the selected profile. For more information about creating matching rules, see ["Creating New Matching Rules" on page 412](#).

Each row represents a rule. A rule can contain one or more attributes. You can modify the matching rules by using the **Edit** and **Delete** buttons.

4. Click **OK** to save the configuration.

The **Local Profile Selection** page ([Figure 171 on page 409](#)) displays an updated list of local profiles selection rule sets.

Creating New Matching Rules

To create new matching rules using the Web GUI:

1. In the **Add Profile Matching Ruleset** dialog box (Figure 172 on page 411), click **Add**.

The **Add Profile Matching Rule** dialog box (Figure 173 on page 412) appears.

Figure 173: Add Profile Matching Rule Dialog

2. From the **Add Profile Matching Rule** dialog box, you can:

- Select the **User Defined Attributes** option to configure user defined attribute matching rules. For more information about configuring user defined attribute matching rules, see [“Configuring User Defined Attribute Matching Rules” on page 413](#).
- Select one of the following options:
 - **Non-3GPP Interworking Allowed Visited Networks**. When this option is selected, the **Allowed Visited Networks** pane similar to [Figure 157 on page 392](#) appears. For more information about the fields in this dialog box, see [“Configuring Non-3GPP Interworking Allowed Visited Networks Policy” on page 391](#).
 - **Global Non-3GPP Network Access**. When this option is selected, the **Global Non-3GPP Network Access** pane similar to [Figure 154 on page 388](#) appears. For more information about the fields in this dialog box, see [“Configuring Global Non-3GPP Network Access Policy” on page 387](#).

- **Global Non-3GPP Network IP Access.** When this option is selected, the **Global Non-3GPP Network IP Access** pane similar to [Figure 159 on page 393](#) appears. For more information about the fields in this dialog box, see [“Configuring Global Non-3GPP Network IP Access Policy” on page 393](#).

Configuring User Defined Attribute Matching Rules

To configure user defined attribute matching rules using the Web GUI:

1. In the **Add Profile Matching Rule** dialog box ([Figure 173 on page 412](#)), select the **User Defined Attributes** option.

The **Match these AVPs from response** pane ([Figure 174 on page 413](#)) appears.

Figure 174: Match These AVPs from Response Pane

Add Profile Matching Rule

Rule Type

- ☒ User Defined Attributes
- ☐ Non-3GPP Interworking Allowed Visited Networks
- ☐ Global Non-3GPP Network Access
- ☐ Global Non-3GPP Network IP Access

Match these AVPs from response

Name	Value
------	-------

Add Edit Delete

OK Cancel

2. Click **Add**.

The **Add AVP** dialog box ([Figure 175 on page 414](#)) appears.

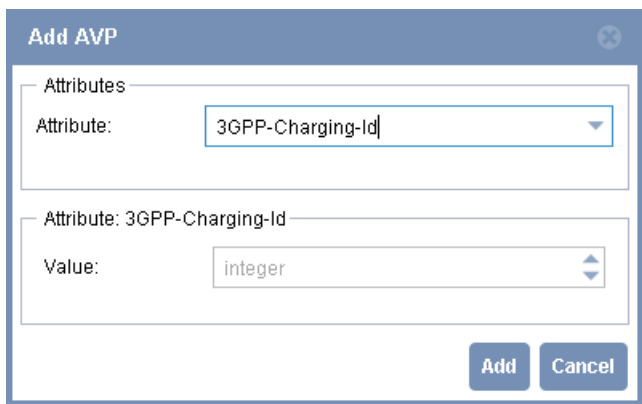
Figure 175: Add AVP Dialog



The 'Add AVP' dialog box has a title bar with a close button. It contains two sections: 'Attributes' and 'Select Attribute'. The 'Attributes' section has a label 'Attribute:' followed by a dropdown menu. The 'Select Attribute' section is a large empty text area. At the bottom right are 'Add' and 'Cancel' buttons.

3. Select the attribute that you want to add from the **Attributes** list. [Figure 176 on page 414](#) shows a sample **Add AVP** dialog box for the **3GPP-Charging-Id** attribute.

Figure 176: Sample Add AVP Dialog



The 'Add AVP' dialog box is shown with the '3GPP-Charging-Id' attribute selected in the 'Attribute:' dropdown. Below this, there is a section labeled 'Attribute: 3GPP-Charging-Id' with a 'Value:' dropdown menu showing 'integer'. The 'Add' and 'Cancel' buttons are at the bottom right.

4. Enter a value for the selected attribute in the **Value** field and click **Add**.
5. Repeat steps 3 and 4 to assign more attributes to the matching rule.
6. When you are finished adding attributes, click **Cancel**. The **Match these AVPs from response** pane ([Figure 174 on page 413](#)) displays the updated list of attributes. You can modify the matching rule attributes list by using the **Edit** and **Delete** buttons.
7. Click **OK**.

The **Add Profile Matching Ruleset** dialog box ([Figure 172 on page 411](#)) displays the updated matching rule list.

Editing Profile Selection Rule Sets

To edit the existing profile selection rule set using the Web GUI:

1. Select **Diameter Configuration > Policy > Local Profile Selection**.

The **Local Profile Selection** page ([Figure 171 on page 409](#)) appears.

2. Select the profile selection rule set entry that you want to edit.

3. Click **Edit**.

The **Edit Profile Matching Ruleset** dialog box ([Figure 177 on page 415](#)) appears.

Figure 177: Edit Profile Matching Ruleset Dialog

Profile: Default

Name	Description
------	-------------

Add Edit Delete

OK Cancel

4. Change the profile for which you want to add or edit the profile selection rule set from the **Profile** list.

5. In the **Rules** area, you can:

- Click **Add** to add a new matching rule for the selected profile. The **Add Profile Matching Rule** dialog box ([Figure 173 on page 412](#)) appears. For more information about creating new matching rules, see [“Creating New Matching Rules” on page 412](#).

- Select a matching rule entry and click **Edit** to edit the existing rule. For more information about the fields in the **Edit Profile Matching Rule** dialog box, see [“Creating New Matching Rules” on page 412](#).
- Select a matching rule entry and click **Delete** to delete the rule.

The **Edit Profile Matching Ruleset** dialog box ([Figure 177 on page 415](#)) displays an updated list of matching rule entries

6. Click **OK** to save the changes.

The **Local Profile Selection** page ([Figure 171 on page 409](#)) displays an updated list of local profiles selection rule sets.

Deleting Profile Selection Rule Sets

To delete the existing profile selection rule set using the Web GUI:

1. Select **Diameter Configuration > Policy > Local Profile Selection**.

The **Local Profile Selection** page ([Figure 171 on page 409](#)) appears.

2. Select the profile selection rule set entry that you want to delete.

3. Click **Delete**.

A confirmation dialog box is displayed.

4. Click **Yes** to delete the profile selection rule set.

The **Local Profile Selection** page ([Figure 171 on page 409](#)) displays an updated list of local profiles selection rule sets.

Administering Request Routing Rules

IN THIS CHAPTER

- Request Routing Rules Overview | 417
- Configuring Request Routing Rules | 419

This chapter describes how SBR Carrier routes requests and how to administer routing rules. It contains the following sections:

Request Routing Rules Overview

SBR Carrier maintains implicit and explicit routing rules for both authentication and authorization requests. When SBR Carrier receives a request, it examines the request to determine whether the type of request is authentication or authorization. Once the server determines the type of request, it consults the respective routing rules to determine how to route the request. The type of routing rules defined for authentication and authorization request are:

- **Implicit Routing Rules**—Specifies the routing rules to be configured along with the function for some network element functions. For more information about the implicit routing rules and how to configure them, see [“Configuring Implicit Routing Rules” on page 369](#).
- **Explicit Routing Rules**—Provides greater flexibility for routing requests when implicit routing is not enough. For more information about the explicit routing rules and how to configure them, see [“Defining Explicit Routing Rules” on page 421](#).

Authentication requests are applicable to the HSS function and the Downstream (server) function, whereas for authorization requests the only network element function associated is the Diameter downstream server function.

Routing Rule Evaluation and Rule Priorities

When a request is received, SBR Carrier identifies the request type and then evaluates the associated routing rules. First, the server evaluates any explicit routing rules defined for the request type. The order

in which explicit routing rules are evaluated is user-defined. If no match is found, the server evaluates the implicit routing rules defined for the request type.

When a match is found, SBR Carrier processes the request by routing it to the specified destination. In some cases, this may mean simply forwarding the request to the destination. In other cases, for example if the HSS is listed as the destination, SBR Carrier check with the HSS for the subscriber credentials and then take the appropriate action.

If no routing rule match is found, SBR Carrier sends a response with an appropriate result code to the requestor.

Both the implicit and explicit routing rules can use the realm portion of the NAI decoration (username) to determine how to route requests. If the request does not contain an NAI, the Destination-Realm in the request is used. If the realm to which the request is to be routed is listed in the **Self Names** field, the realm is ignored and the request is treated as if it contained no realm. If no realm is present in either the NAI or the Destination-Realm, the request is considered to be local.

NOTE: For convenience, the Web GUI displays routing rules in the order in which they are evaluated.

Table 42 on page 418 summarizes the priorities and routing rules for authentication requests. Authentication requests are applicable to the HSS function and the Downstream (server) function.

Table 42: Summary of Routing Rule Priorities for Authentication Requests

Priority	Rule	Functions	
		HSS	Downstream Diameter Server
1	Explicit	✓	✓
2	Implicit-IMSI Prefix	✓	Not applicable
3	Implicit-Realm	✓	✓
4	Implicit-Default IMSI	✓	Not applicable
5	Implicit-Default Realm	✓	✓

Table 43 on page 419 summarizes the priorities and routing rules for authorization requests. Authorization requests are applicable only to the Downstream (server) function.

Table 43: Summary of Routing Rule Priorities for Authorization Requests

Priority	Rule	Downstream Diameter Server
1	Explicit	✓
2	Implicit-Realm	✓
3	Implicit-Default Realm	✓

Configuring Request Routing Rules

To configure request routing rules for an authentication or authorization request type using the Web GUI:

1. Select **Diameter Configuration > Request Routing > Authentication Routing Rules** or **Authorization Routing Rule**.

The **Request Routing** page for the particular request type appears. As an example, [Figure 178 on page 420](#) shows the **Authentication Routing Rules** page.

Figure 178: Authentication Routing Rules Page

The screenshot displays the 'Authentication Routing Rules' configuration page in the Steel-Belted Radius Carrier interface. The page is divided into two main sections: 'Explicit Routing Rules' and 'Implicit Routing Rules'. Both sections contain a table with columns 'Target' and 'Description'. The 'Explicit Routing Rules' section has 'Add', 'Edit', and 'Delete' buttons to its right. The 'Implicit Routing Rules' section is currently empty. At the bottom of the page, there are 'Save', 'Reset', and 'Refresh' buttons. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the user 'root'.

Referring to [Figure 178 on page 420](#), the **Explicit Routing Rules** area displays any explicit routing rules already defined for the request type. Any implicit routing rules are displayed in the **Implicit Routing Rules** area.

2. Define explicit routing rules. For more information about defining explicit routing rules, see [“Defining Explicit Routing Rules” on page 421](#).

You can modify the explicit routing rules entry by using the **Edit** and **Delete** buttons.

NOTE: The implicit routing rules are automatically populated in the **Implicit Routing Rules** area after assigning functions to the Diameter remote network element. For more information about assigning functions to the Diameter remote network element, see [“Assigning Functions to the Diameter Remote Network Element” on page 366](#).

3. Define the order of explicit routing rules by selecting each explicit routing rule and using the **Up** or **Down** arrow.
4. Click **Save** to save the configuration.

Defining Explicit Routing Rules

Explicit routing rules provide greater flexibility for routing requests when implicit routing is not sufficient. Explicit routing rules can be configured for both the authentication and authorization request types.

When defining explicit routing rules, you define a condition or set of conditions and select a target to which the request is to be routed.

When SBR Carrier receives a request, it examines the request to determine the request type. It then consults the respective routing rules to determine how to route the request. If the routing rules contain any user-defined explicit routing rules, they are evaluated before implicit routing rules. The order in which explicit routing rules are evaluated is user-defined. For a rule to be selected, all conditions must be true. If a rule is not selected, the next rule in the list of explicit routing rules is examined, and so on. If a match for all conditions cannot be found in the explicit routing rules, the implicit routing rules are examined.

NOTE: Multiple conditions can be defined for an explicit routing rule. For the explicit routing rule to be selected to route the request, all conditions defined in the rule must be true.

To define explicit routing rules using the Web GUI:

1. In the **Request Routing** page (for a sample **Authentication Routing Rules** page, see [Figure 178 on page 420](#)), click **Add** to add a new explicit routing rule.

The **Add Explicit Routing Rule** dialog box for the selected routing rule appears. [Figure 179 on page 422](#) shows this dialog box for authorization requests. The dialog is same for both the authorization and authentication requests, only the respective target varies.

Figure 179: Add Explicit Routing Rule Dialog for Authorization Requests

The screenshot shows a dialog box titled "Add Explicit Routing Rule". At the top, there is a "Target:" label followed by a dropdown menu. Below this is a section labeled "Conditions" which contains a table. The table has two columns: "Type" and "Description". The table is currently empty, displaying the text "No Conditions to Display". Below the table are three buttons: "Add", "Edit", and "Delete". At the bottom right of the dialog are "OK" and "Cancel" buttons.

2. Select a target for the authentication or authorization request type from the **Target** list.

[Table 44 on page 422](#) shows the targets for each request type.

Table 44: Request Type and Target

Request Type	Target
Authentication	<ul style="list-style-type: none"> • HSS • Downstream (Diameter server)
Authorization	<ul style="list-style-type: none"> • Downstream (Diameter server)

3. Define a condition for the explicit routing rule. For more information about defining conditions for the explicit routing rule, see [“Defining Conditions for the Explicit Routing Rule” on page 423](#).

You can modify the condition entry by using the **Edit** and **Delete** buttons

4. Click **OK** to save the configuration.

The **Explicit Routing Rules** area in the **Request Routing** page (For a sample **Authentication Routing Rules** page, see [Figure 178 on page 420](#)) displays the updated list of explicit routing rules.

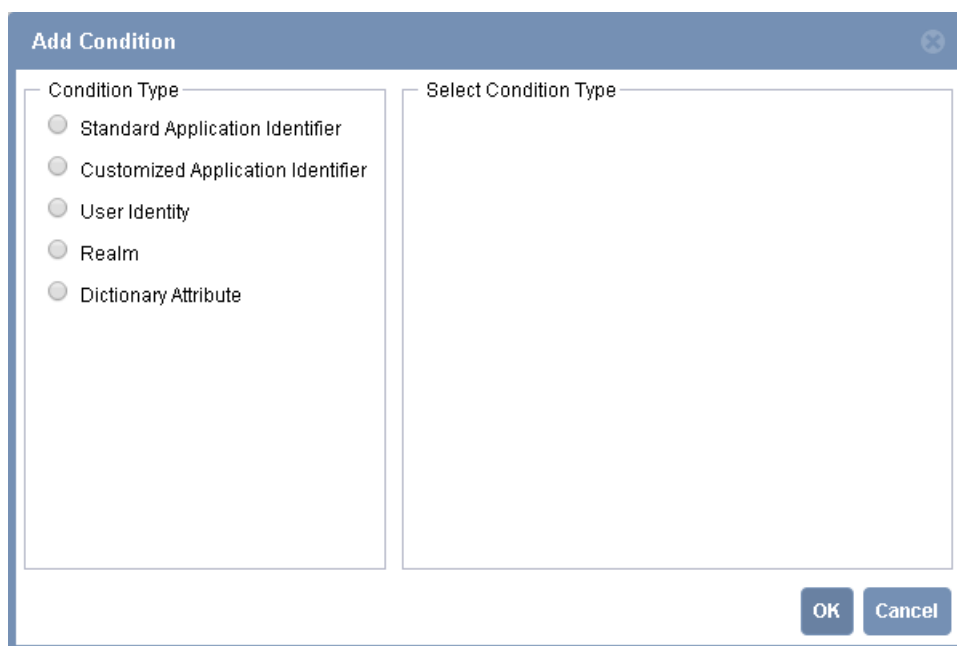
Defining Conditions for the Explicit Routing Rule

To define a condition for the explicit routing rule using the Web GUI:

1. In the **Add Explicit Routing Rule** dialog box ([Figure 179 on page 422](#)), click **Add**.

The **Add Condition** dialog box ([Figure 180 on page 423](#)) appears.

Figure 180: Add Condition Dialog



The **Add Condition** dialog box is shown. It has a title bar with the text "Add Condition" and a close button. The dialog is divided into two main sections. The left section, titled "Condition Type", contains a list of five radio button options: "Standard Application Identifier", "Customized Application Identifier", "User Identity", "Realm", and "Dictionary Attribute". The right section, titled "Select Condition Type", is currently empty. At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

2. In the **Add Condition** dialog box, you can:
 - Select the **Standard Application Identifier** option to define a standard application identifier condition. For more information about defining standard application identifier condition, see [“Defining Standard Application Identifier Conditions” on page 424](#).
 - Select the **Customized Application Identifier** option to define a customized application identifier condition. For more information about defining customized application identifier condition, see [“Defining Customized Application Identifier Conditions” on page 425](#).
 - Select the **User Identity** option to define a user identity condition. For more information about defining user identity condition, see [“Defining User Identity Conditions” on page 426](#).

- Select the **Realm** option to define a realm name condition. For more information about defining realm name condition, see [“Defining Realm Name Conditions” on page 427](#).
- Select the **Dictionary Attribute** option to define a dictionary attribute condition. For more information about defining dictionary attribute condition, see [“Defining Dictionary Attribute Conditions” on page 428](#).

Defining Standard Application Identifier Conditions

This standard application identifier condition allows you to define the routing rule based on the Diameter application identifier in the request. The available condition tests are **equal to** or **not equal to** the standard Diameter application identifier values, such as Diameter network access server application or Diameter EAP application.

To define the standard application identifier condition for the explicit routing rule using the Web GUI:

1. In the **Add Condition** dialog box ([Figure 180 on page 423](#)), select the **Standard Application Identifier** option.

The **Standard Application Identifier** pane ([Figure 181 on page 424](#)) appears.

Figure 181: Standard Application Identifier Pane

The screenshot shows a web-based 'Add Condition' dialog. On the left, under 'Condition Type', five radio buttons are listed: 'Standard Application Identifier' (selected), 'Customized Application Identifier', 'User Identity', 'Realm', and 'Dictionary Attribute'. On the right, the 'Standard Application Identifier' pane is active. It contains a 'Name' field with the text 'DiameterApplicationIdentifier'. Below it is a 'Condition Test' dropdown menu, and further down is a 'Value' dropdown menu. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

2. Select either **Is equal to** or **Is not equal to** in the **Condition Test** field.

NOTE: The **Name** field displays the name of the Diameter application identifier. This is a read-only field.

3. Select a standard Diameter application identifier from the **Value** list.
4. Click **OK**.

The **Conditions** area in the **Add Explicit Routing Rule** dialog box (Figure 179 on page 422) displays the updated list of condition rules.

Defining Customized Application Identifier Conditions

The customized application identifier condition is the same as the standard application identifier condition, the only difference is that here you are allowed to enter a custom Diameter application identifier value. The available condition tests are **equal to** or **not equal to** the specified application identifier value.

To define a custom application identifier condition for the explicit routing rule using the Web GUI:

1. In the **Add Condition** dialog box (Figure 180 on page 423), select the **Customized Application Identifier** option.

The **Customized Application Identifier** (Figure 182 on page 425) pane appears.

Figure 182: Customized Application Identifier Pane

The screenshot shows a dialog box titled "Add Condition". On the left, under "Condition Type", there are five radio button options: "Standard Application Identifier", "Customized Application Identifier" (which is selected), "User Identity", "Realm", and "Dictionary Attribute". On the right, under "Customized Application Identifier", there are three fields: "Name" (containing the text "DiameterApplicationIdentifier"), "Condition Test" (a dropdown menu), and "Value" (a text input field). At the bottom right of the dialog are "OK" and "Cancel" buttons.

2. Select either **Is equal to** or **Is not equal to** in the **Condition Test** field.

NOTE: The **Name** field displays the name of the Diameter application identifier. This is a read-only field.

3. Enter a valid application identifier in the **Value** field.
4. Click **OK**.

The **Conditions** area in the **Add Explicit Routing Rule** dialog box (Figure 179 on page 422) displays the updated list of condition rules.

Defining User Identity Conditions

The user identity condition allows you to define the routing rule based on the user identity, for example the IMSI or username value contained in the request. The available condition tests are **equal to**, **not equal to**, **present**, **not present**, **has this prefix**, and **has this suffix**.

To define a user identity condition for the explicit routing rule using the Web GUI:

1. In the **Add Condition** dialog box (Figure 180 on page 423), select the **User Identity** option.

The **User Identity** pane (Figure 183 on page 426) appears.

Figure 183: User Identity Pane

The screenshot shows a dialog box titled "Add Condition". On the left, under "Condition Type", there are five radio buttons: "Standard Application Identifier", "Customized Application Identifier", "User Identity" (which is selected), "Realm", and "Dictionary Attribute". On the right, under "User Identity", there is a "Name" field containing "User-Name", a "Condition Test" dropdown menu, and a "Value" text input field. At the bottom right of the dialog are "OK" and "Cancel" buttons.

2. Select a condition in the **Condition Test** field.

NOTE: The **Name** field displays the name of the user identity. This is a read-only field.

3. Enter the user identity in the **Value** field.
4. Click **OK**.

The **Conditions** area in the **Add Explicit Routing Rule** dialog box (Figure 179 on page 422) displays the updated list of condition rules.

Defining Realm Name Conditions

The realm name condition allows you to define the routing rule based on the realm name value contained in the request. The available condition tests are **equal to**, **not equal to**, **present**, **not present**, **has this prefix**, or **has this suffix**.

To define a realm name condition for the explicit routing rule using the Web GUI:

1. In the **Add Condition** dialog box (Figure 180 on page 423), select the **Realm** option.

The **Realm** pane (Figure 184 on page 427) appears.

Figure 184: Realm Pane

2. Select a condition in the **Condition Test** field.

NOTE: The **Name** field displays the realm string. This is a read-only field.

3. Enter the realm name in the **Value** field.
4. Click **OK**.

The **Conditions** area in the **Add Explicit Routing Rule** dialog box (Figure 179 on page 422) displays the updated list of condition rules.

Defining Dictionary Attribute Conditions

This dictionary attribute condition allows you to specify the route based on the value of a particular dictionary attribute. The attribute is chosen from a list and the value of the attribute is user defined. The available condition tests are **equal to**, **not equal to**, **present**, **not present**, **has this prefix**, **has this suffix** or **is within this range**.

To define a dictionary attribute condition for the explicit routing rule using the Web GUI:

1. In the **Add Condition** dialog box (Figure 180 on page 423), select the **Dictionary Attribute** option.

The **Dictionary Attribute** pane (Figure 185 on page 428) appears.

Figure 185: Dictionary Attribute Pane

The screenshot shows a dialog box titled "Add Condition". On the left, under "Condition Type", there are five radio button options: "Standard Application Identifier", "Customized Application Identifier", "User Identity", "Realm", and "Dictionary Attribute" (which is selected). On the right, under "Dictionary Attribute", there are three input fields: "Attribute" (a dropdown menu), "Condition Test" (a dropdown menu), and "Value" (a text input field). At the bottom right of the dialog are "OK" and "Cancel" buttons.

2. Select a dictionary attribute from the **Attribute** list.

You can search the attributes by entering the attribute name in the text box.

3. Select a condition in the **Condition Test** field.
4. Enter the value for the dictionary attribute in the **Value** field.
5. Click **OK**.

The **Conditions** area in the **Add Explicit Routing Rule** dialog box (Figure 179 on page 422) displays the updated list of condition rules.

Displaying Diameter Statistics

IN THIS CHAPTER

- Statistics Overview | 429
- Diameter Statistics | 429
- Routing Rule Statistics | 439

This chapter describes the **Statistics** page, which you can use to view and analyze Diameter and routing rules statistics.

This chapter covers the following sections:

Statistics Overview

The **Statistics** page lets you view various levels of Diameter statistics such as host statistics, peer statistics, and request routing statistics.

Diameter Statistics

The Diameter **Statistics** page displays statistics related to the local host and its peers.

Local Diameter Host Statistics

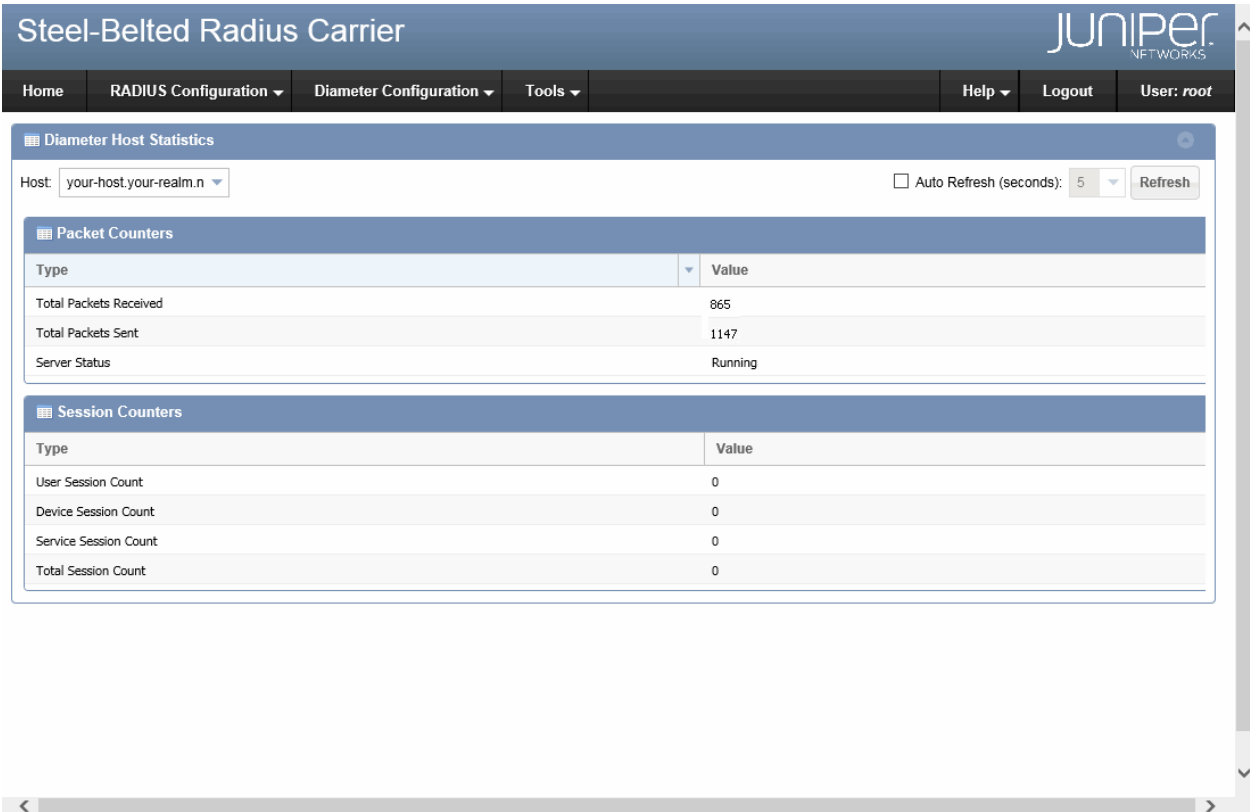
Local host statistics include the host status and the statistics related to total number of packets and sessions.

To view the local host (server) statistics using the Web GUI:

1. Select **Diameter Configuration > Statistics > Diameter > Host**.

The **Diameter Host Statistics** page ([Figure 186 on page 430](#)) appears.

Figure 186: Diameter Host Statistics Page



2. Select the Diameter host entry for which you want to view the statistics from the **Host** list.
- The **Diameter Host Statistics** page (Figure 186 on page 430) displays statistics for the selected Diameter host.
- Table 45 on page 430 describes each statistic.

Table 45: Local Diameter Host (Server) Statistics

Statistic	Meaning
Packet Counters	
Total Packets Received	The total number of packets received.
Total Packets Received	The total number of packets transmitted.
Server Status	The only value supported is: running—The server is currently running.
Session Counters	

Table 45: Local Diameter Host (Server) Statistics (*continued*)

Statistic	Meaning
User Session Count	The total number of user sessions available in the Diameter host.
Device Session Count	The total number of device sessions available in the Diameter host.
Service Session Count	The total number of service sessions available in the Diameter host.
Total Session Count	The sum of user, device, and service session totals.

- Optionally, select the **Auto Refresh (seconds)** check box and select a time period for refreshing the display from the drop-down list. Default value is 5 seconds.

By default, the auto-refresh feature is disabled.

You can click the **Refresh** button to refresh the display.

Diameter Peer Statistics

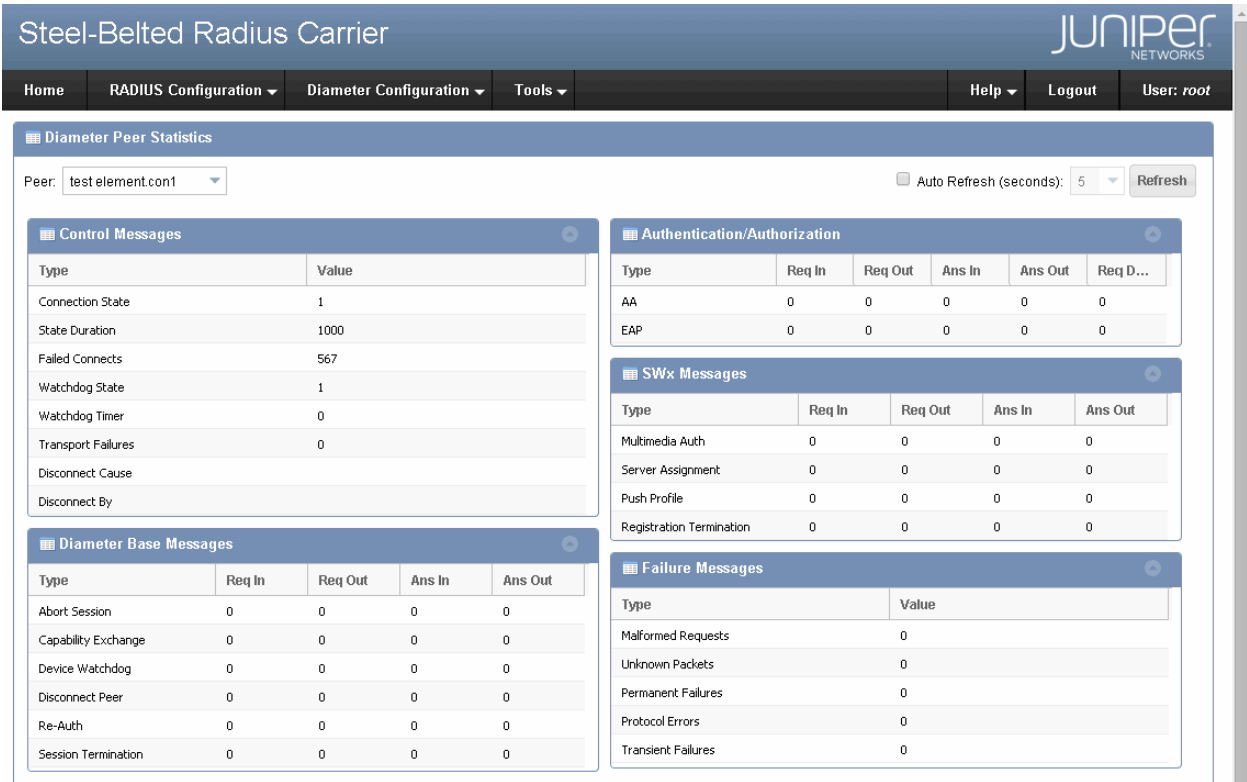
Statistics are collected and can be displayed for each Diameter peer.

To view Diameter peer statistics using the Web GUI:

- Select **Diameter Configuration > Statistics > Diameter > Peers**.

The **Diameter Peer Statistics** page ([Figure 187 on page 432](#)) appears.

Figure 187: Diameter Peer Statistics Page



2. Select the Diameter peer entry for which you want to view statistics from the **Peer** list.
- The **Diameter Peer Statistics** page (Figure 187 on page 432) displays statistics for the selected Diameter peer.
- Table 46 on page 432 describes each statistic.

Table 46: Diameter Peer Statistics

Statistic	Meaning
Control Messages	

Table 46: Diameter Peer Statistics (*continued*)

Statistic	Meaning
Connection State	<p>Connection state of the peer's state machine with which the server is communicating.</p> <p>1=Closed—Connection with the peer is closed.</p> <p>2=waitConnAck—Waiting for an acknowledgment from the peer.</p> <p>3=waitlCea—Waiting for a Capabilities-Exchange-Answer from the peer.</p> <p>4=elect—When the peer and the server are trying to bring up the connection simultaneously, an election process begins which determines which socket remains open.</p> <p>5=waitReturns—Waiting for election returns.</p> <p>6=r-open—Responder transport connection is used for communication.</p> <p>7=i-open—Initiator transport connection is used for communication.</p> <p>8=closing—Actively closing and doing cleanup.</p>
State Duration	Peer state duration, measured in centiseconds.
Failed Connects	If there is no transport connection with a peer, this is the number of times the server attempts to connect to that peer. This value is reset on disconnection.
Watchdog State	<p>1=okay—Indicates that the connection is presumed working.</p> <p>2=suspect—Indicates that the connection is possibly congested or down.</p> <p>3=down—The peer is no longer reachable, causing the transport connection to be shutdown.</p> <p>4=reopen—Three watchdog messages are exchanged with accepted round trip times, and the connection to the peer is considered stabilized.</p>
Watchdog Timer	The interval between watchdog events.
Transport Failures	Number of unexpected transport failures.

Table 46: Diameter Peer Statistics (*continued*)

Statistic	Meaning	
Disconnect Cause	<p>The last cause for a peer’s disconnection:</p> <p>rebooting—A scheduled reboot is imminent.</p> <p>busy—The peer's internal resources are constrained, and it has determined that the transport connection needs to be shutdown.</p> <p>doNotWantToTalk—The peer has determined that it does not see a need for the transport connection to exist, since it does not expect any messages to be exchanged in the foreseeable future.</p> <p>electionLost—The peer has determined that it has lost the election process and has therefore disconnected the transport connection.</p>	
Disconnect By	<p>Identifies who initiated the disconnect:</p> <p>host—The server initiated the disconnect.</p> <p>peer—The peer with which the server was connected initiated the disconnect.</p>	
Diameter Base Messages		
Abort Session	Req In	Number of Abort-Session-Request messages received from the peer.
	Req Out	Number of Abort-Session-Request messages sent to the peer.
	Ans In	Number of Abort-Session-Answer messages received from the peer.
	Ans Out	Number of Abort-Session-Answer messages sent to the peer.

Table 46: Diameter Peer Statistics (*continued*)

Statistic	Meaning	
Accounting	Req In	Number of Accounting-Request messages received from the peer.
	Req Out	Number of Accounting-Request messages sent to the peer.
	Ans In	Number of Accounting-Answer messages received from the peer.
	Ans Out	Number of Accounting-Answer messages sent to the peer.
	Req Dropped	Number of Accounting-Request messages dropped by the peer.
Capability Exchange	Req In	Number of Capabilities-Exchange-Request messages received from the peer.
	Req Out	Number of Capabilities-Exchange-Request messages sent to the peer.
	Ans In	Number of Capabilities-Exchange-Answer messages received from the peer.
	Ans Out	Number of Capabilities-Exchange-Answer messages sent to the peer.
Device Watchdog	Req In	Number of Device-Watchdog-Request messages received from the peer.
	Req Out	Number of Device-Watchdog-Request messages sent to the peer.
	Ans In	Number of Device-Watchdog-Answer messages received from the peer.
	Ans Out	Number of Device-Watchdog-Answer messages sent to the peer.

Table 46: Diameter Peer Statistics (*continued*)

Statistic	Meaning	
Disconnect Peer	Req In	Number of Disconnect-Peer-Request messages received from the peer.
	Req Out	Number of Disconnect-Peer-Request messages sent to the peer.
	Ans In	Number of Disconnect-Peer-Answer messages received from the peer.
	Ans Out	Number of Disconnect-Peer-Answer messages sent to the peer.
Re-Auth	Req In	Number of Re-Auth-Request messages received from the peer.
	Req Out	Number of Re-Auth-Request messages sent to the peer.
	Ans In	Number of Re-Auth-Answer messages received from the peer.
	Ans Out	Number of Re-Auth-Answer messages sent to the peer.
Session Termination	Req In	Number of Session-Termination-Request messages received from the peer.
	Req Out	Number of Session-Termination-Request messages sent to the peer.
	Ans In	Number of Session-Termination-Answer messages received from the peer.
	Ans Out	Number of Session-Termination-Answer messages sent to the peer.
Authentication/Authorization		

Table 46: Diameter Peer Statistics (*continued*)

Statistic	Meaning	
AA	Req In	Number of AA-Request messages received from the peer.
	Req Out	Number of AA-Request messages sent to the peer.
	Ans In	Number of AA-Answer messages received from the peer.
	Ans Out	Number of AA-Answer messages sent to the peer.
	Req Dropped	Number of AA-Request messages dropped by the peer.
EAP	Req In	Number of Diameter-EAP-Request messages received from the peer.
	Req Out	Number of Diameter-EAP-Request messages sent to the peer.
	Ans In	Number of Diameter-EAP-Answer messages received from the peer.
	Ans Out	Number of Diameter-EAP-Answer messages sent to the peer.
	Req Drop	Number of Diameter-EAP-Request messages dropped by the peer.
SWx Messages		

Table 46: Diameter Peer Statistics (*continued*)

Statistic	Meaning	
Multimedia Auth	Req In	Number of Multimedia-Authentication-Request messages received from the peer.
	Req Out	Number of Multimedia-Authentication-Request messages sent to the peer.
	Ans In	Number of Multimedia-Authentication-Answer messages received from the peer.
	Ans Out	Number of Multimedia-Authentication-Answer messages sent to the peer.
Server Assignment	Req In	Number of Server Assignment-Request messages received from the peer.
	Req Out	Number of Server Assignment-Request messages sent to the peer.
	Ans In	Number of Server Assignment-Answer messages received from the peer.
	Ans Out	Number of Server Assignment-Answer messages sent to the peer.
Push Profile	Req In	Number of Push Profile-Request messages received from the peer.
	Req Out	Number of Push Profile-Request messages sent to the peer.
	Ans In	Number of Push Profile-Answer messages received from the peer.
	Ans Out	Number of Push-Profile-Answer messages sent to the peer.

Table 46: Diameter Peer Statistics (*continued*)

Statistic	Meaning	
Registration Termination	Req In	Number of Registration-Termination-Request messages received from the peer.
	Req Out	Number of Registration-Termination-Request messages sent to the peer.
	Ans In	Number of Registration-Termination-Answer messages received from the peer.
	Ans Out	Number of Registration-Termination-Answer messages sent to the peer.
Failure Messages		
Malformed Requests	Number of malformed Diameter packets received from the peer.	
Unknown Packets	Number of Diameter packets of unknown type which were received from the peer.	
Permanent Failures	Number of permanent failures returned to peer.	
Protocol Errors	Number of protocol errors returned to peer, not including redirects.	
Transient Failures	Number of transient failures.	

- Optionally, select the **Auto Refresh (seconds)** check box and select a time period for refreshing the display from the drop-down list. Default value is 5 seconds.

By default, the auto-refresh feature is disabled.

You can click the **Refresh** button to refresh the display.

Routing Rule Statistics

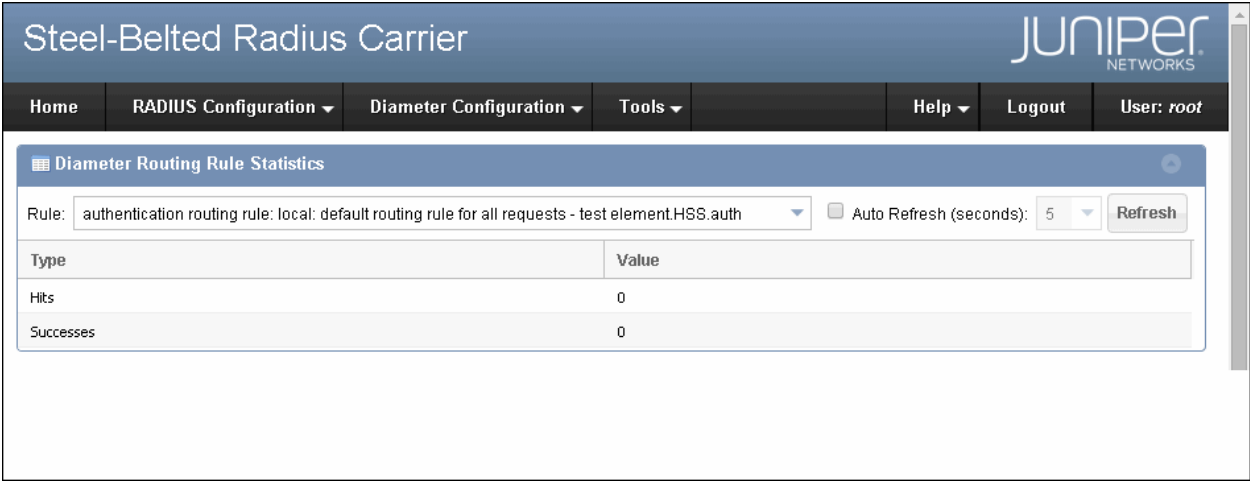
SBR Carrier collects and displays hit and success counts for each routing rule defined in the server (implicit, explicit and default).

To display the statistics for a routing rule:

1. Select **Diameter Configuration > Statistics > Routing Rules**.

The **Diameter Routing Rule Statistics** page (Figure 188 on page 440) appears.

Figure 188: Diameter Routing Rules Statistics Page



2. Select the routing rule entry for which you want to display statistics from the **Rule** list.

The **Diameter Routing Rule Statistics** page (Figure 188 on page 440) displays statistics for the selected routing rule.

Table 47 on page 440 lists routing rule statistics.

Table 47: Routing Rule Statistics

Statistic	Meaning
Hits	Total number of times the routing rule has been selected to route a request.
Successes	Total number of operations that completed successfully as a result of selecting the rule.

3. Optionally, select the **Auto Refresh (seconds)** check box and select a time period for refreshing the display from the drop-down list. Default value is 5 seconds.

By default, the auto-refresh feature is disabled.

You can click the **Refresh** button to refresh the display.

5

PART

Back-End Authentication and Accounting Methods

[Configuring SQL Authentication | 442](#)

[Configuring SQL Accounting | 458](#)

[Configuring LDAP Authentication | 471](#)

[Signalware SIGTRAN Gateway Support | 484](#)

[Proxy RADIUS Authentication and Accounting | 485](#)

[HSS-Subscriber Database | 486](#)

Configuring SQL Authentication

IN THIS CHAPTER

- Overview of SQL Authentication | 442
- Configuring SQL Authentication | 445
- Connecting to the SQL Database | 446
- SQL Statement Construction | 447
- Working with Stored Procedures in Oracle | 453
- Working with Stored Procedures in MS-SQL | 455

This chapter presents an overview of SQL authentication and describes how to configure SQL authentication in Steel-Belted Radius Carrier. This chapter contains these topics:

NOTE: Throughout this chapter, the term *attributes* refers to both standard RADIUS attributes and structured attributes. For information about specifying structured attributes, see the *SBR Carrier Reference Guide*.

NOTE: This SQL authentication information is unrelated to the SQL (MySQL) server used by Steel-Belted Radius Carrier.

Overview of SQL Authentication

Steel-Belted Radius Carrier can authenticate against records stored in an external SQL database. Any RADIUS attributes, such as username and password, can be used to query the database.

External database authentication is typically used when an organization already has a large amount of user information stored in a SQL database, and this information is to be used to authenticate these users using

RADIUS. Authentication against an existing database extends authentication services to user accounts without requiring an administrator to enter user information into the Steel-Belted Radius Carrier database.

Steel-Belted Radius Carrier offers the SQL authentication feature as a plug-in software module. Key features of the SQL plug-in include:

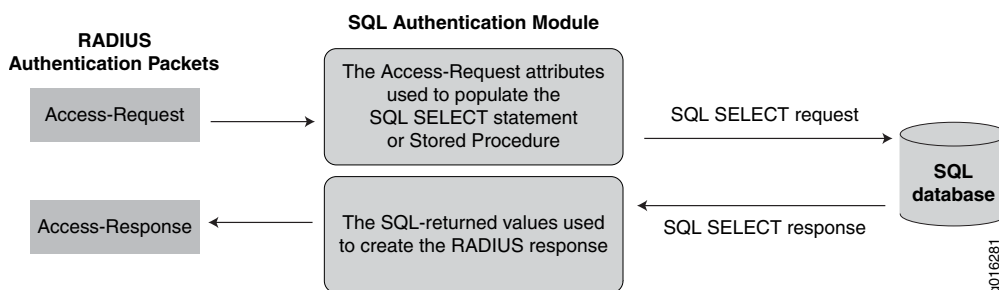
- The SQL statement is completely user-specified, enabling support for existing tables with existing field names and formats.
- The SQL statement supports a wide range of arithmetic and string expressions as part of the statement.
- The SQL statement is parameterized, so it is compiled once, and each execution uses variable data without need for recompilation.
- Multiple authentications may be overlapped at the same time.
- The SQL authentication method, which appears in the **Authentication Methods** page in Web GUI, can be activated/deactivated and ordered with respect to other authentication methods.
- Multiple instances of the SQL authentication module can operate simultaneously, enabling authentication to multiple databases.
- If the database connection drops, it is automatically reestablished after a configurable timeout without Steel-Belted Radius Carrier being restarted.
- Data from the database can be returned as attributes in the Access-Accept message.

NOTE: Although Steel-Belted Radius Carrier does its best to provide uniformity in the operation of databases from different vendors, differences occur, particularly in the way SQL statements are interpreted. The capabilities of the SQL authentication module depend on the capabilities of the underlying databases and their clients; things that work with one database may not work with another.

SQL Authentication Process

Any RADIUS attribute (or Steel-Belted Radius Carrier request variable) from the request can be used in an SQL SELECT statement. Any return list attribute (that is, a Steel-Belted Radius Carrier response variable) can be retrieved from a SQL database and returned in a RADIUS access response message. See [Figure 189 on page 444](#).

Figure 189: SQL Authentication Process



Stored Procedures

A stored procedure is a sequence of SQL statements that form a logical unit and perform a particular task. You can use stored procedures to encapsulate a set of queries or operations that can be executed repeatedly on a database server. For example, you can code operations on an employee database, such as password lookup, as stored procedures that can be executed by application code.

Stored procedures can be compiled and executed with different parameters and results. Stored procedures can use any combination of input parameters (the values passed to the stored procedure at execution time) and output parameters (the values set or returned by the stored procedure to the calling application or environment).

You can write stored procedures for SQL that communicate with Steel-Belted Radius Carrier via input and output parameters to implement custom functions. Stored procedures let you use server-side processing on the SQL server to manipulate the information specified by variables. How you use these stored procedures depends on details specific to the implementation of SQL that you are using.

NOTE: Do not configure a stored procedure to call the same attribute more than once. Doing so may cause Steel-Belted Radius Carrier to fail.

For information about using stored procedures with the Oracle SQL database, see [“Working with Stored Procedures in Oracle” on page 453](#). For information about using stored procedures with the Microsoft SQL database, see [“Working with Stored Procedures in MS-SQL” on page 455](#).

Connectivity Issues

Steel-Belted Radius Carrier may encounter serious problems if the connection between Oracle and Steel-Belted Radius Carrier becomes unstable. The most common reasons for a connection becoming unstable are:

- Slow or unreliable network response times

- Interruptions in connectivity caused by intervening network devices, such as a firewall timing out the connection

To prevent connectivity problems, consider implementation of one of the following solutions:

- To minimize problems caused by intervening firewalls, configure your firewall to pass traffic on the Oracle communications ports between the Steel-Belted Radius Carrier server and the Oracle server without restriction.
- To minimize network latency and firewall-related problems, move the Steel-Belted Radius Carrier server to the same network segment as the Oracle server.
- If moving your Steel-Belted Radius Carrier server is not feasible, locate a second Steel-Belted Radius Carrier server on the same network segment as your Oracle server, and configure your current Steel-Belted Radius Carrier server to proxy all authentication requests to this new device. This configuration enables you to open RADIUS ports on the firewall only for the Steel-Belted Radius Carrier server (instead of opening RADIUS ports for all network access servers). Because proxy functions in Steel-Belted Radius Carrier do not require an uninterrupted connection to process requests, this solution enables you to retain your current firewall timeout settings.

Configuring SQL Authentication

You must configure both Steel-Belted Radius Carrier and the SQL database to support SQL authentication. The configuration procedure must be tailored to the database that you use. However, all procedures must give the following results:

- The required transport must be in place between SQL client software and the SQL server.
- The SQL server must be configured via a plug-in to coordinate with SQL client software.
- The Steel-Belted Radius Carrier server must be configured to communicate with the SQL client software to interact with the back-end SQL server to perform stored procedures or SQL queries.

Files

The files in [Table 48 on page 445](#) establish settings for configuring SQL authentication in Steel-Belted Radius Carrier. For more information about these files, refer to the *SBR Carrier Reference Guide*.

Table 48: SQL Authentication Files

File Name	Function
radsql.aut	Configures settings for SQL authentication using Oracle.

Table 48: SQL Authentication Files (*continued*)

File Name	Function
radsqldb.aut	Configures settings for SQL authentication using JDBC.

Using the SQL Authentication Configuration File

To configure SQL authentication, you must edit the authentication configuration files, **radsqldb.aut** (Solaris with Oracle) or **radsqldb.aut** (Solaris with JDBC), which are located in the same directory that contains the Steel-Belted Radius Carrier daemon. Most of these options may be left at their original settings; however, you must modify certain options to accommodate your own database.

After you complete your changes to the authentication configuration files and restart Steel-Belted Radius Carrier, the **InitializationString** value that you entered in the [Bootstrap] section of the configuration file appears in the **Authentication Methods** page. You can then enable, disable, or prioritize your SQL database like any other authentication method in the list.

Using Multiple SQL Authentication Methods

You can configure Steel-Belted Radius Carrier to authenticate users against more than one SQL database. Each database that you set up in this way becomes a separate selection in the **Authentication Methods** page.

To add an additional database, create a new configuration file with extension **.aut** in the same directory as **radsqldb.aut** (Solaris with Oracle) or **radsqldb.aut** (Solaris with JDBC). You can give this file any name you like, provided its extension is **.aut**. At startup, Steel-Belted Radius Carrier enumerates all **.aut** files to create its list of authentication methods.

When creating the new file, start by copying the original **.aut** file. Be sure to change its **InitializationString** entry to a unique authentication method name; otherwise, Steel-Belted Radius Carrier has no way of distinguishing between the different methods in the authentication methods list.

Connecting to the SQL Database

Upon startup, the SQL authentication module connects to the database, using settings specified by a connect string specified in the configuration file. The connect string contains information such as the name and location of the database, and the password required to connect. The connect string is passed to the database client to establish the connection.

While a sample connect string is provided in the original configuration file, you must configure the **Connect** entry of the configuration file with a connect string appropriate to your database.

The password for database access must be provided as part of the connect string. If it is not, the connection fails.

If the initial attempt to connect to the database fails, or if a processing error occurs that the SQL authentication module interprets as a database connection failure, the SQL authentication module drops the connection and attempts to establish a new connection after a period of time. In the interim, all authentication requests are ignored.

The SQL authentication module uses an exponential back-off strategy in determining how long to wait before attempting a new connection, as well as how frequently this attempt is made. After the first dropped connection, it waits a certain amount of time before attempting to reconnect. If this attempt to reconnect also fails, it waits for twice the amount of time before trying again; and so on, up to some maximum wait time. The initial and maximum wait times are configurable.

NOTE: Detailed error information may not be available if there is an error processing the database logon at connect time. A numeric result code is displayed in the log. You may need to refer to product-specific documentation to decode this result code. With Oracle on Solaris, you can use the `oerr facility-code error-number` command with a facility code of `oerr` from the command shell.

SQL Statement Construction

The authentication transaction is based on an SQL query that returns a password (and possibly other information) based on the name entered by the user attempting to log in.

While a sample SQL query is provided in the original configuration file, you must configure the SQL entry of the configuration file with a query appropriate to your database. The query you enter must be either a **SQL SELECT** or **SQL EXECUTE** statement that contains additional syntax elements which are preprocessed by the SQL authentication module.

The SQL authentication module executes SQL statements in parameterized form. This means that the SQL statement is compiled once, with parameter markers (usually question marks) as placeholders for data items that vary from one execution to the next. Only upon execution of the statement are the actual data values supplied.

The SQL statement you compose must not include parameter markers directly. Instead, include the names of the parameters where parameter markers would appear, in a format described below. The SQL authentication module translates the SQL statement provided, replacing parameter names with parameter markers before passing the SQL statement to the database engine.

The SQL statement can be very simple. Basically, all that is required is to look up a password and possibly some optional information based on a username. The SQL statement can also be quite complex; it can include inner joins, and it can contain expressions. The underlying database engine is responsible for handling the SQL statement; the SQL authentication module performs no interpretation of the SQL statement other than to translate parameter names to parameter markers.

Example:

```
SELECT password, profile, fullname FROM usertable WHERE username = %name/63s
```

As shown in this example, a parameter consists of a percent sign (%), the name of the parameter and a format specifier. [Table 49 on page 448](#) lists SQL statement parameter names.

Table 49: SQL Statement Parameters

Item	Meaning for SQL Authentication
%OriginalUserName	The original full identification of the user, before any processing (that is, user@realm).
%User	The user portion of OriginalUserName (the section before “@”).
%UserName	The full user identification (user and realm strings) after all stripping and processing has been performed.
%Name	Synonym for UserName.
%EffectiveUser	The name of the user (the section before “@”) as presented to the authentication method. This may be a modified version of the original username.
%Realm	The realm portion of the original user identification (the section after “@”) as presented to the authentication method. This may be a modified version of the original realm name.
%EffectiveRealm	The realm portion of the user identification as presented to the method. This may be a modified version of the original realm name.
%NASName	The name of the network access server that originated the request. This may be the name of the RADIUS clients entry in the database or the value of the NAS-Identifier or NAS-IP-Address attribute.

Table 49: SQL Statement Parameters (continued)

Item	Meaning for SQL Authentication
%NASAddress	The address of the network access server, in dotted notation.
%NASModel	The make/model of the network access server, as specified in the Steel-Belted Radius Carrier database.
%Password	The PAP password.
%AllowedAccessHours	The times that the user is allowed to be logged in.
%RADIUSClientName	The name of the network access server, as specified in a RADIUS clients entry in the Steel-Belted Radius Carrier database.

Along with these parameters, any RADIUS attribute received in the Access-Request can be referred to by using an at-sign (“@”) followed by the name of the attribute. If you need to specify a literal at-sign character in an SQL statement, such as in a UserName, you must use two at-signs in a row. For example:

```
SELECT foo FROM bar WHERE field = "abc@@xyz"
```

Likewise, if you need to specify a literal percent character (“%”) in an SQL statement you must use a two percent characters in a row.

The format specifier describes the database storage format of the column that corresponds to the parameter. It consists of a slash (“/”), a length, and a type, which for SQL authentication is always “s” for string. For example, if the user’s name is stored in the database as a string of up to 63 bytes, you would enter:

```
%name/63s
```

NOTE: Be sure to specify a length no greater than the actual field size in the database. The compilation of the SQL statement may fail if a parameter size greater than the actual field size is specified.

Normally, the only parameter you need to include in the WHERE section of a SQL statement is **%name**. The following SQL statement explains how to look up a password based on a username:

```
SELECT Password FROM Usertable WHERE Username = %name/32
```

```
[Results]
Password=1/32
```

It is not mandatory for you to provide the password in the first column of the SELECT statement. Also, you can change the position of the password in the SELECT statement. When the position of the password is changed, the element-number for the password in the **[Results]** section of the **.aut** file must be changed accordingly.

In the following statement, **%name** is an input parameter used to look up the user's profile:

```
SELECT profile FROM database WHERE username = %name
```

Because there is no password output parameter, no password authentication is performed. The following **[Results]** section of the **.aut** file works correctly with the preceding SELECT statement:

```
[Results]
Password=0
Profile=1/50
Alias=0
```

If the record cannot be found in the database, the authentication attempt fails.

NOTE: If you are not using password checking for authentication, the Password parameter must be set to 0 in the **[Results]** section.

Overlapped Execution of SQL Statements

The SQL authentication module is multi-threaded. SQL authentication can be configured with a maximum number of simultaneous executions of any SQL statement, using the MaxConcurrent entry in the **.aut** file's **[Settings]** section.

NOTE: The MaxConcurrent entry is valid only for Oracle SQL authentication (radsql.aut). It is not valid for JDBC SQL authentication (radsqljdbc.aut).

If MaxConcurrent is set to 1, SQL execution occurs serially, and the SQL execution for each authentication request must complete before execution for the next request may begin.

By increasing MaxConcurrent, it may be possible to increase throughput by overlapping operations, especially if the database server is remote and a large part of the time to complete a statement execution is taken up by network latency. If the database server is local, the point of diminishing returns may be reached at a small value of MaxConcurrent, possibly even at 1 or 2. The optimum value requires experimentation.

NOTE: A setting of MaxConcurrent = 1 should be sufficient for all but the most demanding environments. Increase this value slowly and conservatively.

You might expect that databases that are licensed by number of connections would debit a single connection regardless of how many SQL statements are active. This is not necessarily the case; some databases count each open compiled SQL statement against the licensed number of connections. Another factor that determines how MaxConcurrent is set might be the database license.

%result Parameter

The **%result** parameter is a string value that can be returned as a column or stored procedure output parameter. The **%result** parameter can be used with or without password authentication.

The value expected to be returned in this parameter when authenticating a user can be specified in the SuccessResult entry of the [Settings] section. For example, if a user is successfully authenticated by the SQL authentication method, the result signifying success is the text string “okay”. This can be automatically checked by the following setting.

```
[Settings]
SuccessResult = okay
```

NOTE: The string comparison is case insensitive.

If the SQL statement succeeds but the SuccessResult value does not match the expected value returned from the database, Steel-Belted Radius Carrier issues a reject response, which can include any attributes and values configured in the [FailedSuccessResultAttributes] section of the *.aut file.

If PerformSuccessResultCheckAfterPasswordCheck=1 is specified and the SQL statement performs a password check that fails, Steel-Belted Radius Carrier does not process the SuccessResult and does not return the attributes from the [FailedSuccessResultAttributes] section in the reject response. If PerformSuccessResultCheckAfterPasswordCheck=1 is specified and the SQL statement performs a password check that succeeds but the SuccessResult value does not match the value returned from the

database, Steel-Belted Radius Carrier issues a reject response that contains the attributes from the [FailedSuccessResultAttributes] section.

In the following statement, **%password** is passed to a stored procedure, which returns a **%result** of either “okay” or something else (that signifies a rejection):

```
BEGIN CheckUser(%name, %password, %result!o); END;
```

Another example might be a database of usernames, passwords, and account status. The administrator can enable a user by setting the user’s account status to “okay” or disable the user by setting the account status to some other value, without having to delete the record. In the following statement, both **password** and **result** columns are checked:

```
SELECT password, result FROM database WHERE username = %name
```

[Results]

Password=1/50

%Result=2/50

Profile=0

Alias=0

SQL Authentication and Password Format

Steel-Belted Radius Carrier supports the authentication of users residing in a SQL database, in which password values for the users are stored in one of the following formats: clear-text, UNIXcrypt, Secured Hash Algorithm (SHA1+Base64 hash), MD4 hash, or enc-md5 reversibly-encoded password.

Hashed Passwords

Values in the Password column include a prefix that indicates how the password has been processed. The prefix is in clear-text between curly braces { } and is immediately followed by a hash value computed from the password. If no prefix is present in the value retrieved from the table **Password** column, the entire password is assumed to be in clear-text format. In summary:

- **PasswordText** indicates clear-text format (no encryption)
- **{crypt}HashHash** indicates UNIXcrypt format
- **{SHA}HashHashHash** indicates SHA1+Base64 hash
- **{SSHA}HashHashHashSalt** indicates salted SHA1+Base64 hash
- **{md4} HashHash** indicates MD4 hash of the Unicode form of password
- **{enc-md5}EncryptedEncrypted** indicates a reversibly encrypted password

NOTE: Refer to RFC 2759 for details about how MS-CHAP v2 produces an MD4 hash value.

NOTE: Although Steel-Belted Radius Carrier reads passwords encoded in enc-md5 format, you must purchase the Software Developer's Kit to convert clear-text passwords to this format.

UNIXcrypt is the standard hash algorithm that is used for the `/etc/passwd` file on Solaris systems. This may be necessary if, for example, the standard user database on a Solaris machine (the `/etc/passwd` file) is migrated to a SQL database, so that the values in the **Password** column of the SQL table are processed with UNIXcrypt.

Steel-Belted Radius Carrier may be configured to expect that the values retrieved from the SQL table **Password** column during authentication have been run through UNIXcrypt by adding the following entry into the [Settings] section of the SQL authentication configuration file:

```
PasswordFormat=3
```

Automatic Parsing

If **PasswordFormat** is set to 0, Steel-Belted Radius Carrier attempts to determine the password format automatically by parsing it. This is the recommended setting. Automatic parsing expects the password to be stored in one of the formats described in this section.

NOTE: The setting for automatic password parsing in older versions of Steel-Belted Radius Carrier (**auto**) has been deprecated.

Working with Stored Procedures in Oracle

The following notes discuss some considerations specific to Oracle, which uses the term *package* and *package body* when referring to stored procedures.

Assume you have a **SELECT** statement that extracts a user's name, password, and profile from the table **usertable** when it receives the user's name as an input parameter:

```
SELECT fullname, password, profile FROM usertable WHERE username = %name/63s
```

To write a package called **myPack1** that performs the equivalent function, you would enter the following sequence of commands:

```
Package myPack1
  is
  (
    PROCEDURE myProc
    name IN VARCHAR2,
    pass OUT VARCHAR2,
    prof OUT VARCHAR2,
    fName OUT VARCHAR2
  );
End myPack1;
```

When referencing the package from **radsql.aut**, you would point to the package name **myPack1** (not the procedure name **myProc**):

```
Package Body myPack1
  is
  (
    PROCEDURE myProc
    name IN VARCHAR2,
    pass OUT VARCHAR2,
    prof OUT VARCHAR2,
    fName OUT VARCHAR2
  )

  IS

  BEGIN

    SELECT fullname INTO fName, password INTO pass, profile INTO prof FROM usertable WHERE username =
      name;
    END myProc;
  End myPack1;
```

When you invoke the stored procedure, you would delineate each parameter as an input (!i), output (!o), or input/output (!io) variable. The presence of a !io or !o keyword indicates that the value returned from the database is to be included in the Access-Accept response as if it had been coded in the [Results] section. If a **r** value is included in the suffix (for example, !ir, !r, or !or), the parameter is expected to be an output parameter, and the attribute is to be treated as if it were included in the [FailedSuccessResultAttributes] section. Variables that are not specifically marked are considered input parameters by default.

NOTE: Do not configure a stored procedure to call the same attribute more than once. Doing so may cause Steel-Belted Radius Carrier to fail.

Correct: **SQL= {call joeproc2 (@class!o)}**

Incorrect: **SQL= {call joeproc2 (@class!o, @class!o)}**

You can replace the **SELECT** statement by invoking **myProc** as follows:

```
SQL=BEGIN myPack1.myProc(%name!i, %password!o, %profile!o, %fullname!o); END;
```

When using input-output parameters with Oracle, you must set the **DefaultResults** setting to 0. Any other variables that need to be returned (such as **Reply-Message**) must be identified by the “!o” marker within the SQL statement.

Working with Stored Procedures in MS-SQL

A simple example of a stored procedure returns a result set in the same way as a **Select** statement. For example, assume you have a table with the following fields: **username**, **password**, **Alias**, and **active**, where all fields have the datatype **varchar**. You want a stored procedure that will return a password and alias when the username and password received in the request match entries in the database, provided that **active** field has a value of yes.

Example 1

To create a simple stored procedure, run the following command sequence from MS Query Analyzer to create a stored procedure called **rsp_getpword**.

```
CREATE PROCEDURE rsp_getpword
@Uname varchar(21),
@pword varchar(21)

AS
SELECT password, alias FROM authentication WHERE username = @Uname
AND password = @pword AND active = 'yes'
GO
```

This stored procedure can then be executed from a ***.aut** file as follows:

```
SQL= Execute rsp_getpword %username, %password
```

```
[results]
```

```
Password=1
```

```
Alias=2
```

Example 2

More complex stored procedures take input and output parameters in a manner similar to that used by Oracle. For example, assume you have a table with the following fields: **username**, **password**, **profile**, and **active**, where all fields have a datatype of **varchar**. You want a stored procedure that returns a password and profile when the username and password received in the request match a username and password in the database, provided that the active field has a value of **yes**.

First, to create the stored procedure, run the following command from MS query analyzer:

```
CREATE PROCEDURE rsp_authuser
@uname as varchar(20),
@pword as varchar(21) OUTPUT,
@profile as varchar(21)OUTPUT
AS
SELECT @pword = password,@profile = profile FROM authentication WHERE username = @uname AND active
= 'yes'
GO
```

This stored procedure can then be executed from a *.aut file as follows:

```
SQL= {call rsp_authuser (%username!i, %password!o, %profile!o)}
[results]
; No entries should be specified in results, everything but the header should be commented out.
```

Tips on Using SQL Stored Procedures

Calling Stored Procedures

There are two common methods for calling a stored procedure in Steel-Belted Radius Carrier *.aut configuration files:

- The Execute statement. For example:

```
SQL= Execute rsp_getpword %username, %password
```

- A Call statement (Oracle style). For example:

```
SQL= {call rsp_authuser (%username!i, %password!o, %profile!o)}
```

TIP: You might get this error message in the Steel-Belted Radius Carrier log files:

```
diag 1: state = 24000, error = 0, [Microsoft][ODBC SQL Server Driver]Invalid cursor state
```

If you do, use the Call statement method because it has better cursor handling.

Using the Insert Function

When the INSERT function is used in a stored procedure, it must be encapsulated with the NOCOUNT function. This prevents the SQL database from returning a row count with the results.

Example:

```
SET NOCOUNT ON
IF (@Fail = 0) BEGIN
INSERT INTO AccessList (MemberID, MemberPassword, MemberLabel, MemberBill, AccessDate)
VALUES (@InID, @InPass, 'test', 'test', Getdate())
END ELSE BEGIN
INSERT INTO AccessFailures (FailedUser, FailedPassword, AttemptTime, FailureReason)
VALUES (@InID, @InPass, Getdate(), 'test')
END
SET NOCOUNT OFF
GO
```

Configuring SQL Accounting

IN THIS CHAPTER

- SQL Accounting Overview | 458
- Configuring SQL Accounting | 461
- Connecting to the SQL Database | 462
- SQL Statement Construction | 463
- SQL Accounting Return Values | 468
- Accounting Stored Procedure Example | 468

This chapter presents an overview of SQL accounting and describes how to configure SQL accounting in Steel-Belted Radius Carrier. This chapter contains these topics:

NOTE: Throughout this chapter, the term *attributes* refers to both standard RADIUS attributes and structured attributes. For information about specifying structured attributes, see the *SBR Carrier Reference Guide*.

NOTE: This SQL accounting information is unrelated to the SQL (MySQL) server used by Steel-Belted Radius Carrier.

SQL Accounting Overview

Steel-Belted Radius Carrier can write RADIUS accounting information to an external SQL database, independently of the Steel-Belted Radius Carrier accounting log.

To set up an external database for use as a repository for RADIUS accounting data, you must place an **.acc** database configuration file in the same directory that contains the Steel-Belted Radius Carrier daemon. This file must be modified to contain specialized information about your enterprise database.

Steel-Belted Radius Carrier offers the SQL accounting feature as a plug-in software module. Key features of the SQL plug-in include:

- The SQL statement is completely user-specified, allowing support of existing tables with existing field names and formats.
- The SQL statement can include a wide variety of arithmetic and string expressions.
- The SQL statement is parameterized, so it is compiled once, and each execution uses variable data without need for recompilation.
- Attribute and other data from the accounting request can be mapped to any parameter of the SQL statement (and hence to any field in the table) by means of a simple syntax.
- Different request types can be mapped to different SQL statements that may operate against distinct tables within the database.
- Multiple instances of a SQL statement can be overlapped for simultaneous execution.
- Multiple instances of the SQL accounting module can operate simultaneously, allowing logging to multiple databases.
- If the database connection drops, it is automatically reestablished after a configurable timeout without restarting Steel-Belted Radius Carrier.
- SQL accounting responses can return information.
- Stored procedures invoked by SQL accounting can make use of input parameters, record results, and return output parameters.

NOTE: While Steel-Belted Radius Carrier tries to provide uniformity in the operation of databases from different vendors, differences exist, particularly in the way SQL statements are interpreted. The capabilities of the SQL Authentication module depend on the capabilities of the underlying databases and their clients; things that work with one database may not work with another.

Stored Procedures

A stored procedure is a sequence of SQL statements that form a logical unit and perform a particular task. You can use stored procedures to encapsulate a set of queries or operations that can be executed repeatedly on a database server. For example, you can code operations on an employee database, such as password lookup, as stored procedures that can be executed by application code.

Stored procedures can be compiled and executed with different parameters and results. Stored procedures can use any combination of input parameters (the values passed to the stored procedure at execution

time) and output parameters (the values set or returned by the stored procedure to the calling application or environment).

You can write stored procedures for SQL that communicate with Steel-Belted Radius Carrier via input and output parameters to implement custom functions. Stored procedures let you use server-side processing on the SQL server to manipulate the information specified by variables. How you use these stored procedures depends on details specific to the implementation of SQL that you are using.

For information about using stored procedures with the Oracle SQL database, see [“Working with Stored Procedures in Oracle” on page 453](#). For information about using stored procedures with the Microsoft SQL database, see [“Working with Stored Procedures in MS-SQL” on page 455](#).

Connectivity Issues

Steel-Belted Radius Carrier may encounter serious problems if the connection between Oracle and Steel-Belted Radius Carrier becomes unstable. The most common reasons for a connection becoming unstable are:

- Slow or unreliable network response times
- Interruptions in connectivity caused by intervening network devices, such as a firewall timing out the connection

To prevent connectivity problems, consider implementation of one of the following solutions:

- To minimize problems caused by intervening firewalls, configure your firewall to pass traffic on the Oracle communications ports between the Steel-Belted Radius Carrier server and the Oracle server without restriction.
- To minimize network latency and firewall-related problems, move the Steel-Belted Radius Carrier server to the same network segment as the Oracle server.
- If moving your Steel-Belted Radius Carrier server is not feasible, locate a second Steel-Belted Radius Carrier server on the same network segment as your Oracle server, and configure your current Steel-Belted Radius Carrier server to proxy all authentication requests to this new device. This configuration will allow you to open RADIUS ports on the firewall only for the Steel-Belted Radius Carrier server (instead of opening RADIUS ports for all network access servers). Because proxy functions in Steel-Belted Radius Carrier do not require an uninterrupted connection to process requests, this solution allows you to retain your current firewall timeout settings.

Configuring SQL Accounting

You must configure both Steel-Belted Radius Carrier and the SQL database to support SQL accounting. The configuration procedure must be tailored to the database that you use. However, all procedures must give the following results:

- The SQL server must be configured to be listening for client requests. For SQL purposes, the Steel-Belted Radius Carrier server must be a client of the SQL server.
- The Steel-Belted Radius Carrier server must know the machine where the SQL server software runs, and it must know the protocol and port used in communicating with that machine.
- The required transport must be in place between SQL client and server.

Files

The files listed in [Table 50 on page 461](#) establish settings for configuring SQL accounting in Steel-Belted Radius Carrier. For more information about these files, refer to the *SBR Carrier Reference Guide*.

Table 50: SQL Accounting Files

File Name	Function
radsql.acc	Configures settings for SQL accounting (Oracle)
radsqljdbc.acc	Configures settings for SQL accounting (JDBC).

Using the SQL Accounting Configuration File

To configure SQL accounting, you must edit the accounting configuration file (**radsql.acc** (Solaris and Oracle) or **radsqljdbc.acc** (Solaris and JDBC)), located in the same directory that contains the Steel-Belted Radius Carrier daemon.

You must modify certain options in the accounting configuration file to accommodate your own database. After you update your accounting configuration file and restart Steel-Belted Radius Carrier, accounting proceeds as you have configured it.

Using Multiple SQL Databases

You can configure Steel-Belted Radius Carrier to log accounting transactions against more than one SQL database.

To add an additional database, create a new configuration file with extension **.acc** in the same directory as **radsql.acc** (Solaris and Oracle) or **radsqljdbc.acc** (Solaris and JDBC). You can give this file any name you

like, provided its extension is **.acc**. At startup, Steel-Belted Radius Carrier enumerates all **.acc** files to create its list of accounting modules.

NOTE: When creating the new file, start by duplicating the original **.acc** file, then make whatever modifications are necessary.

Connecting to the SQL Database

Upon startup, the SQL accounting module connects to the database, based on a connect string specified in your accounting configuration file. The connect string contains information such as the name and location of the database, and the password required to connect. The connect string is passed to the database client to establish the connection.

While a sample connect string is provided in the original configuration file, you must configure the Connect entry of the configuration file with a connect string appropriate to your database.

The password for database access must be provided as part of the connect string or the connection fails.

If the initial attempt to connect to the database fails, or if a processing error occurs that the SQL accounting module interprets as a database connection failure, the SQL accounting module drops the connection and attempts to establish a new connection after a period of time. In the interim, all authentication requests are ignored.

The SQL accounting module uses an exponential back-off strategy in determining how long to wait before attempting a new connection, as well as how frequently this attempt should be made. After the first dropped connection, it waits a certain amount of time before attempting to reconnect. If this attempt to reconnect also fails, it waits for twice the amount of time before trying again; and so on, up to some maximum wait time. The initial and maximum wait times are configurable.

NOTE: Detailed error information may not be available if there is an error processing the database logon at connect time. A numeric result code appears in the log. You may need to refer to product-specific documentation to decode this result code. With Oracle on Solaris, you can use the **oerr facility-code error-number** command with a facility code of ora from the command shell.

SQL Statement Construction

For each accounting request whose Acct-Status-Type is mapped to a SQL statement, that accounting request is logged to the back-end database by executing the associated SQL statement.

While a sample SQL statement is provided in the original configuration file, you must configure one or more SQL entries of the configuration file with a statement appropriate to your database. Each SQL statement is typically an **INSERT INTO** statement and may contain additional syntax elements that are preprocessed by the SQL accounting module.

The SQL accounting module executes SQL statements in parameterized form. This means that the SQL statement is compiled once, with parameter markers (usually question marks) as placeholders for data items that vary from one execution to the next. Only upon execution of the statement are the actual data values supplied.

The SQL statement you compose must not include parameter markers directly. Instead, the names of the parameters should be included where parameter markers would appear, in a format described below. The SQL authentication module translates the SQL statement provided, replacing parameter names with parameter markers before passing the SQL statement to the database engine.

A SQL statement can be very simple. Basically, all that is required is to set fields of the database record with values from the request. The SQL statement can also be quite complex; it can include inner joins, and it can contain expressions. The underlying database engine is responsible for handling the SQL statement; The SQL accounting module performs no interpretation of the SQL statement other than to translate parameter names to parameter markers.

INSERT Statement and VALUES Section

The following example shows a SQL **INSERT** statement that might be found in a Steel-Belted Radius Carrier .acc file:

```
INSERT INTO usagelog (Time, NASAddress, SessionID, Type, Name, BytesIn, BytesOut) VALUES
  (%TransactionTime/t, %NASAddress, @Acct-Session-Id, @Acct-Status-Type, %FullName/40s,
  @Acct-Input-Octets, @Acct-Output-Octets)
```

In the **VALUES** section, the names (between parentheses) represent the values inserted into the SQL table columns. To support the SQL accounting module, each item in the VALUES section must be prefixed with a @ sign or a % sign.

- @ indicates a RADIUS accounting attribute. The attribute name must also be listed in the **account.ini** file. This remains true even if the **account.ini** file is disabled.
- % indicates an item associated with the **INSERT** request that is not a RADIUS accounting attribute. [Table 51 on page 464](#) lists the Steel-Belted Radius Carrier items that may be provided.

Table 51: Insert Statement Syntax

Item	Data Type	Meaning
%TransactionTime	Time	<p>The date/time that the event occurred that is the subject of the request.</p> <p>NOTE: You should include the <code>/t</code> (timestamp) data type qualifier with the %TransactionTime argument in SQL statements. If you do not, the %TransactionTime output is formatted as character, with differing results on JDBC and Oracle.</p>
%Time	Time	<p>The date/time when the request is being processed. (This is later than %TransactionTime if the request is a retry.)</p> <p>NOTE: You should include the <code>/t</code> (timestamp) data type qualifier with the %Time argument in SQL statements. If you do not, the %Time output is formatted as character, with differing results on JDBC and Oracle.</p>
%Type	String	The RADIUS accounting request type.
%NASAddress	IP address	The IP address of the requesting NAS.
%NASName	String	The name of the network access server that originated the request. This may be the name of the RADIUS client entry in the database or the value of the NAS-Identifier or NAS-IP-Address attribute.
%NASModel	String	The NAS make/model.
%FullName	String	The full name of the logged in user.
%AuthType	String	The method by which the user was authenticated.
%RADIUSClientName	String	The name of the network access server, as specified in a RADIUS client entry in the Steel-Belted Radius Carrier database.

A format specifier may appear immediately following each parameter. The format specifier should describe the database storage format of the column that corresponds to the parameter. It consists of a slash (/), possibly a length, and a data type. [Table 52 on page 465](#) lists the available data types.

Table 52: Data Types

Format Specifier	Meaning
/xs	A text string of length x. /s indicates a string with the default length of 256.
/xb	A binary data string of length x. A binary string is different from a text string in that it is not NULL-terminated and is not restricted to ASCII characters. /b indicates a binary data string with the default length of 256.
/n	32-bit integer
/n8	8-bit integer
/n16	16-bit integer
/n32	32-bit integer (same as /n)
/nxx	Integer xx bits in length. For example, /n64 indicates a number with a length of 64 bits.
/t	Timestamp

NOTE: Steel-Belted Radius Carrier supports integers larger than 32 bits by manipulating them as binary data strings. The Solaris Oracle 8 plug-ins are able to convert binary data strings between Oracle VARRAW types (/xb) and Oracle NUMBER types (/n). Oracle types must be declared with enough precision to avoid truncation when inserting into the database, and care must also be taken to avoid truncation when retrieving from the database. In particular, avoid retrieving Oracle VARRAW types larger than 256 bytes. Other database/operating-system combinations may not allow for integers larger than 32 bits.

If a format specifier is not present in the SQL statement syntax, Steel-Belted Radius Carrier automatically defaults to an appropriate specifier based on the actual parameter type. For example, @Acct-Input-Octets is a number, and defaults to /n.

NOTE: For strings, always include a format specifier, and be sure to specify a length no greater than the actual field size in the database. The compilation of the SQL statement may fail if a length greater than the actual field size is specified. If no format specifier is present, the length defaults to 256 characters, which may cause the compilation to fail.

Steel-Belted Radius Carrier automatically attempts to convert between the internal format of a parameter and its format in the database, as described by the format specifier. In most cases, the formats are equivalent; if not, Steel-Belted Radius Carrier performs reasonable conversions.

Table 53 on page 466 lists the internal formats and their compatible database formats:

Table 53: Internal Formats and Compatible Database Formats

Internal Format	Compatible Database Formats
Binary data string	/b, /xb, /n, /n8, /n16, /n32
Number	/n, /n8, /n16, /n32, /xs, /s
String	/xs, /s
Time (seconds since 1/1/70)	/t, /n, /n32, /xs, /s
IP address	/n, /n32, /xs, /s

As you write the INSERT statement for your SQL accounting configuration file (.acc), we recommend the following syntax check list:

- The column names and their corresponding attributes in the VALUES section are order-dependent. In the example, the %TransactionTime/t value would be inserted into the Time column (and formatted as a timestamp), the %NASAddress value would be inserted into the NASAddress column, and so forth. The ordering of these settings is critical to proper RADIUS accounting data insertion, since each column in the SQL table may be a specific data type, such as **varchar** or **int**.
- The use of left and right parentheses (), the backslash \, the forward slash / and even blank spaces are all extremely important and must be exact. You can add as many columns and attributes as you want for your RADIUS accounting needs; however, be sure to model your **INSERT** statement syntax as shown in example.
- An attribute listed incorrectly in the VALUES section, such as **@Acct_Session-Id** rather than **@Acct-Session-Id**, causes the SQL statement to fail during a RADIUS accounting transaction. The attribute's syntax must match its corresponding attribute name in the **account.ini** file, which in turn

matches the attribute's name in the appropriate dictionary file, which allows Steel-Belted Radius Carrier to process the attribute correctly when it is received from the NAS (the RADIUS client).

- An attribute listed in the **VALUES** section that is missing its prefix of @ or % causes the SQL statement to fail during a RADIUS accounting transaction.
- If a carriage return is present within the **INSERT** statement without the backslash \ to indicate the end of the line, the SQL statement fails during a RADIUS accounting transaction.
- Do not make the lines in the .acc file too long. There is a line length limit of 255 characters. Use the backslash \ to indicate the end of the line before that limit is reached. If a line exceeds this limit, the SQL statement fails during a RADIUS accounting transaction.

Using Multiple SQL Statements

The most common use of accounting is to track user sessions. However, accounting requests are generated when the NAS starts up and shuts down; and, vendor-specific uses of accounting are used to track other NAS phenomena. Clearly, it might be advisable to log different types of accounting events to different tables.

The Acct-Status-Type attribute of an accounting request indicates the request type. You may, if you like, create multiple SQL statements, and map each Acct-Status-Type to one of these SQL statements. The different statements may update different tables in the database, but they all share the single database connection.

Overlapped Execution of SQL Statements

The SQL accounting module is multi-threaded. SQL accounting can be configured with a maximum number of simultaneous executions of any SQL statement, using the **MaxConcurrent** entry in the .acc file's [Settings] section.

NOTE: The MaxConcurrent entry is valid only for Oracle SQL accounting (**radsql.acc**). It is not valid for JDBC SQL accounting (**radsqljdbc.acc**).

If **MaxConcurrent** is set to 1, SQL execution occurs serially, and the SQL execution for each accounting request must complete before execution for the next request may begin.

By increasing **MaxConcurrent**, it may be possible to increase throughput by overlapping operations, especially if the database server is remote and a large part of the time to complete a statement execution is taken up by network latency. If the database server is local, the point of diminishing returns may be reached at a small value of **MaxConcurrent**, possibly even at 1 or 2. You can find the optimum value for your system by experimentation.

NOTE: A setting of **MaxConcurrent** = 1 should be sufficient for all but the most demanding environments. Increase this value slowly and conservatively.

MaxConcurrent determines the maximum overlap for executing any single SQL statement. Multiple SQL statements for different request types are not interdependent, and executions of one statement do not affect executions of a different statement.

You might expect that databases that are licensed by number of connections would debit a single connection regardless of how many SQL statements are active. This is not necessarily the case; some databases count each open compiled SQL statement against the licensed number of connections. The database license may also have an influence on the optimum setting for **MaxConcurrent**.

SQL Accounting Return Values

SQL accounting statements can return information in RADIUS attributes in an accounting response. This is useful only if you are using a client that expects and supports attributes embedded in a RADIUS accounting response message.

Stored procedures can also return output parameters. To call an Oracle stored procedure in a Solaris environment:

```
BEGIN storedProcedure(parameters ...); END;
```

Accounting Stored Procedure Example

A simple stored procedure can return a result set in the same way as a **Select** statement. For example, assume you have a table with the following fields: **username**, **password**, **Alias**, and **active**, where all fields have the datatype **varchar**. You want a stored procedure that will return a password and alias when the username and password received in the request match entries in the database, provided that **active** field has a value of Yes.

The following example executes a stored procedure to update an accounting table in Steel-Belted Radius Carrier.

1. Create an accounting table by executing the following command:

```
create table accounting
```

```
(TransactionDate varchar(20), Username varchar(21), SessionID varchar(12), NASIPAddr varchar(15), NASPort
  varchar(5), UserIPAddr varchar(15), CallingNum varchar(12), CalledNum varchar(12),
  type varchar(4), Sessiontime varchar(14), Disconnect varchar(12))
```

2. Create an **rsp_account** stored procedure that can be called by a *.acc file.

```
create procedure rsp_account

@transactiontime varchar(21),
@username varchar(21),
@AcctSessionID varchar(21),
@NASIPAddress varchar(21),
@NASPORTTYPE varchar(21),
@FRAMEDIPADDRESS varchar(21),
@callingstationid varchar(21),
@calledstationid varchar(21),
@TYPE varchar(21),
@ACCTSESSIONTIME varchar(21),
@ACCTTERMINATIONCAUSE varchar(21)
AS
INSERT INTO Accounting (TransactionDate, username, SessionID, NASIPAddr, NASPort, UserIPAddr,
  CallingNum, CalledNum, type, Sessiontime, Disconnect)
VALUES (@transactiontime, @username, @AcctSessionID, @NASIPAddress, @NASPORTTYPE,
  @FRAMEDIPADDRESS, @callingstationid, @calledstationid, @TYPE, @ACCTSESSIONTIME,
  @ACCTTERMINATIONCAUSE)
```

3. Create the **mysqlacct.acc** file to call the **rsp_account** stored procedure.

The **mysqlacct.acc** file uses an **SQL=EXECUTE *procedure_name value1,...valueN*** statement instead of an **SQL=INSERT into table (*column1, ...columnN*) Values (*value1,...valueN*)**, because the stored procedure performs the INSERT action.

Be sure to configure the CONNECT statement to reflect your operating environment.

```
[Bootstrap]
LibraryName=sqlacct.dll
Enable=1
InitializationString=

[Settings]
Connect=DSN=<dsn_name_here>;UID=<username_for_db>;PWD=<password_for_db>
ConnectTimeout=25
WaitReconnect=2
```

MaxWaitReconnect=360

ParameterMarker=?

loglevel=2

[Type]

1=User

2=User

3=User

[Type/User]

SQL=Execute rsp_account %transactiontime/t, \

@user-name/21s, \

@Acct-Session-ID/12s, \

@NAS-IP-Address/15s, \

@NAS-PORT-TYPE/5s, \

@FRAMED-IP-ADDRESS/15s, \

@calling-station-id/12s, \

@called-station-id/12s, \

%TYPE/4s, \

@ACCT-SESSION-TIME/14s, \

@ACCT-TERMINATION-CAUSE/12s

ConcurrentTimeout=30

MaxConcurrent=2

Configuring LDAP Authentication

IN THIS CHAPTER

- [LDAP Authentication Overview | 471](#)
- [Configuring LDAP Authentication | 474](#)
- [LDAP Authentication Sequence | 479](#)
- [LDAP Authentication Examples | 480](#)

This chapter presents an overview of LDAP authentication and describes how to configure LDAP authentication in Steel-Belted Radius Carrier. This chapter contains these topics:

NOTE: Throughout this chapter, the term *attributes* refers to both standard RADIUS attributes and structured attributes. For information about specifying structured attributes, see the *SBR Carrier Reference Guide*.

LDAP Authentication Overview

Steel-Belted Radius Carrier can authenticate against records stored in an external LDAP database. Any RADIUS attribute, such as username and password, can be used to query the database.

External database authentication is typically used when an organization has a large amount of user information stored in an LDAP database, and wants to authenticate these users using RADIUS. Authentication against an existing LDAP database extends authentication services to user accounts without requiring an administrator to enter user information into the Steel-Belted Radius Carrier database.

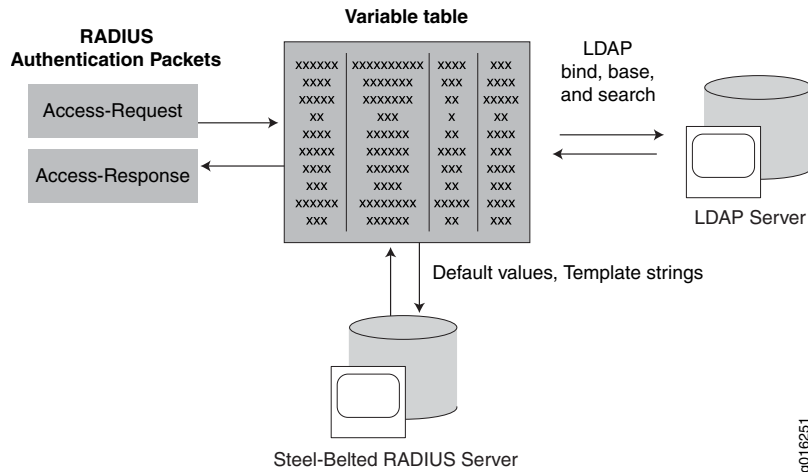
Steel-Belted Radius Carrier offers LDAP authentication as a plug-in software module. Key features of the LDAP plug-in include the following:

- Support for LDAP Version 3.
- Linux SBR Carrier (starting from Release 7.4.0) and Solaris SBR Carrier (starting from Release 7.5.0) use OpenLDAP libraries to process LDAP requests.
- In case of SSL, for SBR Carrier to process LDAP requests, you have to configure OpenLDAP to accept the server certificate.
- You can authenticate via LDAP Bind or via a password returned from an LDAP Search request (BindName).
- A single Search request or a sequence of Search requests can be specified.
- Bind, Base, and Search strings can include variables.
- New Bind parameters can be specified during a sequence of searches.
- Other authentication credentials can be specified in a string that can include variable values.
- Variables may be set from the RADIUS request packet and from LDAP Search results.
- Variables may be used to specify RADIUS response attributes and other response information.
- The RADIUS response can include RADIUS attributes found in the LDAP database, or it can reference a Steel-Belted Radius Carrier profile or user entry.
- Several features similar to SQL authentication are supported, such as round-robin load balancing and activation targets.
- Decorated usernames can be parsed into two variables within the variable table. For example, *simon@xyz.com* would be parsed into *simon* and *xyz.com* for use later in the authentication process.
- The variable table allows both attributes and %Profile in the [Response] section.
- Conditional search logic is supported by branching using the OnFound and OnNotFound fields.

LDAP Variable Table

The LDAP Variable Table lets you translate a RADIUS request into an LDAP lookup. At the beginning of each LDAP authentication request, Steel-Belted Radius Carrier creates a Variable Table. Attributes and other information from the RADIUS request are entered in the Variable Table for use in LDAP Bind, Base, and Search strings. When attributes are returned by LDAP requests, they too are entered in the Variable Table. Finally, selected information from the Variable Table is returned to the RADIUS client in the RADIUS response packet. See [Figure 190 on page 473](#).

Figure 190: Role of the Variable Table in LDAP Authentication



Types of LDAP Authentication

To design an LDAP authentication method, consider how you want to validate the username and password.

The LDAP plug-in offers two techniques for validating the username and password. Each configuration file that you write to control LDAP authentication must employ **Bind** or **BindName**. The differences between the two techniques have to do with how Steel-Belted Radius Carrier connects to the LDAP server and whether the username/password validation is performed by the LDAP server or by Steel-Belted Radius Carrier.

BindName Authentication

When you use **BindName** authentication, your LDAP configuration file provides Steel-Belted Radius Carrier with the username and password of an account on the LDAP server. This must be an account that has privileges to access all of the information that you require to authenticate users. In the LDAP configuration file, you provide the username in the **BindName** parameter, and the password in the **BindPassword** parameter.

After you complete the LDAP configuration file, each time Steel-Belted Radius Carrier starts up, it executes a **Bind** request to the LDAP server using the **BindName** and **BindPassword** parameters as its credentials. If the LDAP server can validate these credentials, a connection is established between the two servers. This connection remains *up* all the time. It is disconnected only if the Steel-Belted Radius Carrier server or the LDAP server goes down, and it is re-established as soon as possible after the *down* server comes back up. The LDAP configuration file offers a number of connection and re-connection timeouts and other parameters that regulate this relationship.

Any time authentication via LDAP is required, Steel-Belted Radius Carrier consults the corresponding LDAP configuration file. When you use **BindName** authentication, this file must contain a **Search** command that maps the username from the Access-Request to a password attribute in the LDAP database. The Search may retrieve other LDAP attributes as well. When the Search returns its results, Steel-Belted Radius

Carrier compares the value of the password returned from the LDAP database with the password from the incoming Access-Request. If the two values are the same, the password is considered validated.

When the connection to the LDAP server is established using BindName, multiple authentications can be performed at the same time over the same connection. This is done using the **MaxConcurrent** setting in the [Settings] section of the **ldapauth.aut** file.

Bind Authentication

When you use Bind authentication, Steel-Belted Radius Carrier authenticates connection requests by attempting to Bind to the LDAP server using the username and password from the incoming Access-Request or from a configured username and password. If this Bind request succeeds, the password is validated. This is essentially *pass-through* authentication; Steel-Belted Radius Carrier presents an LDAP user's credentials to the LDAP server and asks to have them validated.

In the simplest case, a single connection is established for each Access-Request and is kept open only long enough for the LDAP server to validate the password and respond to any Search requests. Then Steel-Belted Radius Carrier closes the connection and completes any processing that remains to generate an Access-Response.

A more sophisticated search technique can take advantage of flexible Bind, which allows you to allocate a sequence of connections for each Access-Request. Each in turn is kept open only long enough for the server to process each search criterion. Then Steel-Belted Radius Carrier closes the connection and completes any processing that remains to generate an Access-Response.

Attributes and LDAP Authentication

A username and password may be all the information that you require to authenticate users. However, the LDAP plug-in offers a number of techniques for working with check list or return list attributes, should you need them.

Configuring LDAP Authentication

To configure an LDAP authentication method, you must edit the **ldapauth.aut** configuration file, which controls the LDAP authentication sequence.

[Table 54 on page 475](#) summarizes the process of configuring an LDAP authentication method for Steel-Belted Radius Carrier. It lists the sections that you must edit in the **ldapauth.aut** configuration file to accomplish each step.

You must perform all steps.

You must at least consider the entries that you want to put in each section of the configuration file, even if you decide to leave most of that section blank.

Table 54: LDAP Authentication (ldapauth.aut) Configuration File Tasks

Step	LDAP Configuration Task	.aut File Sections
1	Decide how you want Steel-Belted Radius Carrier to validate RADIUS access requests. Two major areas of choice are described above: (1) Bind or BindName ; and (2) Profile, Alias, or attribute list.	All sections
2	Determine which incoming RADIUS attributes are required to perform the LDAP search.	[Response]
3	Determine which LDAP attributes support are required to perform the LDAP search.	[Attribute/name]
4	Design Search template(s) that can find the necessary data in your LDAP database schema.	[Search/name]
5	Extract the data from the incoming RADIUS packet that Steel-Belted Radius Carrier will use to perform the LDAP Bind and Search requests.	[Request]
6	Select defaults that you want Steel-Belted Radius Carrier to use when corresponding values are not provided.	[Defaults]
7	Enable connections between the Steel-Belted Radius Carrier server and LDAP server(s).	[Server] [Server/name] [Settings] [Failure]
8	Enable the LDAP plug-in and name the authentication method.	[Bootstrap]

The order to edit configuration file sections is the reverse of the order in which Steel-Belted Radius Carrier processes them. The processing sequence is described in [“LDAP Authentication Sequence” on page 479](#).

Supporting Secure Sockets Layer

To configure SSL to be supported by the LDAP plug-in:

1. Set SSL in the [Settings] (or [Server/name]) section to 1.
2. Set the port in the [Server] section to the SSL port of the LDAP server.

NOTE: If SSL=1, the Host parameter in [Server/*name*] accepts only LDAP-style URIs. For example, ldaps://hostname:port. The default port setting is ignored and the LDAP-style URIs for Host is applied.

3. Set OpenLDAP to accept the server certificate.

This is configured in the **ldap.conf** file located in the following path:

- For Linux: /etc/openldap/ldap.conf
- For Solaris: /usr/local/etc/openldap/ldap.conf

TLS_REQCERT never parameter setting allows SBR Carrier to accept the server certificate and the TLS connection, which allows the LDAP traffic to be encrypted.

NOTE: A minimum DH key size of 1024 bits is recommended for use of SSL with LDAP.

Files

The file in [Table 55 on page 476](#) establishes settings for LDAP authentication. For more information about this file, refer to the *SBR Carrier Reference Guide*.

Table 55: LDAP Authentication File

File Name	Function
ldapauth.aut	Specifies settings for LDAP authentication in Steel-Belted Radius Carrier. For complete details see the LDAP authentication file in the <i>SBR Carrier Reference Guide</i> .

LDAP Database Schema

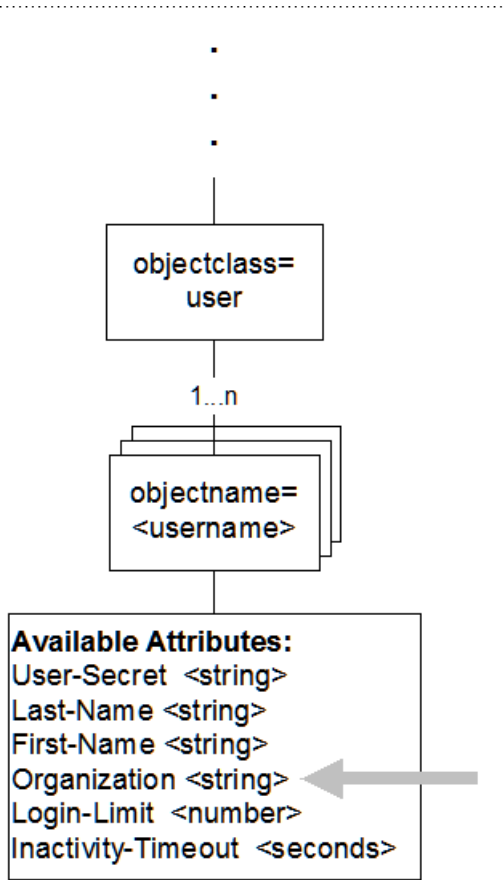
The most important factor in the success of your LDAP authentication methods is the design of your LDAP database schema. This discussion assumes that you already have a schema in place.

Often, you can use the LDAP plug-in *without* changing the LDAP database schema at all. In [Figure 191 on page 477](#), the user record already provides an LDAP attribute called Organization. If you intend to grant connection privileges according to the organization to which each user belongs, you can create profiles in the Steel-Belted Radius Carrier database whose names match the strings you are already using for the Organization attribute. You can then create an LDAP authentication configuration file that

retrieves the value of the Organization attribute from the LDAP database and returns it to Steel-Belted Radius Carrier as the name of the profile to use.

NOTE: If you are using BindName authentication, you need to be able to identify which LDAP attribute contains the user's password. In the schema in [Figure 191 on page 477](#), this attribute is called User-Secret.

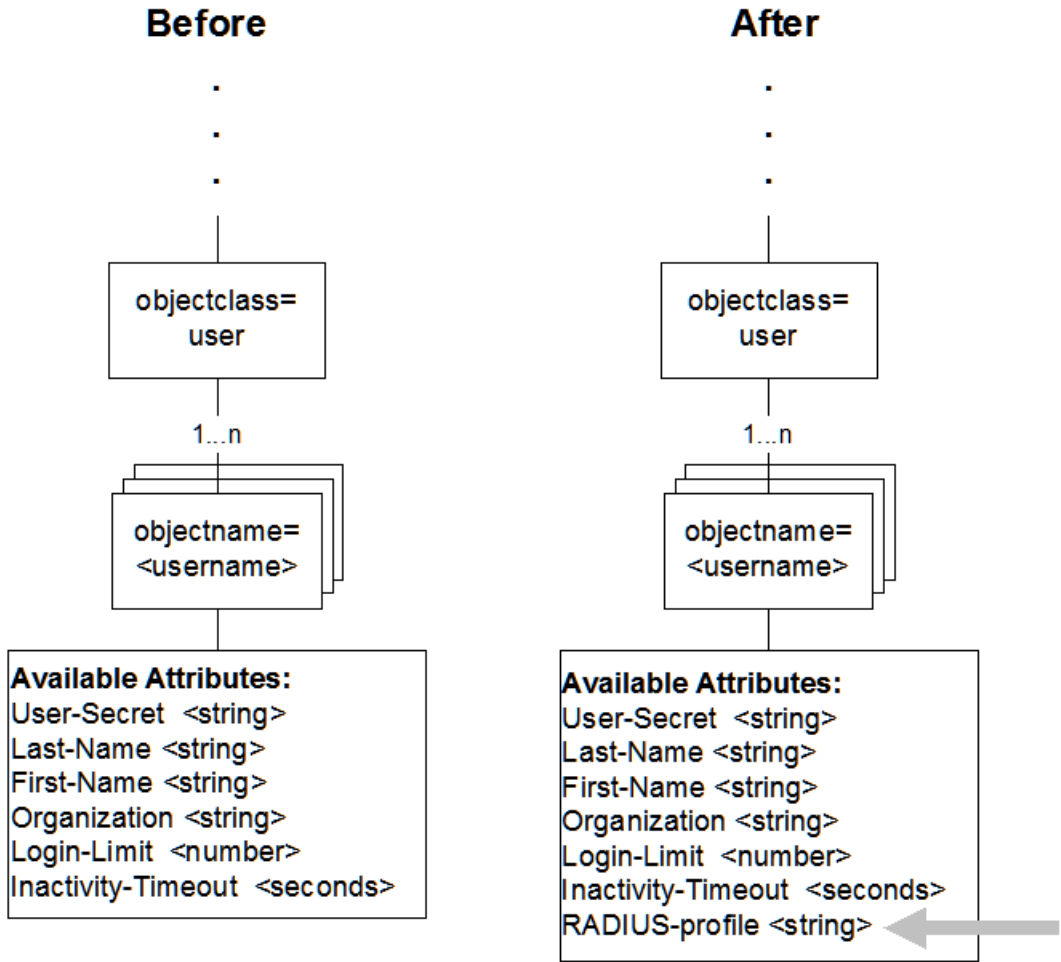
Figure 191: Capitalizing on an Existing Schema for LDAP Authentication



When the authentication strategy you have chosen requires data that is not currently in the schema, you might need to modify the schema.

The name of a Steel-Belted Radius Carrier profile is a typical example. Consider the example shown in [Figure 191 on page 477](#). If you want to assign connection privileges to users in some way other than by Organization, and no other LDAP attribute seems appropriate, you can add an LDAP attribute that names a profile. In [Figure 192 on page 478](#), this attribute is called RADIUS-Profile. This attribute contains a string value that can be set to the name of a profile defined in the Steel-Belted Radius Carrier database.

Figure 192: Modifying a Schema to Enhance LDAP Authentication



LDAP Authentication and Password Format

Steel-Belted Radius Carrier supports authentication of users whose records reside in an LDAP table in which password values are stored in one of the following formats: clear-text, UNIXcrypt, Secured Hash Algorithm (SHA1+Base64 hash), MD4 hash, or enc-md5 reversibly-encoded password.

Hashed Passwords

Encoded values include a prefix that indicates how the password has been processed. The prefix is in clear-text between curly braces {} and is immediately followed by a hash value computed from the password. If no prefix is present in the value retrieved, the entire password is assumed to be in clear-text format. In summary:

- **PasswordText** indicates clear-text format (no encryption)
- {crypt} **HashHash** indicates UNIXcrypt format
- {SHA} **HashHashHash** indicates SHA1+Base64 hash

- **{SSHA} HashHashHashSalt** indicates salted SHA1+Base64 hash
- **{md4} HashHash** indicates MD4 hash of the Unicode form of password
- **{enc-md5} EncryptedEncrypted** indicates a reversibly encrypted password. (Although Steel-Belted Radius Carrier reads passwords encoded in this format, you must purchase the Software Developer's Kit to convert clear-text passwords to this format.)

UNIXcrypt is the standard hash algorithm used for the **/etc/passwd** file on Solaris systems. This may be necessary if, for example, the standard user database on a Solaris machine (the **/etc/passwd** file) is migrated to a SQL database, so that the values in the **Password** column of the SQL table are processed with UNIXcrypt.

You can configure Steel-Belted Radius Carrier to expect that the values retrieved from a table have been run through UNIXcrypt by adding the following entry into the [Settings] section of the LDAP authentication configuration file:

```
PasswordFormat=3
```

Automatic Parsing

If **PasswordFormat** is set to **0**, Steel-Belted Radius Carrier attempts to determine the password format automatically by parsing it. This is the recommended setting. Automatic parsing expects the password to be stored in one of the formats described in this chapter.

This technique is useful if clear-text passwords are available to Steel-Belted Radius Carrier (that is, if PAP is used). If you set **PasswordFormat** to **0**, the stored password can be returned to Steel-Belted Radius Carrier still encrypted, and the comparison with the password received from the RADIUS client can be done on the Steel-Belted Radius Carrier side.

NOTE: The setting for automatic password parsing in previous versions of Steel-Belted Radius Carrier (auto) has been deprecated.

LDAP Authentication Sequence

The sequence of an LDAP authentication transaction is controlled by the LDAP authentication configuration file as follows:

1. The Variable Table is initialized to default values as specified in the [Defaults] section. All variables that are not listed in the [Defaults] section are initialized to null values.
2. The values of RADIUS attributes in the Access-Request are copied to the Variable Table, as specified in the [Request] section.
3. If a **Bind** entry was specified in the [Settings] section, authentication via LDAP **Bind** is now performed. The **Bind** entry is used as a template to construct a bind string, using replacement values from the Variable Table. An LDAP **Bind** is then performed to authenticate the user.
4. An LDAP Search request is performed for each [Search/*name*] section specified. You may specify zero or more separate Search requests.

For each Search request, LDAP Base and Filter strings are constructed from templates, using replacement values from the Variable Table. These Base and Filter strings are then transmitted to the LDAP server in a Search request.

Each attribute/value pair returned by the LDAP Search is used to set the value of the corresponding entry in the Variable Table. Also, the DN returned by the search may be used to set a variable.

5. If a %Password entry appears in the [Response] section, authentication is now performed. The password entered by the user is validated against the value that appears in the %Password variable, and the user is rejected if the passwords do not match.
6. If a %Profile entry appears in the [Response] section, the value of the %Profile variable is used to look up a Profile entry in the Steel-Belted Radius Carrier database. The check list and return list attributes in that Profile are used to validate the request and return an appropriate response.
7. If a %Alias entry appears in the [Response] section, the value of the %Alias variable is used to look up a Native User entry in the Steel-Belted Radius Carrier database. The current transaction is treated as if it came from the “alias” user; that is, the check list and return list attributes of the alias user are used to validate the request and return an appropriate response.
8. If neither a %Profile nor a %Alias entry appears in the [Response] section, then RADIUS attributes for the response packet are created from the Variable Table, based on attribute entries in the [Response] section.

LDAP Authentication Examples

This topic provides examples of LDAP authentication configuration file syntax. The examples illustrate how you might:

- Authenticate passwords (**Bind** or **BindName**).
- Specify check list and return list attributes (list the attributes or name a profile entry in the Steel-Belted Radius Carrier database).

Bind Authentication with Default Profile

The following example is a simple LDAP authentication configuration file. Every user is authenticated using a **Bind** request to the LDAP database. The same Steel-Belted Radius Carrier attribute profile is applied to every Access-Request.

```
[Settings]
MaxConcurrent=1
Timeout=20
ConnectTimeout=25
QueryTimeout=10
WaitReconnect=2
MaxWaitReconnect=360
Bind=uid=<User-Name>, ou=Special Users, o=bigco.com
LogLevel = 2
UpperCaseName = 0
PasswordCase=original
SSL = 0
[Server]
s1=
[Server/s1]
Host=199.185.162.147
Port = 389
[Defaults]
TheUserProfile = Sample
[Request]
%User-Name = User-Name
[Response]
%Profile = TheUserProfile
[Search/DoLdapSearch]
Base = ou=Special Users, o=bigco.com
Scope = 2
Filter = uid=<dialup>
Attributes = AttrList
Timeout = 20
%DN = dn
[Attributes/AttrList]
```

If the [Response] section is empty, Steel-Belted Radius Carrier passes the **Bind** results (accept or reject) directly to its client; no additional RADIUS attributes are returned in the Access-Response.

BindName Authentication with Callback Number Returned

In the following example, requests are authenticated using Search. **BindName** and **BindPassword** values are supplied to permit a connection to the LDAP database. Return list attributes for authentication are listed in the [Response] section. In this example, the network access server needs a callback number to complete the connection. The value of the incoming DNIS attribute Calling-Station-ID is used to ensure that the callback number is the number from which the user's request originated.

NOTE: This example is incomplete; it omits the [Bootstrap] and [Settings] sections to save space.

```
[Server]
s1=
[Server/s1]
Host = 67.186.4.3
Port = 389
BindName=uid=admin, ou=Administrators, ou=TopologyManagement, o=NetscapeRoot
BindPassword=ourlittlesecret
Search = DoLdapSearch
[Defaults]
SendThis = DidLDAPAuthSearch
[Request]
%UserName = dialup
Calling-Station-ID = thenumbertocall
[Search/DoLdapSearch]
Base = ou=Special Users, o=bigco.com
Scope = 2
Filter = uid=<dialup>
Attributes = AttrList
Timeout = 20
%DN = dn
[Attributes/AttrList]
dialuppassword
[Response]
%Password = dialuppassword
Reply-Message = SendThis
Ascend-Callback-No = thenumbertocall
```

LDAP Bind with Profile Based on Network Access Server

In the following example, requests are authenticated using **Bind**. Check list and return list attributes for authentication are provided by referencing a profile entry in the Steel-Belted Radius Carrier database. The

profile to be used depends on the specific network access server from which the user's request originates. Steel-Belted Radius Carrier retrieves the profile name by the LDAP database for an IP address that matches the address of the requesting NAS. If this search fails, a profile called limited is used. If a profile name is successfully retrieved from the LDAP database, but no profile by that name can be found in the Steel-Belted Radius Carrier database, authentication fails due to lack of resources and the user is rejected.

NOTE: This example is incomplete; it omits the [Bootstrap] section and many [Settings] entries to save space.

```
[Settings]
Bind=uid=<loginID>, ou=Special Users, o=bigco.com
Search = DoLdapSearch
[Server]
s1=
[Server/s1]
Host = 67.186.4.3
Port = 389
[Request]
%UserName = loginID
%NASAddress = deviceIP
[Defaults]
%Profile = limited
[Search/DoLdapSearch]
Base = ou=CommServers, o=bigco.com
Scope = 1
Filter = ipaddr=<deviceIP>
Attributes = AttrList
Timeout = 20
%DN = dn
[Attributes/AttrList]
profile
[Response]
%Profile = profile
```

Signalware SIGTRAN Gateway Support

IN THIS CHAPTER

- [Overview of Signalware SIGTRAN Protocol Stacks | 484](#)

Overview of Signalware SIGTRAN Protocol Stacks

Optionally, the Mavenir Signalware SIGTRAN protocol stack can be used to access Signaling System 7 (SS7) networks where the Home Location Register (HLR) database resides. Signalware acts as a link between the Steel-Belted Radius Carrier server and the SS7 network. ANSI SS7, CCITT/ITU SS7, Japanese, and Chinese networks are supported. Signalware provides SS7 communication over IP networks.

The Signalware SIGTRAN protocol stack is used in conjunction with the optional SIM Authentication module to provide gateway functionality so Steel-Belted Radius Carrier can pass Mobile Access Part (MAP) requests to the SS7 network. MAP requests are passed over the SS7 network to the Home Location Register (HLR), which is the primary subscriber database in the Global System for the Mobile Communications (GSM) network. The HLR then performs a database lookup and returns the requested authentication or authorization information to Steel-Belted Radius Carrier.

To process MAP requests to an HLR, you must configure the optional SIM authentication module. See [“SIM Authentication Module” on page 536](#).

NOTE: SBR Carrier does not support the Signalware communication stack on Solaris.

For information about installing and configuring the Signalware SIGTRAN protocol stacks, see the *SBR Carrier Installation Guide*.

Proxy RADIUS Authentication and Accounting

IN THIS CHAPTER

- [Proxy RADIUS Accounting](#) | 485

This chapter presents an overview of how you might use Proxy RADIUS as a back-end authentication or accounting method. This chapter contains these topics:

For complete details on proxy RADIUS, see [“Administering Proxy RADIUS” on page 153](#).

Proxy RADIUS Accounting

Steel-Belted Radius Carrier can relay an Accounting-Request to some other RADIUS server, which records the data according to its own, locally configured RADIUS accounting options. (You have the option of specifying that the data also be recorded locally on the Steel-Belted Radius Carrier server.) The set of conventions for relaying packets between cooperating RADIUS servers is known as *proxy RADIUS*, and is defined in the RADIUS standard.

For more information about Steel-Belted Radius Carrier’s proxy capabilities and configuration details, refer to [“Administering Proxy RADIUS” on page 153](#) and [“Configuring a Proxy RADIUS Realm” on page 201](#).

HSS-Subscriber Database

The Home Subscriber Server (HSS)-subscriber database is the primary subscriber database that contains subscriber and authentication information required for the 3GPP subscribers to access the WLAN interworking service. The SBR Carrier matches data received in the authentication/authorization request with the information in the HSS-subscriber database. If a match is found and the subscriber's credentials are correct, then the SBR Carrier accepts the request. If no match is found or a problem is found with the credentials, the SBR Carrier rejects the request. The HSS-subscriber database also stores the primary profile data which is downloaded by the SBR Carrier for policy decisions.

The SBR Carrier coordinates with other SBR Carriers through a registration mechanism in the HSS-subscriber database. The first SBR Carrier that authenticates a particular subscriber is registered as the “responsible” server in the HSS-subscriber database. Any subsequent authentications for that subscriber will be redirected to the “responsible” server, as long as it remains registered. After the subscriber has left the network, the registration may be purged by the SBR Carrier or the HSS-subscriber database. Once the subscriber joins back, the “responsible” server is again elected.

For more information about usage of HSS-subscriber database, see [“Administering the Diameter Policy” on page 376](#) and [“Administering Request Routing Rules” on page 417](#).



Management Interfaces

[Simple Network Management Protocol](#) | **488**

[Using the LDAP Configuration Interface](#) | **497**

Simple Network Management Protocol

IN THIS CHAPTER

- [SNMP and Steel-Belted Radius Carrier Overview | 488](#)
- [Configuring the SNMP Agent | 491](#)
- [Running the SNMP Agent | 492](#)
- [Logging Behavior of the SNMP Agent | 493](#)
- [Verifying SNMP Agent Operation | 494](#)
- [Resetting Rate Statistics | 496](#)
- [Troubleshooting | 496](#)

This chapter describes how to configure and to use the Simple Network Management Protocol (SNMP) to monitor the Steel-Belted Radius Carrier server. This chapter contains these topics:

SNMP and Steel-Belted Radius Carrier Overview

The Steel-Belted Radius Carrier uses standard MIBs to report routine server operations and seven proprietary MIBs to report RADIUS and Diameter operations and statistics. SNMP-GET, SNMP-GETNEXT, SNMP-TRAP (SNMPv1), and SNMP-NOTIFICATION (SNMPv2c) are fully supported; SNMP-SET is supported, but all values in the MIB are Read-Only.

For more information about SNMP, see the *SBR Carrier Reference Guide*.

The SBR Carrier SNMP Package

The SBR Carrier SNMP agent is based on version 5.4.0 of the open-source **net-snmp** toolkit, customized and optimized to support Steel-Belted Radius Carrier.

NOTE: You *cannot* replace or update the SBR Carrier SNMP agent (**jnp_{rsnmpd}**) with newer software downloaded from the net-snmp website (www.net-snmp.org), but the net-snmp website is a good source of additional information about SNMP usage and operation.

Steel-Belted Radius Carrier supports SNMP Versions 1 (SNMPv1) and 2c (SNMPv2c).

Steel-Belted Radius Carrier does not support SNMP version 3 (SNMPv3).

Supported MIBs

Steel-Belted Radius Carrier supports IETF-standard MIBs for RADIUS server authentication and accounting, and proprietary MIBs ([Table 56 on page 489](#)) that record information about SBR Carrier traps and operating statistics. All are stored in **radiusdir/snmp/mibs**.

Table 56: Proprietary Steel-Belted Radius Carrier MIBs

MIB	Description
fnkrate.mib	Maintains SBR Carrier rate statistics.
fnkradtr.mib	Defines the structure of SBR Carrier traps (SNMP version 1).
funkradtr-v2.mib	Defines the structure of SBR Carrier traps (SNMP version 2c).
jnx-smi.mib	Defines Juniper Networks overall MIB hierarchy.
jnx-aaa.mib	Defines Juniper Networks AAA specific MIB hierarchy.
jnx-diameter-base-protocol.mib	Defines the structure of Diameter traps.
jnx-diameter-nas-application.mib	Maintains Diameter NASREQ application specific counters.

Detailed trap information about these MIBs is in the *SBR Carrier Reference Guide*.

Of the public MIBs delivered with SBR Carrier and listed in [Table 57 on page 490](#), several provide a foundation for the Juniper Networks proprietary MIBs. Although the exact list of required MIBs may vary from browser to browser, the files in [Table 57 on page 490](#) marked **required** form the default set that most MIB browsers use to support the proprietary MIBs. Check your MIB browser's specific requirements, but in most cases, this is also the list of MIBs that should be copied to the browser.

Steel-Belted Radius Carrier supports the IETF-standard MIBs for RADIUS server authentication and accounting, MIBs provided by net-snmp.net, and three proprietary MIBs that record information about SBR Carrier traps and operating statistics. [Table 57 on page 490](#) lists the MIBs delivered with Steel-Belted Radius Carrier.

Table 57: MIBs Delivered with Steel-Belted Radius Carrier

MIB	Required	MIB Origin	Function
fnkradtr.mib	Yes	Proprietary	Defines the structure of Steel-Belted Radius Carrier traps (SNMP version 1).
fnkradtr-v2.mib	Yes	Proprietary	Defines the structure of Steel-Belted Radius Carrier traps (SNMP version 2c).
fnkrate.mib	Yes	Proprietary	Maintains Steel-Belted Radius Carrier rate statistics.
jnx-smi.mib	Yes	Proprietary	Defines Juniper Networks overall MIB hierarchy
jnx-aaa.mib	Yes	Proprietary	Defines Juniper Networks AAA specific MIB hierarchy
jnx-diameter-base-protocol.mib	Yes	Proprietary	Defines the structure of Diameter traps
jnx-diameter-nas-application.mib	Yes	Proprietary	Maintains Diameter NASREQ application specific counters
net-snmp.mib	Yes	net-snmp.net	Define the infrastructure of the Net-SNMP project enterprise MIB tree.
net-snmp-agent.mib	Yes	net-snmp.net	Defines control and monitoring structures for the Net-SNMP agent.
rfc1155-smi.mib	Yes	IETF	Defines the structure of management information for SNMP version 1.
rfc1212.mib	Yes	IETF	Defines MIB syntax.
rfc1213.mib	No	IETF	Provides network management of TCP/IP-based networks (MIB-II).
rfc1215.mib	Yes	IETF	Provides SNMP trap definitions.
rfc2271.mib	Yes	IETF	Provides SNMP framework definitions.

Table 57: MIBs Delivered with Steel-Belted Radius Carrier (continued)

MIB	Required	MIB Origin	Function
rfc4668.mib	No	IETF	Maintains authentication client statistics.
rfc4669.mib	No	IETF	Maintains authentication server statistics.
rfc4670.mib	No	IETF	Maintains accounting client statistics.
rfc4671.mib	No	IETF	Maintains accounting server statistics.
SNMPv2.mib	No	Proprietary	Defines traps about the state of the SNMP agent.
SNMPv2-CONF.mib	Yes.	Proprietary	Defines conformance groups for SNMP version 2c.
SNMPv2-SMI.mib	Yes.	Proprietary	Defines the structure of management information for SNMP version 2c.

Configuring the SNMP Agent

All SNMP files are installed on all Steel-Belted Radius Carrier servers, but installers have the option not to activate SNMP during the initial server configuration. If SNMP is not configured during server installation, you can activate it by rerunning the Steel-Belted Radius Carrier configuration script file, taking care not to change any other system parameters.

The files listed in [Table 58 on page 491](#) control SNMP.

Table 58: SNMP Configuration Files

Filename	Function
jnprsnmpd.conf	Stores settings for the Steel-Belted Radius Carrier SNMP agent.
testagent.sh	Test script that verifies the Steel-Belted Radius Carrier SNMP agent is operating correctly.

To change the current SNMP configuration edit ***radiusdir/snmp/install/jnprsnmpd.conf*** to reflect the network environment.

The file is self-documenting. For more information about the **jnprsnmpd.conf** file, see the *SBR Carrier Reference Guide*.



CAUTION: The **jnprsnmpd.conf** file is very sensitive to stray white space and the order in which sections and parameters appear. Mistakes in this file can disable SNMP.

- Make sure to make a backup copy of the file before making any changes.
- While editing the file, do not make any unnecessary changes and follow the embedded examples as closely as possible.
- When specifying networks as in 172.28.68.0/24 in the line “com2sec mynetwork 172.28.68.0/24 public” make sure the trailing 32-x bits of the IP address is zero as specified by the trailing /x notation; for example, 32-24=8 bits in this case.

To verify that the **jnprsnmpd** SNMP agent functions, run the **radiusdir/snmp/bin/testagent.sh** script.

If the SNMP environment has been changed to use a port other than the default SNMP port, 161, be sure to specify that port number in the **testagent.sh** file.

NOTE: Steel-Belted Radius Carrier runs its own SNMP agent, but on most servers, additional SNMP agents are also in operation. In general, only one application can use a socket port; they are not shared resources. This means that SBR Carrier SNMP agent may conflict with Solaris SNMP or any other SNMP agent that uses socket port 161, the default SNMP port.

If you know that other agents already use port 161, change the Steel-Belted Radius Carrier port assignment by editing both **radiusdir/snmp/install/init.jnprsnmpd** and **radiusdir/snmp/bin/testagent.sh**. Remember to check your MIB browser to determine whether it also needs a corresponding port adjustment to communicate with the SBR Carrier server.

Running the SNMP Agent

You can start and stop the SNMP daemon manually, and force it to reread the **radiusdir/snmp/install/jnprsnmpd.conf** configuration file.

Starting the SNMP Agent

To start the SNMP agent, on the command line, execute:

```
/etc/init.d/init.jnprsnmpd start
```

Stopping the SNMP Agent

To stop the SNMP agent, on the command line, execute:

```
/etc/init.d/init.jnprsnmpd stop
```

Rereading the `jnprsnmpd.conf` File

To force the `jnprsnmpd` agent to reread its configuration file, first identify the process id of the `jnprsnmpd` agent and execute:

```
kill -HUP pldServer
```

Where *pldServer* is the process ID of the `jnprsnmpd` agent process.

Logging Behavior of the SNMP Agent

The logging behavior of the SNMP agent process, `jnprsnmpd`, is controlled by command line options that are specified in the `/etc/init.d/init.jnprsnmpd` script.

- The “-Lf <file>” option specifies the absolute path name of the log <file> to be written.
- The “-A” option, if present, causes the log file to be appended rather than rewritten each time the SNMP agent is restarted (not recommended unless a periodic clean-up mechanism is in place).
- The “-D ALL” option, if present, enables all possible debug messages to be logged (extremely verbose and not recommended for normal operation).

By default, the SNMP agent process, `jnprsnmpd`, is configured to log to the `/opt/JNPRsbr/radius/snmp/jnprsnmpd.log` file, rewriting the file each time it is restarted.

- After starting the SNMP agent (`/etc/init.d/init.jnprsnmpd start`), the following line is logged:
NET-SNMP version 5.4
- Upon receiving a SIGHUP (1) signal (`/etc/init.d/init.jnprsnmpd hup`), the following line is logged:
Reconfiguring daemon
- Shortly before a successful stop (`/etc/init.d/init.jnprsnmpd stop`), the following line is logged:
Received TERM or STOP signal... shutting down...

NOTE: It is possible for operators to stop the **jnpnsnmpd** process abruptly by directly sending other signals, for example, **kill -9**, in which case no messages will be logged.

Verifying SNMP Agent Operation

You can perform limited SNMP operations directly on the SBR Carrier server platform so that you can verify that the SNMP agent is running properly.

Running the `testagent.sh` Script

To execute a Get command that retrieves the system description and confirms basic SNMP operation, execute:

```
radiusdir /snmp/bin/testagent.sh
```

Using the `snmpget` Command

To use the **snmpget** command-line tool to obtain a scalar value, first set the library variable, then run the command.

To set the **LD_LIBRARY_PATH** variable for 64-bit Linux, execute these two strings:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:radiusdir/snmplib
export LD_LIBRARY_PATH
```

To set the **LD_LIBRARY_PATH_64** variable for 64-bit Solaris, execute the following strings:

```
LD_LIBRARY_PATH_64=radiusdir:radiusdir/system/lib/:radiusdir
/openldap:/usr/sfw/lib/sparcv9:/usr/lib/mps/sparcv9:/usr/sbin/lib/sparcv9:/lib/sparcv9:/
usr/lib/sparcv9:/usr/openwin/lib/sparcv9:/usr/dt/lib/sparcv9:/usr/proc/lib/
usr/local/lib/sparcv9:/opt/sfw/lib/sparcv9:/usr/ccs/lib/sparcv9:/usr/ucblib/sparcv9:
```

To run **snmpget**, follow this example syntax that combines the OID prefix with the name of the scalar:

```
radiusdir /snmp/bin/snmpget -M radiusdir /snmp/mibs -m all -c community host
iso.org.dod.internet.mgmt.mib-2.radiusMIB.radiusAuthentication.radiusAuthServMIB.
radiusAuthServMIBObjects.radiusAuthServ.radiusAuthServIdent
```

To execute a similar command, replace **radiusdir**, **community**, and **host** in the example with the values appropriate for your network and put the entire command on a single line. You can use the numerical equivalent for the OID prefix and a 0 suffix. For example:

```
radiusdir /snmp/bin/snmpget -M radiusdir /snmp/mibs -m all -c community host 1.3.6.1.2.1.67.1.1.1.1.1.0
```

Using the snmpwalk Command

To use the **snmpwalk** command-line tool to obtain a table, first set the library path variable, then run the command.

To set the **LD_LIBRARY_PATH** variable for 64-bit Linux, execute these two strings:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:radiusdir/snmplib
export LD_LIBRARY_PATH
```

To set the **LD_LIBRARY_PATH_64** variable for 64-bit Solaris, execute the following strings:

```
LD_LIBRARY_PATH_64=radiusdir:radiusdir/system/lib/:radiusdir
/openldap:/usr/sfw/lib/sparcv9:/usr/lib/mps/sparcv9:/usr/sbin/lib/sparcv9:/lib/sparcv9:/
usr/lib/sparcv9:/usr/openwin/lib/sparcv9:/usr/dt/lib/sparcv9:/usr/proc/lib:/
usr/local/lib/sparcv9:/opt/sfw/lib/sparcv9:/usr/ccs/lib/sparcv9:/usr/ucblib/sparcv9:
```

Then follow this example syntax, which uses the OID prefix of the table:

```
radiusdir /snmp/bin/snmpwalk -M radiusdir /snmp/mibs -m all -c community host
iso.org.dod.internet.mgmt.mib-2.radiusMIB.radiusAuthentication.radiusAuthServMIB.
radiusAuthServMIBObjects.radiusAuthServ.radiusAuthClientTable. radiusAuthClientEntry
```

To execute a similar command, replace **radiusdir**, **community**, and **host** in the preceding example with the values appropriate for your network, and put the entire command on a single line.

You can use the numerical equivalent for the OID prefix and a 0 suffix. For example:

```
radiusdir /snmp/bin/snmpwalk -M radiusdir /snmp/mibs -m all -c community host 1.3.6.1.2.1.67.1.1.1.1.15.1
```

Resetting Rate Statistics

SNMP statistics are automatically reset each time the server is restarted.

To reset all statistics to zero, identify the process id of the server and execute:

```
kill -USR2 pIdServer
```

Where *pIdServer* is the process ID of your Steel-Belted Radius Carrier server.

Troubleshooting

- Check the SNMP log file (*radiusdir/snmp/jnprsnmpd.log*) for event information related to the SNMP agent.
- If the SNMP agent fails to run (that is, you do not see it listed when you run the **ps -A -f | grep jnprsnmpd** command), review the log file to determine whether another SNMP agent is running.
- Two SNMP agents cannot share the same port. If another SNMP that uses the same port (port 161 by default) is running on your system, the Steel-Belted Radius Carrier SNMP agent may fail to start. Review the log file to determine whether a conflict over Port 161 is occurring.
- If the Steel-Belted Radius Carrier SNMP agent conflicts with another SNMP agent and you cannot uninstall or disable the other SNMP agent, change the port assignment.
- If an SNMP MIB browser or other management station cannot reach the SNMP agent, verify that the agent is running by executing: **ps -Af | grep jnprsnmpd**. If the agent is listed, run the **testagent.sh** script to verify the agent is responding.
 - If the **testagent.sh** script succeeds, verify that the community strings and access controls configured in the **jnprsnmpd.conf** file are correct.
 - If the **testagent.sh** script fails, verify that the script uses the same port number specified in the **jnprsnmpd.conf** file (the default is Port 161) and that the **jnprsnmpd.conf** file grants access to **localhost**, which **testagent.sh** requires to function.

Using the LDAP Configuration Interface

IN THIS CHAPTER

- [LDAP Configuration Interface File | 497](#)
- [LDAP Configuration Interface Overview | 498](#)
- [LDAP Virtual Schema | 503](#)
- [LDAP Rules and Limitations | 509](#)
- [LDAP Command Examples | 513](#)
- [LDIF File Examples | 522](#)
- [Statistics Variables | 529](#)

This chapter contains information about configuring Steel-Belted Radius Carrier to support the Lightweight Directory Access Protocol (LDAP) using the Steel-Belted Radius Carrier LDAP Configuration Interface (LCI). This chapter contains these topics:

LDAP Configuration Interface File

Use the **radius.ini** file to establish settings for the LCI ([Table 59 on page 497](#)). For more information about **radius.ini**, see the *SBR Carrier Reference Guide*.

Table 59: LDAP Configuration Interface File

File Name	Function
radius.ini	Specifies whether the LCI is enabled, the port used for LCI communication, and the interfaces on which Steel-Belted Radius Carrier listens for LCI requests.

NOTE: While running the Steel-Belted Radius Carrier configuration script, the following information is gathered and the corresponding parameters in the **radius.ini** file are populated:

- The LCI status (enabled or disabled).
- The LDAP TCP port number that is used for communication between Steel-Belted Radius Carrier and the LDAP client.
- The interface addresses on which LCI should be enabled.
- LCI password.

For more information about running the Steel-Belted Radius Carrier configuration script, see the *Running the Steel-Belted Radius Carrier Configure Script* section in the *SBR Carrier Installation Guide*.

LDAP Configuration Interface Overview

The LCI provided by Steel-Belted Radius Carrier consists of an LDAP interface in the Steel-Belted Radius Carrier server and an LDAP virtual schema. The LDAP virtual schema presents the structure of the Steel-Belted Radius Carrier database in a manner that can be understood by the LDAP client utilities. The LCI uses the virtual schema to retrieve, modify, and delete entries in the database.

NOTE: The LDAP-SQL bridge, previously shipped as part of SBR HA 5.5, has been replaced by SBR Carrier's LDAP Configuration Interface (LCI).

To use, enter a search query such as:

```
ldapsearch -V2 -h localhost -p667 -D "cn=admin, o=radius" -w radius -s
sub -b "framed-ip-address=10.1.105.122,radiusstatus=sessions_by_ipaddress,
o=radius" framed-ip-address="*"
```

This search produces the following output:

```
dn:acct-session-id=f9248bc54c60230c003d9fbd00000000,
client=10.13.101.201,framed-ip-address=10.1.105.122,
radiusstatus=sessions_by_ipaddress,o=radius
objectclass: top
objectclass: radiusstatus
```



```

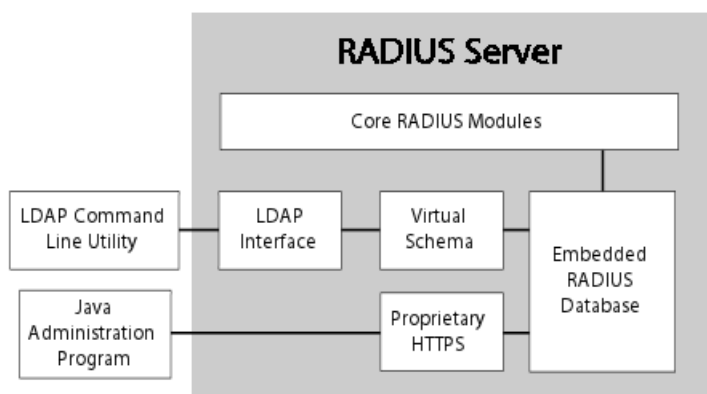
radiusstatus: sessions_by_ipaddress
client: 10.13.101.201
acct-session-id: 13
nas-ip-address: 192.168.1.16
nas-port: 0
framed-ip-address: 10.1.105.122
session-start-time: 1281473604
fullname: TEST
elapsed: 616

```

NOTE: The LCI query limits only to 100 records when you search for current sessions using the `ldapsearch` utility. This avoids any LCI query from destabilizing either the SBR Carrier on which it is running, or the SSR itself by holding record locks while reading.

Figure 193 on page 499 illustrates the relationship between LDAP components, the Administrator, and the configuration database.

Figure 193: LDAP Components



LDAP Utilities

Freeware LDAP utilities, such as `ldapsearch`, `ldapdelete`, and `ldapmodify`, act as clients of the LDAP interface. LDAP utilities let you read and modify an LDAP database.

- **ldapsearch**—The `ldapsearch` utility locates and retrieves LDAP directory entries. The `ldapsearch` utility opens a connection to an LDAP interface using the specified distinguished name and password, binds, and locates entries based on the specified search filter. A search can return a single entry, an entry's

immediate subentries, or an entire tree or subtree. Search results are returned in LDAP Data Interchange Format (LDIF) format.

- **ldapdelete**—The **ldapdelete** utility deletes entries from an existing LDAP directory. **ldapdelete** opens a connection to the specified server using the distinguished name and password you provide, binds, and deletes the entry or entries.
- **ldapmodify**—The **ldapmodify** utility adds or modifies entries in an existing LDAP directory. **ldapmodify** opens a connection to an LDAP interface using the distinguished name and password you supply, binds, and adds or modifies the entries based on the LDIF update statements contained in a specified file.

LDAP Requests

LDAP requests are submitted in two ways:

- By specifying options on the LDAP configuration interface command line.
- By placing instructions and data into an LDIF file, which you then process by invoking an LDAP command line utility using the **-f** option.

Because communication between the LDAP client and server is unencrypted, the LDAP utilities should be run on the same computer as Steel-Belted Radius Carrier.

Downloading the LDAP Utilities

To use the LCI, you need the **ldapsearch**, **ldapmodify**, and **ldapdelete** utilities. You can download the **ldapsearch** utility as follows:

1. Use a browser to navigate to <http://www.oracle.com/technetwork/indexes/downloads/index.html>.
2. When the Sun ONE Directory SDK (software development kit) download page appears, click the **Download** link at the bottom of the page.
3. If you are prompted to register, complete the registration form.
4. When you are prompted to accept the license agreement, click the **Accept** button and then click **Continue**.
5. Download the SDK by clicking the link for the version of the SDK that is appropriate for your computer.
6. When the download is completed, extract the files from the compressed image to a directory on your computer.

To run the LDAP utilities, execute them from this directory. If you set the path environment variable to point to this directory, you can run them from any location on the system.

NOTE: The examples that follow assume you are using the LDAP utilities provided as part of the Sun ONE Directory SDK. If you are using LDAP utilities from another source, the command options you use may be different. Consult the documentation for your LDAP utilities for more information.

LDAP Version Compliance

The LDAP interface in Steel-Belted Radius Carrier complies with version 2 of the LDAP specification. You should use the **-V 2** command option to direct the utilities to use version 2 features. For example:

```
ldapmodify -c -V 2 -p 354 -D "cn=admin,o=radius" -w radius -f filename
```

Configuring the LDAP TCP Port

To avoid conflicts with LDAP services that may already be installed, the default port number for communication between Steel-Belted Radius Carrier and the LDAP client is 667. You can configure Steel-Belted Radius Carrier to use a different TCP port to communicate with an LDAP client. For example, you can change this port number to 389, the standard LDAP TCP port, if you are certain doing so will not create port number conflicts with other applications.

The following example configures Steel-Belted Radius Carrier to use TCP port 354.

1. In the **radius.ini** file, uncomment the [LDAP] section, set **Enable** to 1, and set the **TCPPort** field to the port number you want to use. For example:

```
[LDAP]
Enable = 1
TCPPort = 354
```

2. If you want to specify the interfaces on which Steel-Belted Radius Carrier listens for LCI requests, add a [LDAPAddresses] section to the **radius.ini** file. This section should contain a list of IP addresses, one per line. For example:

```
[LDAPAddresses]
192.168.12.45
10.10.10.25
```

If the [LDAPAddresses] section is omitted or empty, Steel-Belted Radius Carrier listens for LCI requests on all bound IP interfaces.

You must specify the port number (by means of the **-p** option) when you run the LDAP utilities. For example:

```
ldapsearch -V 2 -p 354 -D "cn=admin,o=radius" -w radius -s sub -T -b "radiusclass=Client,o=radius" radiusname=*
```

Example

The Steel-Belted Radius Carrier server at the Good Times Clock Company has two network interfaces. The first interface (192.168.10.40) connects to the corporate network. The second interface (192.168.20.50) connects to a dedicated administrative VLAN accessible only from the local subnet. To limit access to the LCI to network administrators, the [LDAPAddresses] section of **radius.ini** specifies that LCI requests must come through the administrative interface (192.168.20.50). LCI requests coming through the corporate network interface (192.168.10.40) are ignored.

Configuring the LCI Password

After you enable the LCI, change the default LCI password to prevent unauthorized LDAP clients from accessing your database. After you install the LDAP utilities and verify that they work, perform the following steps:

1. Create a text file called **temp.ldif** with the following contents:

```
dn: radiusclass=server,o=radius
changetype: modify
replace: server-password
server-password: new-password
```

Where *new-password* is the LCI password you want to use.

2. Change the **radius.ini** [LDAP] setting to **Enable=1**.
3. Restart Steel-Belted Radius Carrier.
4. Execute the following command:

```
ldapmodify -V 2 -h ip-address -p port -D "cn=admin,o=radius" -w oldpassword -f temp.ldif
```

Where:

-h ip-address specifies the IP address of the Steel-Belted Radius Carrier server.

-p port specifies the port number specified in the [LDAP] section of the **radius.ini** file.

-w oldpassword specifies the current password (which is **radius** by default).

5. Verify that the password change was successful by executing the following command:

```
ldapsearch -V 2 -h ip-address -p port -D "cn=admin,o=radius" -w newpassword -s sub -T -b "o=radius"
radiusclass=server
```

Where:

- h *ip-address* specifies the IP address of the Steel-Belted Radius Carrier server.
- p *port* specifies the port number specified in the [LDAP] section of the **radius.ini** file.
- w *newpassword* specifies the password configured in the **temp.ldif** file.

After you verify that the password change has been successful, delete the **temp.ldif** file and any other file that contains a clear-text copy of the modified LCI password.

NOTE: The LDAP Configuration Interface does not support Secure Sockets Layer (SSL).

LDAP Virtual Schema

The LDAP interface uses the virtual schema illustrated in [Figure 194 on page 505](#)—[Figure 199 on page 508](#) to represent the structure of the Steel-Belted Radius Carrier database. LDAP clients use the virtual schema to exchange configuration data over the LDAP configuration interface.

Many of the top-level items in the LDAP virtual schema correspond to pages in the WEB GUI, as shown in [Table 60 on page 503](#).

Table 60: LDAP Schema and Web GUI Dialogs

Item	See
radiusclass=client	“Administering RADIUS Clients and Client Groups” on page 104
radiusclass=ClientGroup	“Administering RADIUS Location Groups” on page 114
radiusclass=native-user, ...	“Administering Users” on page 120
radiusclass=profile	“Administering Profiles” on page 144
radiusclass=proxy	“Administering Proxy RADIUS” on page 153
radiusclass=tunnel	“Administering RADIUS Tunnels” on page 168
radiusclass=server	“Setting Up EAP Methods” on page 253

Table 60: LDAP Schema and Web GUI Dialogs *(continued)*

Item	See
radiusclass=ip-addr-pool	“Administering Address Pools” on page 182
radiusstatus=statistics	“Displaying Statistics” on page 817
radiusstatus=sessions	“Current Sessions Overview” on page 718 NOTE: LDAP searches that call radiusstatus=sessions can adversely affect Steel-Belted Radius Carrier performance. When possible, search using the sessions_by keywords. New attributes can be added by updating the session table.
radiusstatus=sessions_by_mobile_session	<p>You can use the LCI to locate a session by its mobile session identifier (the concatenation of the values of the User-Name and 3GPP-NSAPI attributes included in authentication and accounting requests).</p> <p>Use the radiusstatus=sessions_by_mobile_session node to specify a filter of mobile-session-id=session_id to locate a specific mobile session. You can also use this part of the schema to delete sessions.</p>

NOTE: **radiusstatus** items can be read, but they cannot be modified.

NOTE: [Figure 196 on page 506](#) and [Figure 198 on page 508](#) incorrectly list Tribe as an available attribute for session queries. This attribute is not supported in the LDAP schema.

Figure 194: LDAP Schema (1 of 6)

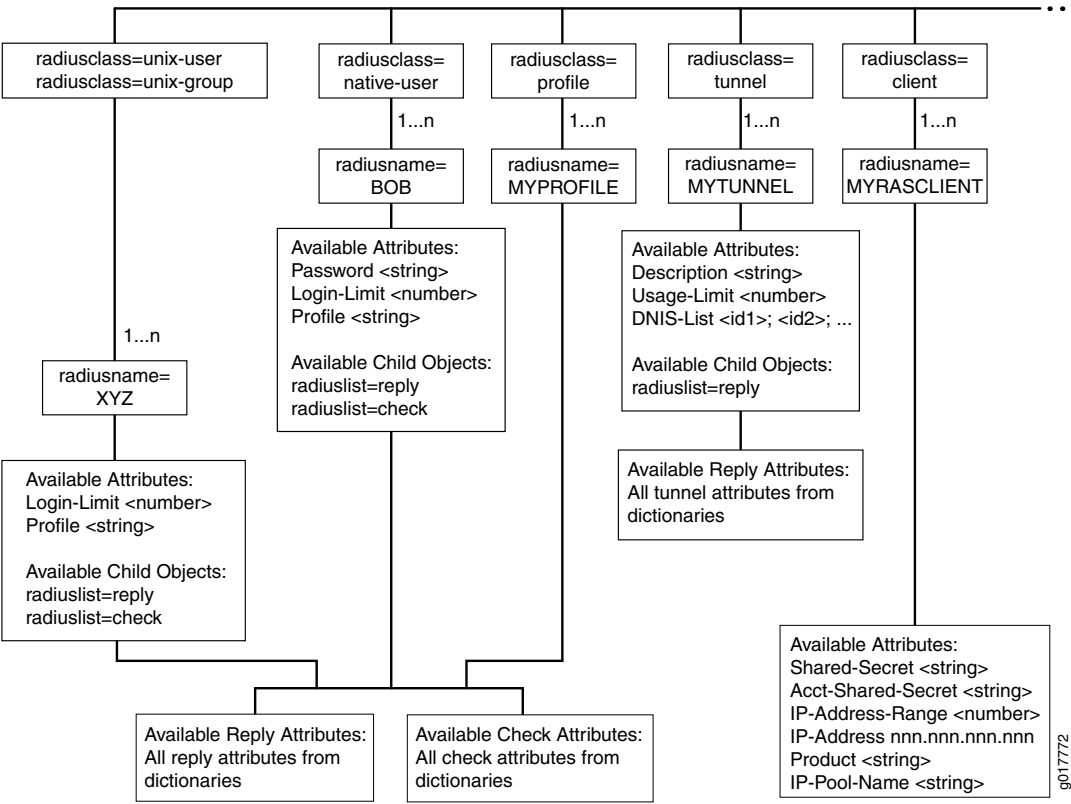
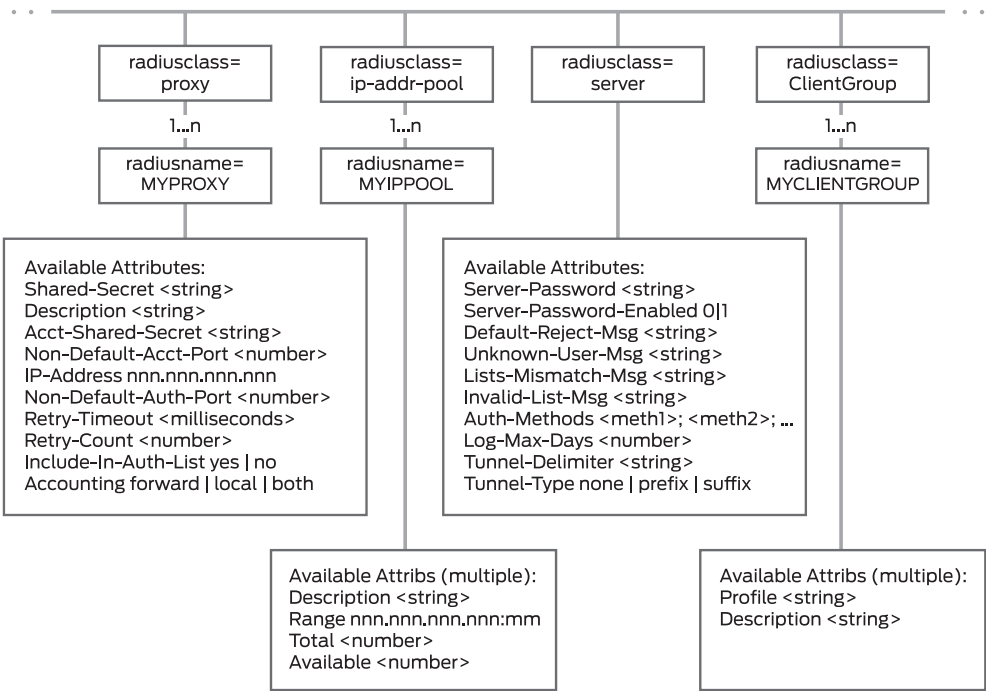


Figure 195: LDAP Schema (2 of 6)



NOTE: The radiusclass=ip-addr-pool applies only to the standalone version and not to the cluster.

Figure 196: LDAP Schema (3 of 6)

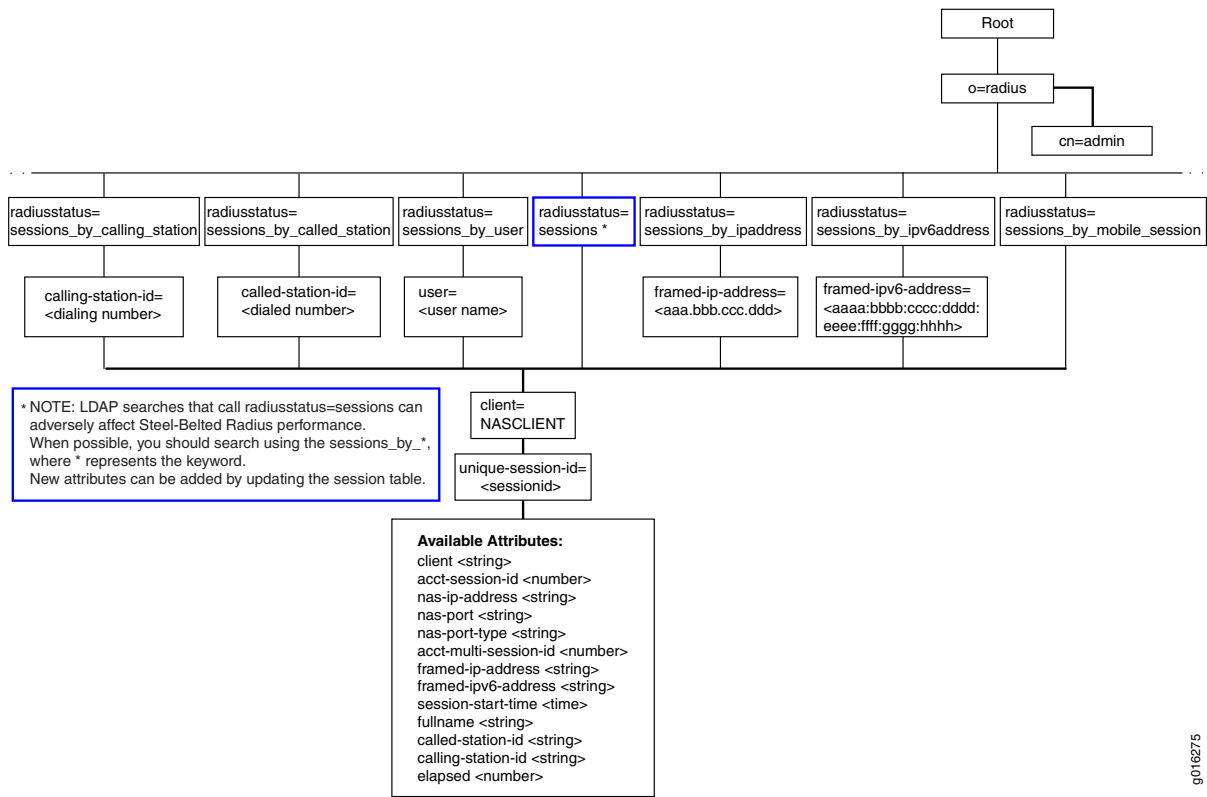
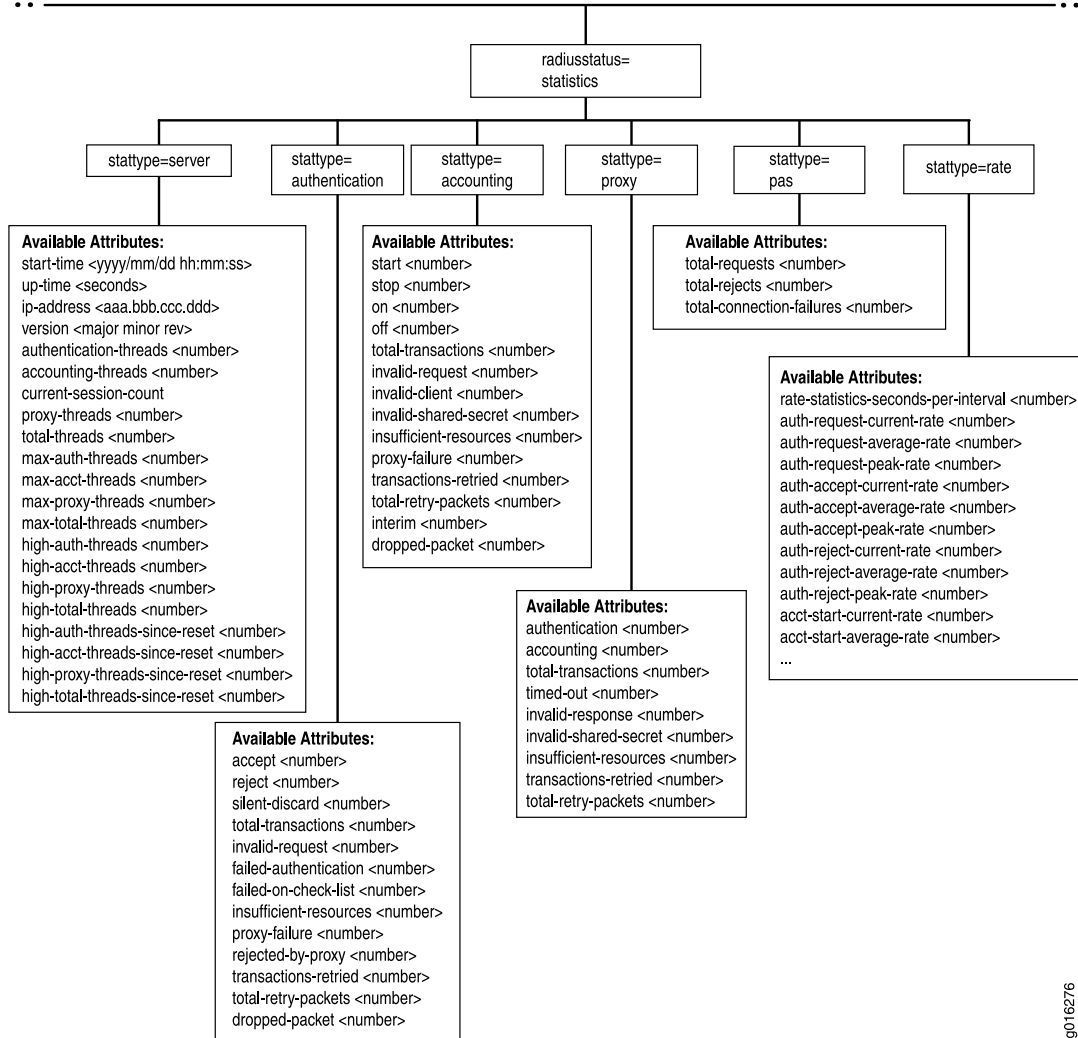


Figure 197: LDAP Schema (4 of 6)



g016276

Figure 198: LDAP Schema (5 of 6)

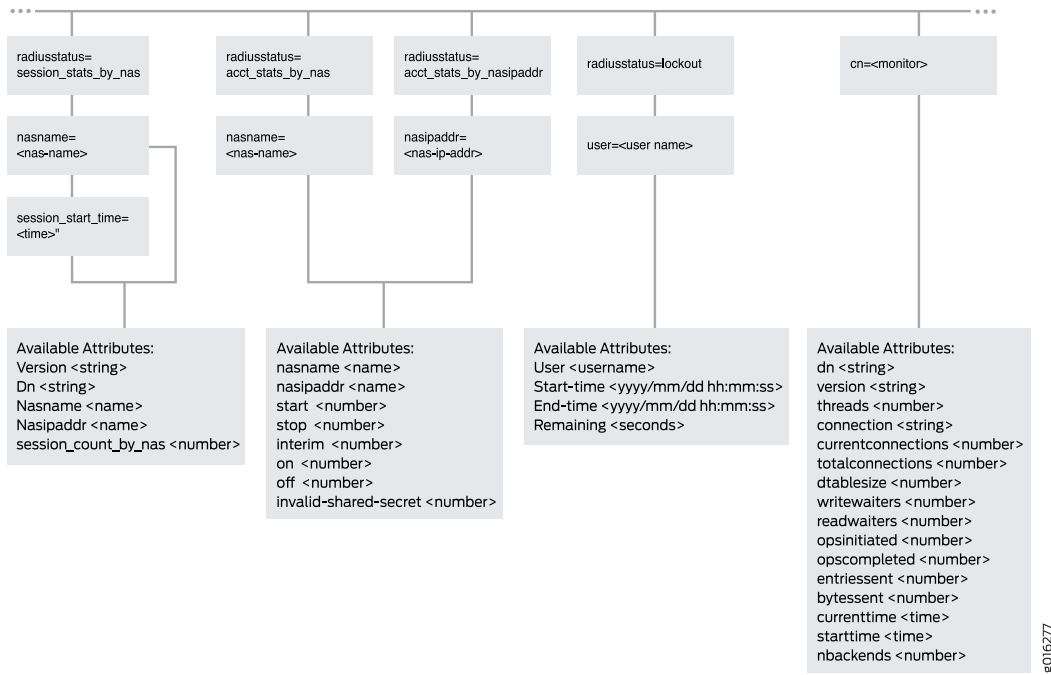
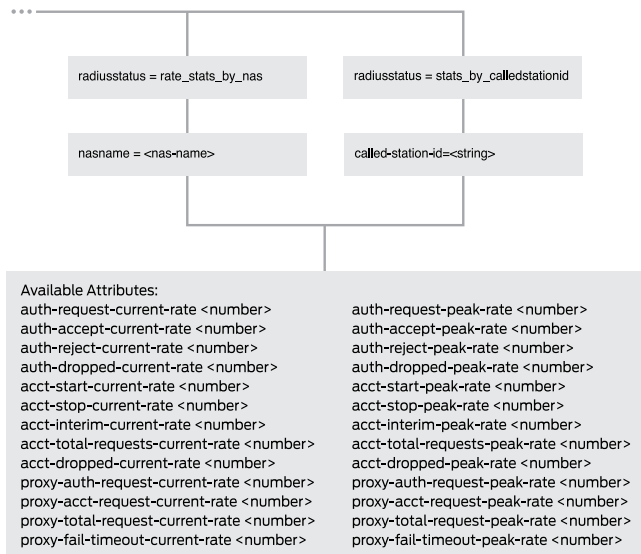


Figure 199: LDAP Schema (6 of 6)



LDAP Rules and Limitations

While the LDAP virtual schema diagram shows as much of the detail of the LDAP virtual schema as possible, take these rules and limitations into consideration.

- **Bind request**—All attempts to perform operations on the virtual schema must be preceded by an LDAP Bind request that authenticates the administrator to the Steel-Belted Radius Carrier server. The Bind request must reference a Steel-Belted Radius Carrier administrative account and must provide the password that authenticates that account. This translates into the following command line options for each invocation of the LDAP utilities:

```
-D "cn=AdminName,o=radius" -w AdminPassword
```

Where *AdminName* is the administrative account name and *AdminPassword* is its password.

- **Uppercase and lowercase**—The uppercase/lowercase rules for object names are the same as in the Web GUI; that is, almost all object names are stored in the database in uppercase format. The exception to this rule is that UNIX User/Group names are maintained in the case specified in the LDIF files.
- **Attributes**—When you enter attributes, make sure that the attribute name matches the name found in the dictionary and that the attribute's value is consistent with the syntax type for the attribute. The LDAP virtual schema does not list all the dictionary attributes that are available in Steel-Belted Radius Carrier.
- **IP addresses**—The **ipaddr-pool** type in the dictionary can represent an IP address or a pool name. If the value specified begins with the marker string *[pool]*, the token that follows the marker string is assumed to be an IP pool name; otherwise, it must be a valid IP address. If it is neither, the operation fails.

Address ranges in IP address pool objects are specified in the form *IPaddress:NumberOfAddresses*. An example of a valid range is 128.22.12.45:34.

- **Substrings**—An attribute may have a value that consists of a list of strings. For example, the DNIS list in a tunnel entry and the authentication method list may consist of multiple substrings. The rule for specifying the data portion for these lists is that semicolons must delimit substrings. For example, a DNIS list for a tunnel entry might be specified as **555-1212;5551212**. If a semicolon needs to appear inside a substring, it must have an escape character (\) before it.
- **Password syntax**—Passwords that are set or retrieved from the database may consist of one of the following:
 - A clear-text password of the form **{x-clear}clear-text-password-string** if the password is weakly encrypted in the database.
 - A string of the form **{x-md5}xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx** if the password is stored as a one-way md5 hash.

- A string of the form `{x-md5}[encrypt]clear-text-password-string` indicates that, although the password is specified in clear-text form, it is to be stored as a hash. This is applicable only for setting the password through LCI.

White space in a password is treated as follows:

- When clear-text passwords are specified, the password is assumed to begin immediately after the right brace or right bracket. Adding a white space character, such as a space or tab, after the right brace or right bracket causes the white space to be considered part of the password.
- White space entered at the beginning of the attribute (before the left brace or left bracket) is ignored.
- White space entered between the right brace of `{x-md5}` and the left bracket of `[encrypt]` is ignored.
- All white space specified in the hexadecimal sequence describing a password hash is ignored.
- Profiles, check lists, and return lists—Steel-Belted Radius Carrier supports user definitions that include attribute subtractions of profile entries. To specify that a user attribute is to be considered a subtraction of a profile attribute, preface the attribute value with the string `%subtract%`.

Steel-Belted Radius Carrier permits user and profile check lists to include default values for attributes. Configuring a default value for an attribute means that, if a RADIUS request does not include this attribute, the request is not rejected. Instead, the value supplied as the default is used as if it were received as part of the request. To specify that a check list attribute is to be considered a default attribute, preface the attribute value with the string `%default%`.

Steel-Belted Radius Carrier permits user and profile return lists to include attributes whose values are set by copying the contents of received attributes. This feature is referred to as *attribute echoing*. To specify that a return list attribute is to be treated as an echo attribute, enter `%echo%` for the attribute value.

- Unspecified or 0.0.0.0 NAS IP address—When you display `acct_stats_by_nasipaddr` information, any NAS entries with an unspecified IP address or an IP address of 0.0.0.0 are omitted. Similarly, when you display `acct_stats_by_nas` information, any NAS entries with an unspecified IP address or an IP address of 0.0.0.0 have their `nasipaddr` attribute omitted.
- Duplicate NAS IP addresses—When displaying `acct_stats_by_nasipaddr` information, two NAS entries that contain the same (non-zero) IP address cause information about one of the entries to be displayed twice. This is the result of the ambiguity of the query and is not a bug.
- RADIUS client information displayed after deletion—If you define a RADIUS client entry, send some accounting traffic to it, and then delete the entry, the output of `ldapsearch` queries continue to list the deleted RADIUS client so that the per-NAS statistics add up to the total NAS statistics.

Using the LCI to Define Structured Attributes in Check Lists and Return Lists

The LCI (LDAP Control Interface) has been extended to facilitate structured attributes in check lists and return lists. Subject to certain restrictions, a structured attribute must be defined as a whole (as opposed

to defining individual subattributes) using either the raw hexadecimal representation of the entire structured attribute's binary payload, or an XML representation of the structured attribute hierarchy using the format outlined in this section.

When the response packet is formatted, the hexadecimal or XML data is formatted into the packet. For XML data, it is parsed into a structured attribute hierarchy and then formatted as normal.

If the hexadecimal representation is used, then the entire RADIUS attribute must be specified. If the XML representation is used, then it is not necessary to specify subattributes for which default values are defined.

The LCI assumes the data is in the hexadecimal representation unless the following conditions are met for the XML representation:

- The XML representation must be at least 16 characters and not more than 8192 characters in length.
- The XML representation must obey generic XML grammar. For example: it must begin and end with angle brackets, all angle brackets must be properly paired, and so on.

The LCI does not validate the contents of either hexadecimal or XML values. The LCI accepts hexadecimal and XML representations with flawed contents. It is the user's responsibility to ensure that the content is legal and, if XML representation is used, that the XML obeys the schema described in [“LCI XML Format” on page 512](#) and shown in [Figure 200 on page 512](#).

NOTE: Structured attributes (VSAs with subattributes) defined in return lists are added to the reply message as a whole unit, rather than their subattributes being added individually to any existing response VSAs. In this way they are treated just as unstructured VSAs.

For example:

- Attribute "ParentAttr" is defined as being a multivalue return list attribute, with possible subattributes "ChildAttrA" and "ChildAttrB".
- A response already has a copy of "ParentAttr" with subattribute "ChildAttrA", for example from an authentication process.
- A profile specifies that "ParentAttr" must be added with subattribute "ChildAttrB".

The result is a response with two ParentAttr structured attributes:

```
ParentAttr
  ChildAttrA
ParentAttr
  ChildAttrB
```

The result will *not* be a response with a single ParentAttr:

```
ParentAttr
  ChildAttrA
  ChildAttrB
```

LCI XML Format

Make the XML hierarchy reflect the hierarchy defined in the **.jdict** subattribute dictionary file in a nested set of **<attribute>** elements. All **<attribute>** elements must have a name attribute. Ensure that groups and sequences (the parent attribute, and subattribute types) are represented by **<attribute>** nodes without a value attribute, and will further **<attribute>** elements as children.

The proper format is shown in [Figure 200 on page 512](#).

Figure 200: LCI XML Format

```
<attribute name="attribute name" value="attribute value">
  <attribute .../>
</attribute>
```

For more information about subattributes and an example of the proper XML format you need to use with the LCI, as well as structured attribute dictionary definitions, see the *SBR Carrier Reference Guide*.

NOTE: When using the LCI command line utilities such as `ldapquery`, XML values for structured attributes are displayed encoded in a non-readable format. This encoding is base64 encoding which can be decoded with many command line or web-based utilities.

Alternatively, the problem can be avoided by using a graphical LDAP client.

LDAP Command Examples

This section explains how to use the `ldapdelete`, `ldapmodify`, and `ldapsearch` utilities to configure the server.

Searching for Records

You can use the `ldapsearch` command to extract information from the LDAP tree. The `ldapsearch` command can be used to find sessions either by a specific NAS or NAS and session start time combination. The command shown in [Figure 201 on page 513](#) lets you extract information about all RADIUS Native Users.

Figure 201: `ldapsearch` Command

```
ldapsearch -V 2 -p 354 -h 192.168.45.12  
-D "cn=oper,o=radius" -w radadmin -s sub -T -b  
"radiusclass=Native-User,o=radius" radiusname=*
```

NOTE: The LCI query limits only to 100 records when you search for current sessions using the `ldapsearch` utility. This avoids any LCI query from destabilizing either the SBRC on which it is running, or the SSR itself by holding record locks while reading.

Make sure to include a blank space between each option (for example, `-p`) and its value (for example, `354`). Command syntax is case-sensitive. See [Table 61 on page 514](#).

Table 61: Searching for Records Using the `ldapsearch` Command

ldapsearch Option	Meaning
<code>-V 2</code>	<p>Use LDAP Version 2 to communicate with the server.</p> <p>This option is not required, but it improves the performance of the transaction.</p> <p>NOTE: You must use the <code>-P 2</code> option in the following conditions:</p> <ul style="list-style-type: none"> • On a Solaris machine, if you use the LDAP version 2 that is shipped with the SBR Carrier package. • On a Linux machine, if you use the LDAP version 2 that is installed in your system. <p>NOTE: The LDAP interface in SBR Carrier complies only with version 2 of the LDAP specification.</p>
<code>-p 354</code>	<p>Use TCP port 354 to communicate with the LDAP interface of the server.</p> <p>The <code>-p</code> value must match the TCP port setting in the [LDAP] section of radius.ini. If the <code>-p</code> option is not specified, the LDAP utilities contact Steel-Belted Radius Carrier on the default port number (TCP port 389).</p>
<code>-h 192.168.45.12</code>	<p>Contact a remote host at the specified address or name.</p> <p>By default, ldapsearch tries to connect to the local host.</p>
<code>-D "cn=oper,o=radius"</code>	<p>Use the oper administrative account to authenticate the command.</p> <p>NOTE: You can use any administrative account name in place of oper in this example. Do not change the o=radius argument.</p>
<code>-w radadmin</code>	<p>Use an authentication password of radadmin.</p> <p>NOTE: The <code>-w</code> parameter value (in this case, radadmin) must match the password of the account named by the <code>-D</code> parameter.</p>
<code>-s sub</code>	<p>Perform a recursive subtree search from the base.</p>
<code>-T</code>	<p>Do not wrap long output lines to the next line.</p>

Table 61: Searching for Records Using the `Idapsearch` Command (continued)

Idapsearch Option	Meaning
-b "radiusclass=Client,o=radius"	Specifies the base from which the search operation starts.
radiusname=*	Specifies the selection criteria for the search.

Executing the `Idapsearch` command shown in [Figure 201 on page 513](#) against a Steel-Belted Radius Carrier server containing two Native User definitions produces an LDIF file similar to the output shown in [Figure 202 on page 515](#).

Figure 202: Search Results

```
dn: radiusname=KEVIN,radiusclass=Native-User,o=radius
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: KEVIN
password: {x-clear}secret1
profile: ISDN
login-limit: 2

dn: radiusname=MICHAEL,radiusclass=Native-User,o=radius
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: MICHAEL
password: {x-clear}secret99
profile: ISDN
login-limit: 2
```

Modifying Records

You can use the `Idapmodify` utility to update the Steel-Belted Radius Carrier configuration.

```
Idapmodify -c -V 2 -h example.host.com -p 354
-D "cn=oper,o=radius" -w radadmin -f filename
```

Be sure to include a blank space between each option (for example, `-p`) and its value (for example, `354`). Command syntax is case-sensitive. See [Table 62 on page 516](#).

Table 62: Modifying Records Using the `ldapmodify` Command

ldapmodify Option	Meaning
<code>-c</code>	Run the command in continuous mode; do not stop on errors.
<code>-V 2</code>	<p>Use LDAP Version 2 to communicate with the server.</p> <p>This option is not required, but it improves the performance of the transaction.</p> <p>NOTE: You must use the <code>-P 2</code> option in the following conditions:</p> <ul style="list-style-type: none"> • On a Solaris machine, if you use the LDAP version 2 that is shipped the SBR Carrier package. • On a Linux machine, if you use the LDAP version 2 that is installed in your system. <p>NOTE: The LDAP interface in SBR Carrier complies only with version 2 of the LDAP specification.</p>
<code>-h example.host.com</code>	<p>Contact a remote host at the specified address or name.</p> <p>If the <code>-h</code> option is not used, ldapsearch connects to the local database.</p>
<code>-p 354</code>	<p>Use TCP port 354 to communicate with the LDAP interface of the server.</p> <p>The <code>-p</code> value must match the <code>TCPPort</code> setting in the [LDAP] section of radius.ini. If the <code>-p</code> option is not specified, the LDAP utilities contact Steel-Belted Radius Carrier on the default port number (TCP port 389).</p>
<code>-D "cn=oper,o=radius"</code>	<p>Use the oper administrative account to authenticate the command.</p> <p>NOTE: You can use any administrative account name in place of oper in this example. Do not change the o=radius argument.</p>
<code>-w radadmin</code>	<p>Use an authentication password of radadmin.</p> <p>NOTE: The <code>-w</code> parameter value (in this case, radadmin) must match the password of the account named by the <code>-D</code> parameter.</p>
<code>-f filename</code>	Specifies the input LDIF file to process.

The LDIF files generated by **ldapsearch** differ from those required for input to **ldapmodify**. The **ldapmodify** input files must contain a **changetype** entry immediately after each dn entry. The **changetype** entry specifies how to use the data to change the LDAP database.

The full syntax for **changetype** within each transaction is as follows:

```
dn: distinguished-name-of-entry
changetype: keyword
subkeyword: attribute
attribute: value
changetype: keyword
subkeyword: attribute
attribute: value
.
.
.
```

Where:

- *keyword* can be **add**, **modify**, or **delete**.
- *subkeyword* can be (respectively): **add**, **replace**, or **delete**.
- *attribute* can be any LDAP attribute in the entry.
- *value* is the value to assign to the attribute.

Repeated **changetype: keyword** entries are not required within a transaction unless you change the keyword. From top to bottom within the transaction, the latest keyword applies until another **changetype: keyword** entry is provided. The following syntax is valid if the same keyword applies throughout the transaction:

```
dn: distinguished-name-of-entry
changetype: keyword
subkeyword: attribute
attribute: value
subkeyword: attribute
attribute: value
subkeyword: attribute
attribute: value
.
.
.
```

subkeyword: attribute entries are optional and indicate that you want to apply the change to a specific attribute within the entry. If no *subkeyword: attribute* entries in the transaction are found, the change applies to the entire entry. For example, it is faster to delete an entire entry:

```
dn: radiusname=TINYCO.COM,radiusclass=Proxy,o=radius
changetype: delete
```

but if you want to delete only a few attributes from the entry, you can do so:

```
dn: radiusname=TINYCO.COM,radiusclass=Proxy,o=radius
changetype: delete
delete: retry-count
-
delete: include-in-auth-list
```

If the *subkeyword* is **add** or **replace**, an *attribute: value* entry must appear immediately following the *subkeyword: attribute* entry. If the *subkeyword* is **delete**, the *attribute: value* entry does not apply and should be omitted.

The following LDIF file can be used with an **ldapmodify** command.

Figure 203: Sample LDIF File

```
dn: radiusname=BIGCO.COM,radiusclass=Proxy,o=radius
changetype: add
radiusname: BIGCO.COM
ip-address: 194.132.5.89
accounting: both
retry-count: 3
retry-timeout: 5000
shared-secret: testing123
include-in-auth-list: no

dn: radiusname=BIGGERCO.COM,radiusclass=Proxy,o=radius
changetype: modify
replace: shared-secret
shared-secret: hereistheseecret
-
replace: ip-address
ip-address: 192.7.2.121

dn: radiusname=TINYCO.COM,radiusclass=Proxy,o=radius
changetype: modify
delete: include-in-auth-list
```

NOTE: To delete the proxy entry for TINYCO.COM, issue the following command:

```
dn: radiusname=TINYCO.COM,radiusclass=Proxy,o=radius
changetype: delete
```

Importing Records from Another LDAP Database

To import entries from one LDAP database into another, run the **ldapsearch** command on the first database. Request only the attributes you want for the new database. When **ldapsearch** completes processing, edit the output LDIF file. After each line that begins with **dn:**, add a single line containing the text **changetype: add**. Once your editing is complete, run an **ldapmodify -f** command that references the new LDIF file. After the **ldapmodify** command is executed, your new database is populated with the records you extracted from the old database.

The LDIF file shown in [Figure 204 on page 519](#) is derived from the output of the **ldapsearch** command. When specified as the input to an **ldapmodify -f** command, the contents of the file are added to the target database.

Figure 204: Adding Records with an LDIF File

```
dn: radiusname=KEVIN,radiusclass=Native-User,o=radius
changetype: add
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: KEVIN
password: {x-clear}secret1
profile: ISDN
login-limit: 2

dn: radiusname=MICHAEL,radiusclass=Native-User,o=radius
changetype: add
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: MICHAEL
password: {x-clear}secret99
profile: ISDN
login-limit: 2
```

Deleting Records

The **ldapdelete** command removes records from the LDAP database. The **ldapdelete** command can be used to delete records either by a specific NAS or NAS and session start time combination. For example, to delete entries names USER1 through USER5, add the information shown in [Figure 205 on page 520](#) to a file called **deletedemo.ldf**.

Figure 205: Deleting Records with an LDIF File

```
radiusname=USER1,radiusclass=Native-User,o=radius  
radiusname=USER2,radiusclass=Native-User,o=radius  
radiusname=USER3,radiusclass=Native-User,o=radius  
radiusname=USER4,radiusclass=Native-User,o=radius  
radiusname=USER5,radiusclass=Native-User,o=radius
```

Now, pass the **deletedemo.ldf** file to the **ldapdelete** command.

```
ldapdelete -V2 -h hostname -p 667  
-D "cn=admin,o=radius" -w password -f deletedemo.ldf
```

NOTE: Verify that the **dn:** values that usually appear in these entries are not a part of the entries in your file, because they cause the command to fail.

You can use **ldapdelete** to remove records from the LDAP database without having to supply a file. For example, to delete the native user record identified as USER1, enter the following:

```
ldapdelete -V2 -h hostname -p 667  
-D "cn=admin,o=radius" -w password "radiusname=USER1,radiusclass=native-user,o=radius"
```

You can cause records to be deleted by means of the **ldapmodify** command, if the entries in the text file contain the line **changetype: delete**. Consider the sample LDIF file named **deletemodify.ldf** shown in [Figure 206 on page 521](#).

Figure 206: deletemodify.ldf Example

```
dn: radiusname=barry,radiusclass=Native-User,o=radius
changetype: delete
dn: radiusname=maurice,radiusclass=Native-User,o=radius
changetype: delete
dn: radiusname=robin,radiusclass=Native-User,o=radius
changetype: delete
```

The **deletemodify.ldf** file can be passed to the **ldapmodify** command as follows:

```
ldapmodify -V2 -h hostname -p 667 -D"cn=admin,o=radius"
-w password -f deletemodify.ldf
```



CAUTION: On some LDAP servers, an error can cause the deletion of a container without prompting for confirmation. This can, in turn, cause the entire directory server to fail.

Searching for Active Sessions

You can use the **ldapsearch** command to search for an active session and display custom CST attributes, as shown in the following example:

```
# ldapsearch -p 667 -Dcn=admin,o=radius -w radius -b
user=test,radiusstatus=sessions_by_user,o=radius objectclass=*
```

The following is a sample output:

```
# ldapsearch -p 667 -h 10.14.1.2 -D cn=admin,o=radius -w radius -b
user=5c:0a:5b:77:7d:8a,radiusstatus=sessions_by_user,o=radius generic1=*
```

```
ldap_simple_bind: Protocol error
ldap_simple_bind: additional info: version not supported
ldapsearch: the server doesn't understand LDAPv3; trying LDAPv2 instead...
version: 1
dn: unique-session-id=025d1c495037fd500000001d00000000,client=WLC2800,user=5C:
0A:5B:77:7D:8A,radiusstatus=sessions_by_user,o=radius
objectclass: top
objectclass: radiusstatus
radiusstatus: sessions_by_user
```

```

generic1: EAP-SRV
client: WLC2800
acct-session-id: SESS-1069-080844-77203-f1e48
nas-ip-address: 10.15.1.1
nas-port: 1069
nas-port-type: 19
session-start-time: 1346091650
fullname: 5c:0a:5b:77:7d:8a
called-station-id: A8-D0-E5-3C-38-40:JWO-SEC
calling-station-id: 5C-0A-5B-77-7D-8A
elapsed: 1001

```

LDIF File Examples

This section explains how to construct LDIF files that, when input to the **ldapmodify** command, add entries to the Steel-Belted Radius Carrier database.

NOTE: If the **radiusname** string contains an equals sign (=), then it must be prefixed with a double backslash (\\) in the LDIF file entry. For example:

dn: radiusname="C\\=SE,O\\=GOOGLE",radiusclass=Native-User,o=radius

Adding RADIUS Clients with LDIF

The sample LDIF entry shown in [Figure 207 on page 522](#) adds a RADIUS client named **ANNEX105** to the Steel-Belted Radius Carrier database.

Figure 207: Adding RADIUS Clients

```

dn: radiusname=ANNEX105,radiusclass=Client,o=radius
changetype: add
objectclass: top
objectclass: Client
radiusname: ANNEX105
ip-address: 193.162.45.12
product: Nortel Networks Remote Annex
shared-secret: testing123

```

The syntax in this LDIF entry is shown in [Figure 208 on page 523](#).

Figure 208: LDIF Syntax

```
dn: radiusname=String,radiusclass=Client,o=radius
changetype: add
objectclass: top
objectclass: Client
radiusname: String
ip-address: IPAddressOfTheClientDevice
product: Make&ModelChoiceFromVendor.IniFile | ...
shared-secret: SharedSecretThatWasConfiguredOnTheClientDevice
RASClientField: RASClientFieldValue
RASClientField: RASClientFieldValue
:
```

Adding Users with LDIF

The sample LDIF entry shown in [Figure 209 on page 523](#) adds a Local (Native) User, named **KEVIN** to the Steel-Belted Radius Carrier database.

Figure 209: Adding Users

```
dn: radiusname=KEVIN,radiusclass=Native-User,o=radius
changetype: add
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: KEVIN
password: {x-clear}secret1
profile: ISDN
login-limit: 2
```

The syntax in this LDIF entry is shown in [Figure 210 on page 524](#).

Figure 210: LDIF Syntax

```
dn: radiusname=String,radiusclass=Native-User |
    Solaris-User |..., o=radius
changetype: add
objectclass: top
objectclass: Native-User | Solaris-User | ...
objectclass: user
radiusname: String
password: {x-clear}PString | {x-md5}Hash |
    {x-md5}{encrypt}PString |...
profile: NameOfProfileEntryInTheServerDatabase
login-limit: IntegerGivingConcurrentConnectionLimit
UserField: UserFieldValue
UserField: UserFieldValue
:
```

The LDIF file shown in [Figure 211 on page 524](#) add a local (native) user named **CHRISTIAN**, who has various attribute/value pairs assigned to his check list and return list.

Figure 211: Adding a Native User

```
dn: radiusname=christian,radiusclass=native-user,o=radius
changetype: add
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: CHRISTIAN
password: {x-clear}password
login-limit: 2

dn: radiuslist=check,radiusname=CHRISTIAN,radiusclass=Native-User,o=radius
changetype: add
objectclass: top
objectclass: check
radiuslist: check
NAS-IP-Address: 50.50.50.50
Framed-protocol: PPP

dn: radiuslist=reply,radiusname=CHRISTIAN,radiusclass=Native-User,o=radius
changetype: add
objectclass: top
objectclass: reply
radiuslist: reply
framed-ip-address: 100.100.100.100
framed-IP-Netmask: 255.255.255.224
```

Check lists and return lists are objects in the LDAP virtual schema, but the individual RADIUS attributes are not. Therefore, you must use a separate LDIF entry for each check list and return list object, but each LDIF entry can name multiple attribute/value pairs.

To indicate that a transaction applies to the user's check list (rather than to the user entry itself), use the keyword **check** as the value for **radiuslist** and **objectclass** within the transaction. You must assign this value to **radiuslist** in the distinguished name, and again before the list of attributes. You must also assign the value to **objectclass**, above the second **radiuslist** entry.

To indicate the return list, use the keyword **reply**.

The LDIF syntax to add a user entry, complete with a check list and return list, is shown in [Figure 212 on page 525](#). The **radiusname** and **radiusclass** values for all of the transactions that apply to the same User entry must be the same.

Figure 212: Adding a User with Check List and Return List Attributes

```
dn: radiusname=String,radiusclass=Native-User | ...,o=radius
changetype: add
objectclass: top
objectclass: Native-User | Solaris-User| ...
objectclass: user
radiusname: String
password: {x-clear}PString | {x-md5}Hash | {x-md5}{encrypt}PString |...
profile: NameOfProfileEntryInTheServerDatabase
login-limit: IntegerGivingConcurrentConnectionLimit
UserField: UserFieldValue
UserField: UserFieldValue

dn: radiuslist=check,radiusname=String,radiusclass=Native-User | ...,o=radius
changetype: add
objectclass: top
objectclass: check
radiuslist: check
AttributeName: AttributeValue
AttributeName: AttributeValue
:
dn: radiuslist=reply,radiusname=String,radiusclass=
Native-User | ...,o=radius
changetype: add
objectclass: top
objectclass: reply
radiuslist: reply
AttributeName: AttributeValue
AttributeName: AttributeValue
:
```

Adding Proxy Targets with LDIF

The sample LDIF entry shown in [Figure 213 on page 526](#) adds the proxy RADIUS target **BIGCO.COM** to the Steel-Belted Radius Carrier database.

Figure 213: Adding Proxy Targets

```
dn: radiusname=BIGCO.COM,radiusclass=Proxy,o=radius
changetype: add
objectclass: top
objectclass: Proxy
radiusname: BIGCO.COM
ip-address: 194.132.5.89
accounting: both
retry-count: 3
retry-timeout: 5000
shared-secret: testing123
include-in-auth-list: no
```

The syntax in this LDIF entry is shown in [Figure 214 on page 526](#).

Figure 214: LDIF Syntax

```
dn: radiusname=StringToParseAsProxyName, radiusclass=Proxy, o=radius
changetype: add
objectclass: top
objectclass: Proxy
radiusname: StringToParseAsProxyName
ip-address: IPAddressOfTheTargetServer
accounting: Both | ...
retry-count: Integer
retry-timeout: Integer
shared-secret: SharedSecretThatWasConfiguredOnTheTargetServer
include-in-auth-list: Yes | No
ProxyField: ProxyFieldValue
ProxyField: ProxyFieldValue
:
```

Adding Tunnels with LDIF

The sample LDIF entry shown in [Figure 215 on page 527](#) adds the tunnel **ACME.COM** to the Steel-Belted Radius Carrier database.

Figure 215: Adding Tunnels

```
dn: radiusname=ACME.COM,radiusclass=Tunnel,o=radius
changetype: add
objectclass: top
objectclass: Tunnel
radiusname: ACME.COM
dnis-list: 8005551212;6171231234;12343210
description: Tunnel configuration for Acme Corp.
usage-limit: 24
```

The syntax in this LDIF entry is shown in [Figure 216 on page 527](#).

Figure 216: LDIF Syntax

```
dn: radiusname=StringToParseAsTunnelName,radiusclass=Tunnel,o=radius
changetype: add
objectclass: top
objectclass: Tunnel
radiusname: StringToParseAsTunnelName
dnis-list: PhoneNumber;PhoneNumber;etc
description: StringDescribingTunnel
usage-limit: IntegerGivingConcurrentConnectionLimit
TunnelField: TunnelFieldValue
TunnelField: TunnelFieldValue
:
```

Adding IP Address Pools with LDIF

The sample LDIF entry shown in [Figure 217 on page 527](#) adds an IP address pool named **POOL1** to the Steel-Belted Radius Carrier database.

Figure 217: Adding IP Address Pools

```
dn: radiusname=POOL1,radiusclass=IP-Addr-Pool,o=radius
changetype: add
objectclass: top
objectclass: IP-Addr-Pool
radiusname: POOL1
description: Address pool for common users
range: 198.187.100.1:50
range: 198.187.101.1:50
```

The syntax in this LDIF entry is shown in [Figure 218 on page 528](#).

Figure 218: LDIF Syntax

```
dn: radiusname=String,radiusclass=IP-Addr-Pool,o=radius
changetype: add
objectclass: top
objectclass: IP-Addr-Pool
radiusname: String
description: StringDescribingPool
range: IPAddress:Range
range: IPAddress:Range
:
```

Configuring a RADIUS Server with LDIF

The sample LDIF entry shown in [Figure 219 on page 528](#) lets you configure your Steel-Belted Radius Carrier server by adding the Native User authentication method and defining conventions for tunnel name parsing.

Figure 219: Adding a RADIUS Server

```
dn: radiusclass=Server, o=radius
changetype: add
objectclass: top
objectclass: RadiusClass
radiusclass: Server
auth-methods: Native User
tunnel-delimiter: $
tunnel-type: prefix
```

The syntax in this LDIF entry is shown in [Figure 220 on page 528](#).

Figure 220: LDIF Syntax

```
dn: radiusclass=Server, o=radius
changetype: add
objectclass: top
objectclass: RadiusClass
radiusclass: Server
auth-methods: Native User | Solaris User | SecurID Prefix | ...
tunnel-delimiter: Character
tunnel-type: Prefix | Suffix | Neither
ConfigurationField: ConfigurationFieldValue
ConfigurationField: ConfigurationFieldValue
:
```

Statistics Variables

Server statistics counters record the number of certain types of events. The LCI allows you to read these statistics to monitor the performance of your Steel-Belted Radius Carrier server.

Counter Statistics

The statistics counters can be accessed via the LCI by executing the following one line command:

```
ldapsearch -V 2 -h 127.0.0.1 -p 667 -D "cn=admin,o=radius" -w radius -s sub -T -b "radiusstatus=statistics,o=radius"
stattype=typeofstatus
```

The following sections illustrate the variables displayed for different settings of the `stattype` parameter.

stattype: server

```
dn: stattype=server,radiusstatus=statistics,o=radius
objectclass: top
objectclass: radiusstatus
radiusstatus: statistics
stattype: server
start-time: 2002/05/08 13:29:08
up-time: 26188
ip-address: 192.168.21.142
version: v 2.20.33
authentication-threads: 0
accounting-threads: 0
total-threads: 0
max-auth-threads: 100
max-acct-threads: 100
max-total-threads: 200
high-auth-threads: 2
high-acct-threads: 0
high-total-threads: 2
```

stattype: authentication

```
dn: stattype=authentication,radiusstatus=statistics,o=radius
objectclass: top
objectclass: radiusstatus
radiusstatus: statistics
stattype: authentication
accept: 1
```

```

reject: 0
silent-discard: 0
total-transactions: 8
invalid-request: 0
failed-authentication: 0
failed-on-check-list: 0
insufficient-resources: 0
proxy-failure: 0
rejected-by-proxy: 0
transactions-retried: 0
total-retry-packets: 0

```

stattype: accounting

```

dn: stattype=accounting,radiusstatus=statistics,o=radius
objectclass: top
objectclass: radiusstatus
radiusstatus: statistics
stattype: accounting
start: 0
stop: 0
on: 0
off: 0
total-transactions: 0
invalid-request: 0
invalid-client: 0
invalid-shared-secret: 0
insufficient-resources: 0
proxy-failure: 0
transactions-retried: 0
total-retry-packets: 0

```

stattype: proxy

```

dn: stattype=proxy,radiusstatus=statistics,o=radius
objectclass: top
objectclass: radiusstatus
radiusstatus: statistics
stattype: proxy
authentication: 0
accounting: 0
total-transactions: 0
timed-out: 0
invalid-response: 0

```



```
invalid-shared-secret: 0
insufficient-resources: 0
transactions-retried: 0
total-retry-packets: 0
proxytransaction: 0
```

Rate Statistics

Rate statistics are derived from existing counter statistics by taking time into consideration. Overall server rate values calculated for each of these counter statistics consist of the following:

- **Current rate**—The rate measured over the most recent rate interval.
- **Average rate**—The rate measured since the Steel-Belted Radius Carrier server was started or since the last time statistics were reset to zero.
- **Peak rate**—The highest rate observed since the Steel-Belted Radius Carrier server was started or since the last time statistics were reset to zero.

Rate values calculated per NAD client and per Called-Station-ID consist of only current and peak rates.

NOTE: NAD client and Called-Station-ID specific rate statistics are calculated only if you set the **EnhancedRateStats** parameter in the **radius.ini** file to 1.

Additionally, there is a (read-only) time value used in calculations:

- Rate statistics reports the current and peak rate statistics handled by SBR at the time of query.

To read overall server rate statistics from the LCI, you must set **radiusstatus= statistics** and **stattype= rate**. This results in output such as the following:

```
rate-statistics-seconds-per-interval: 1
auth-request-current-rate: 0
auth-request-average-rate: 0
auth-request-peak-rate: 7
auth-accept-current-rate: 0
auth-accept-average-rate: 0
auth-accept-peak-rate: 1
auth-reject-current-rate: 0
auth-reject-average-rate: 0
auth-reject-peak-rate: 0
acct-start-current-rate: 0
acct-start-average-rate: 0
```

```

acct-start-peak-rate: 0
acct-stop-current-rate: 0
acct-stop-average-rate: 0
acct-stop-peak-rate: 0
proxy-auth-request-current-rate: 0
proxy-auth-request-average-rate: 0
proxy-auth-request-peak-rate: 0
proxy-acct-request-current-rate: 0
proxy-acct-request-average-rate: 0
proxy-acct-request-peak-rate: 0
proxy-fail-timeout-current-rate: 0
proxy-fail-timeout-average-rate: 0
proxy-fail-timeout-peak-rate: 0
proxy-fail-badresp-current-rate: 0
proxy-fail-badresp-average-rate: 0
proxy-fail-badresp-peak-rate: 0
proxy-fail-badsecret-current-rate: 0
proxy-fail-badsecret-average-rate: 0
proxy-fail-badsecret-peak-rate: 0
proxy-fail-missingresr-current-rate: 0
proxy-fail-missingresr-average-rate: 0
proxy-fail-missingresr-peak-rate: 0
proxy-retries-current-rate: 0
proxy-retries-average-rate: 0
proxy-retries-peak-rate: 0
proxy-auth-rej-proxy-current-rate: 0
proxy-auth-rej-proxy-average-rate: 0
proxy-auth-rej-proxy-peak-rate: 0
proxy-acct-fail-proxy-current-rate: 0
proxy-acct-fail-proxy-average-rate: 0
proxy-acct-fail-proxy-peak-rate: 0
proxy-auth-rej-proxy-error-current-rate: 0
proxy-auth-rej-proxy-error-average-rate: 0
proxy-auth-rej-proxy-error-peak-rate: 0
proxy-transaction-current-rate: 0
proxy-transaction-average-rate: 0
proxy-transaction-peak-rate: 0

```

To read rate statistics per NAD client from the LCI, you must set **radiusstatus=rate_stats_by_nas** and also input the NAD name for which rate statistics should be displayed. This results in output such as the following:

```

dn: nasname=TEST-NAS,radiusstatus=rate_stats_by_nas,o=radius
objectclass: top

```

```

objectclass: radiusstatus
radiusstatus: rate_stats_by_nas
nasname: TEST-NAS
nasipaddr: 10.212.98.76
auth-request-current-rate: 101
auth-accept-current-rate: 101
auth-reject-current-rate: 0
auth-dropped-current-rate: 0
acct-total-requests-current-rate: 102
acct-start-current-rate: 102
acct-stop-current-rate: 0
acct-interim-current-rate: 0
acct-dropped-current-rate: 0
proxy-auth-request-current-rate: 0
proxy-acct-request-current-rate: 0
proxy-total-request-current-rate: 0
proxy-fail-timeout-current-rate: 0
auth-request-peak-rate: 101
auth-accept-peak-rate: 101
auth-reject-peak-rate: 0
auth-dropped-peak-rate: 0
acct-total-requests-peak-rate: 102
acct-start-peak-rate: 102
acct-stop-peak-rate: 0
acct-interim-peak-rate: 0
acct-dropped-peak-rate: 0
proxy-auth-request-peak-rate: 0
proxy-acct-request-peak-rate: 0
proxy-total-request-peak-rate: 0
proxy-fail-timeout-peak-rate: 0

```

To read rate statistics per Called-Station-ID from the LCI, you must set **radiusstatus=stats_by_calledstationid** and also input the string representing Called-Station-ID for which rate statistics should be displayed. This results in output such as the following:

```

dn: called-station-id=00-10-A4-23-19-C0,radiusstatus=stats_by_calledstationid,o=radius
objectclass: top
objectclass: radiusstatus
radiusstatus: stats_by_calledstationid
called-station-id: 00-10-A4-23-19-C0
auth-request-current-rate: 101
auth-accept-current-rate: 101
auth-reject-current-rate: 0
auth-dropped-current-rate: 0

```

acct-total-requests-current-rate: 102
acct-start-current-rate: 102
acct-stop-current-rate: 0
acct-interim-current-rate: 0
acct-dropped-current-rate: 0
proxy-auth-request-current-rate: 0
proxy-acct-request-current-rate: 0
proxy-total-request-current-rate: 0
proxy-fail-timeout-current-rate: 0
auth-request-peak-rate: 101
auth-accept-peak-rate: 101
auth-reject-peak-rate: 0
auth-dropped-peak-rate: 0
acct-total-requests-peak-rate: 102
acct-start-peak-rate: 102
acct-stop-peak-rate: 0
acct-interim-peak-rate: 0
acct-dropped-peak-rate: 0
proxy-auth-request-peak-rate: 0
proxy-acct-request-peak-rate: 0
proxy-total-request-peak-rate: 0
proxy-fail-timeout-peak-rate: 0

7

PART

Optional Authentication Modules

[SIM Authentication Module | 536](#)

[Summary of Configuration Tasks for the SIM Authentication Module | 550](#)

[SIM Authentication Module Configuration with a SIGHUP \(1\) Signal | 553](#)

[Overview of the WiMAX Mobility Module | 554](#)

[Configuring the WiMAX Mobility Module | 580](#)

SIM Authentication Module

IN THIS CHAPTER

- [SIM Authentication Module Component Overview | 536](#)
- [Operation Overview | 538](#)
- [SIM Authentication Module Configuration | 543](#)
- [Special Attribute Handling Features | 543](#)
- [Kineto S1 Support | 549](#)

SIM Authentication Module Component Overview

The optional SIM authentication module enables Global System for Mobile Communications (GSM) and Universal Mobile Telecommunications System (UMTS) service providers to offer wireless network access to subscribers through hotspot concession operators while leveraging existing customer care, roaming, and billing infrastructures. This section describes the hardware and software components that make up the SIM authentication module.

NOTE: Because Signalware is not supported on Solaris 11.3, the SIM authentication module cannot use the SIGTRAN protocol to communicate with an HLR to process RADIUS requests. However, the SIM authentication module can convert RADIUS requests to Diameter requests to communicate with an HSS.

SIMAuth

The SIM authentication module performs EAP-SIM and EAP-AKA authentication using a software module called *SIMAuth*. SIMAuth manages all EAP-SIM-based and EAP-AKA based authentication requests from subscribers.

- EAP-SIM authentication is used with older SIM cards.

- EAP-AKA authentication is used with third-generation USIM (Universal Subscriber Identity Module) cards.

SIMAuth supports user authentication based on SIM IMSI (International Mobile Subscriber Identity) values or supports anonymous authentication based on pseudonym values that are assigned after the first successful authentication.

Signalware SIGTRAN Protocol Stacks

The Signalware SIGTRAN protocol stack handles the various SS7 protocol layers to put Mobile Access Part (MAP) requests onto the SS7 network. ANSI SS7, CCITT/ITU SS7, Japanese, and Chinese networks are supported. Signalware provides SS7 signaling over IP networks.

The Signalware SIGTRAN protocol stack is used in conjunction with the SIM Authentication module to provide gateway functionality, which enables Steel-Belted Radius Carrier to pass MAP requests to the SS7 network. MAP requests are passed over the SS7 network to the Home Location Register (HLR), which is the primary subscriber database in the Global System for Mobile Communications (GSM) network. The HLR then performs a database lookup and returns the requested authentication or authorization information to Steel-Belted Radius Carrier.

MAP Gateway (authGateway) Application

The MAP gateway or *authGateway* application acts as a link between Steel-Belted Radius Carrier and the SS7 network. It formats and transmits MAP requests to the HLR over Signalware. The MAP gateway processes requests for authentication and authorization information. Multiple *authGateway* instances can be used with the *GWrelay* application to process multiple authentication and authorization requests at the same time.

NOTE: SBR Carrier supports up to 256 *authGateway* instances. However, we recommend that you use a maximum of 100 *authGateway* instances.

GWrelay Application

The *GWrelay* application is used to pass authentication requests between SBR Carrier and the *authGateway* instances in a round-robin method. The *GWrelay* application establishes an SCTP connection with each *authGateway* instance through unique source and destination ports, but does not track any EAP-SIM/AKA requests.

CDR Accounting

CDR capability manages all CDR-based subscriber accounting for the purposes of billing. CDRs are forwarded through an FTP server or other transport method to a billing application.

NOTE: Class attributes need to be included in accounting requests in order for CDR accounting to work properly.

Data Accessors

Data accessors enable the SIM authentication module to query an external SQL database or LDAP directory server for WLAN authorization and IMSI/MSISDN lookups. Data accessors can be used to supplement or replace the interaction between the SIM authentication module and a service provider's HLR.

You must install an Oracle 10, 11, or 12 client or JDBC (Java Database Connectivity) if you want to use the SQL data accessor.

Operation Overview

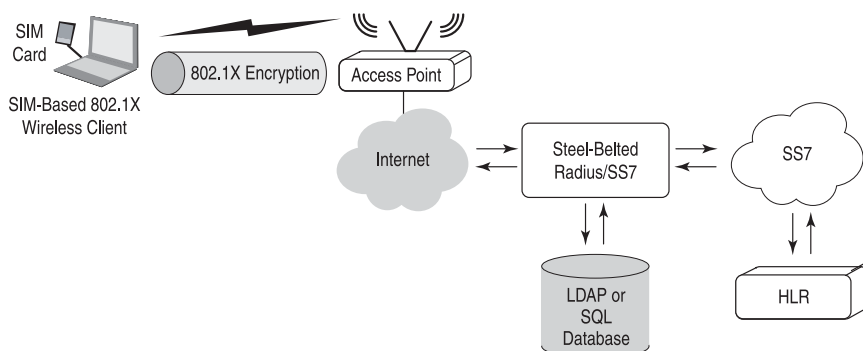
The optional SIM authentication module or SIMauth enables you to provide IP-based services such as secure public WLAN access, and Unlicensed Mobile Alliance (UMA) and Femtocell access to your subscribers, while leveraging your existing customer care and authentication infrastructure. Appropriate for Global System for Mobile Communications (GSM) infrastructures, the optional SIM authentication module provides AAA services for 802.1X and non-802.1X hotspots and Unlicensed Mobile Access (UMA) networks, enabling several types of service offerings. You can offer secure hotspot access via 802.1X and Extensible Authentication Protocol–Subscriber Identity Module (EAP-SIM) or Extensible Authentication Protocol–Authentication and Key Agreement (EAP-AKA) user authentication. The SIM authentication module extends mobile services over IP access networks for UMA environments, providing the same mobile identity on unlicensed wireless networks as on mobile networks, and enabling roaming and handover between networks.

The SIM authentication module offers secure public WLAN access. Subscriber data is transmitted securely over the wireless link. Data security is ensured with either the Wi-Fi Protected Access (WPA) or the dynamic Wired Equivalent Privacy (WEP) encryption protocol to prevent wireless eavesdropping within the hotspot.

[Figure 221 on page 539](#) shows the typical network infrastructure for using the SIM authentication module in Steel-Belted Radius Carrier. The SIM authentication module serves as the link between the IP network and the SS7 network, verifying SIM-based or USIM-based credentials provided by a subscriber against an

existing provider Home Locator Register (HLR), SQL database, or LDAP directory, to authenticate the subscriber and obtain authorization information.

Figure 221: SIM Authentication Module Support in Steel-Belted Radius Carrier



The SIM authentication module can also generate accounting data on mobile subscriber activities that can be used by Call Detail Record (CDR) accounting.

To access a public hotspot, the subscriber's wireless laptop, smart phone, tablet, or personal digital assistant (PDA) must include an IEEE 802.1X supplicant (client application) that supports EAP-SIM or EAP-AKA. The SIM authentication module uses the service provider's SS7 infrastructure to facilitate secure 802.1X-based subscriber authentication and billing. For SIM authentication, a service provider's 802.1X access point directs authentication requests from user devices to the Steel-Belted Radius Carrier server.

SIMAuth verifies SIM-based or USIM-based credentials provided by a subscriber against an existing provider HLR, SQL database, or LDAP directory, to authenticate the subscriber and obtain authorization information.

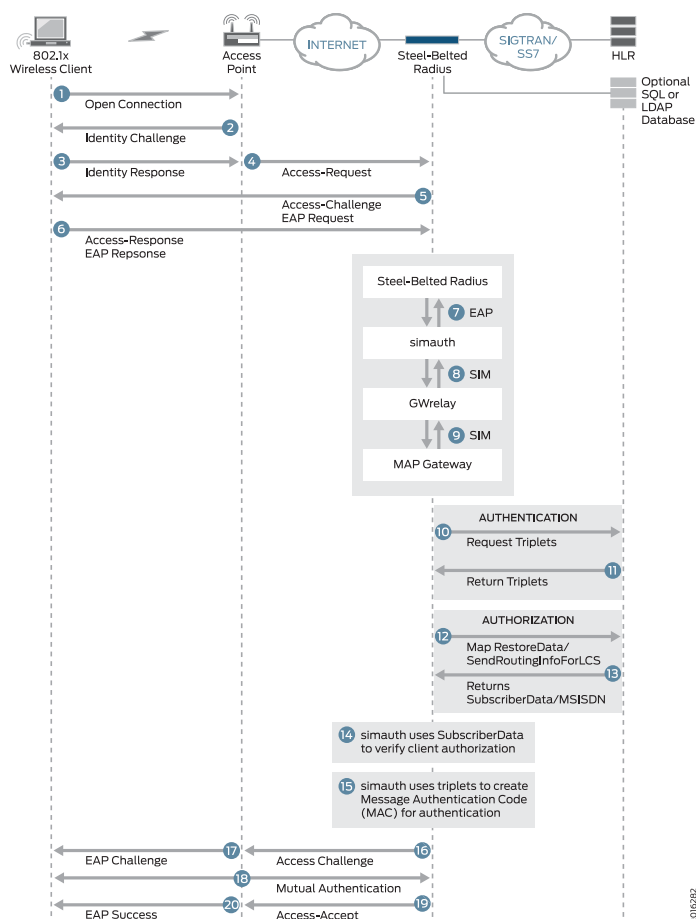
Steel-Belted Radius Carrier supports EAP-SIM/EAP-AKA fast reauthentication. As a result, your server can regularly replace the encryption keys used over the wireless link to provide identity protection to subscribers. Reauthentication occurs securely without requiring interaction between Steel-Belted Radius Carrier and the HLR, thereby decreasing traffic load on the RADIUS server.

To perform an EAP-SIM or EAP-AKA authentication, the wireless access point must have an EAP-capable 802.1X supplicant. EAP-SIM can be used with HLRs that support MAP application context version 2 and EAP-AKA can be used with HLRs that support version 3.

SIM Card-Based Authentication

[Figure 222 on page 540](#) shows the SIM-based provider/subscriber solution supported by Steel-Belted Radius Carrier.

Figure 222: Authentication in Steel-Belted Radius Carrier with SIM Authentication Module



The following sequence takes place when a wireless client with a SIM card installed requests access to an IP network protected by Steel-Belted Radius Carrier:

1. A user running an EAP-SIM-compatible 802.1X supplicant opens a wireless connection to an 802.1X access point at a WLAN hotspot.
2. The 802.1X-configured access point challenges the supplicant for its identity.
3. The 802.1X supplicant on the wireless client responds with an EAP-SIM-based identity.
4. The access point forwards this information to Steel-Belted Radius Carrier (directly or through a proxy RADIUS server).
5. Steel-Belted Radius Carrier sends an Access-Challenge request for EAP-SIM authentication to the access point, which forwards the request to the wireless client.
6. The wireless client receives the EAP-SIM request, and, through the access point, agrees to start EAP-SIM by sending an EAP-SIM response. Included in the wireless client response is a *nonce* (large random number) that protects against playback attacks.

7. The Steel-Belted Radius Carrier software passes the EAP information to the SIMauth module.
8. The SIMauth module converts the EAP information to SIM requests and passes them to the GWrelay application.
9. The GWrelay application passes the requests to the MAP gateway instances in a round-robin method.
10. The MAP gateway instances pass requests for information that they need to perform the authentication (called *triplets*) through Signalware to the HLR.
11. The HLR does a database lookup and returns the requested triplets to the SIMauth module.
12. If Steel-Belted Radius Carrier is configured to support user authorization, the SIMauth module sends a request for authorization information to an external (LDAP or SQL) database or through Signalware to the HLR.
13. The external database or HLR returns the requested authorization information for the user.
14. The SIMauth module uses subscriber data to verify authorization.
15. The SIMauth module uses these triplets to create the message authentication code (MAC) that it sends as part of its challenge to the wireless client.
16. Steel-Belted Radius Carrier sends an Access-Challenge containing the MAC to the access point.
17. The access point forwards the challenge to the wireless client.
18. The wireless client verifies that the message is authentic (by running the appropriate authentication algorithms) and responds by sending its own message authentication code to Steel-Belted Radius Carrier.
19. The Steel-Belted Radius Carrier server verifies the client message authentication code, and sends an Access-Accept response to the wireless access point. The Access-Accept includes keying material for encrypting data sent on the wireless connection.
20. The wireless access point uses the keying material from the Access-Accept to establish an encrypted session with the wireless client.

Authentication using a USIM card and EAP-AKA is similar, although the authentication information retrieved from the HLR is different.

EAP-SIM/EAP-AKA Authorization/Service Delivery

Steel-Belted Radius Carrier server enables you to configure subscriber connections according to authorization strings that you can configure in the subscriber database or HLR. You can map one or more authorization strings to a profile that you configure in Steel-Belted Radius Carrier. This profile is applied to the user connection.

EAP-SIM/EAP-AKA Identities

To provide identity protection and fast reauthentication, the EAP-SIM/EAP-AKA protocol includes three types of identities for the client. For any given authentication between the client and server, only one of these three is required:

- The *Permanent Identity* is the identity required by the EAP-SIM/EAP-AKA server to retrieve the authentication information (triplets) and authorization information from the external database or HLR. The Permanent Identity must contain the IMSI from the SIM card being used.
- The *Pseudonym Identity* is used in place of the Permanent Identity whenever it is available. The EAP-SIM/EAP-AKA server (Steel-Belted Radius Carrier server) creates the pseudonym, and sends it to the client on the first authentication. A new pseudonym is created every time that authentication occurs. Steel-Belted Radius Carrier server uses an encrypted form of the IMSI as the pseudonym so that pseudonyms can be shared among all Steel-Belted Radius Carrier servers that share the same encryption key. The Pseudonym Identity provides identity protection by hiding the Permanent Identity (the IMSI) on the second and all future authentications.
- The *Reauthentication Identity* also provides identity protection. Like the pseudonym, a new Reauthentication Identity is created by the server on each authentication. However, the purpose of using the Reauthentication Identity is to begin a fast reauthentication exchange. During this exchange in EAP-SIM or EAP-AKA, new key material is generated based on the current authentication information. The device's SIM card and the HLR do not need to participate in this exchange. This feature is used to regularly replace the encryption keys used over the wireless link.

You can disable the Pseudonym and Reauthentication identities. For more details, see the section on configuring EAP-SIM and EAP-AKA for the SIM authentication module in the *SBR Carrier Reference Guide*.

Any of these identities can also contain a **@realm** tag to form a Network Address Identifier (NAI). The wireless access point typically uses the EAP identity as the RADIUS username for communicating to the RADIUS server. In this case, the RADIUS server or RADIUS proxy server can use the realm to direct the RADIUS request to another server or use the realm in another way.

The EAP-SIM and EAP-AKA protocols enable the server to provide a realm when it creates a Reauthentication Identity. Steel-Belted Radius Carrier server typically uses the realm that was received with the last Permanent Identity or Pseudonym Identity for the realm returned with the new Reauthentication Identity. You can configure a different realm to be returned in the **simauth.aut** configuration file.

EAP-SIM/EAP-AKA Fast Reauthentication

You can configure the SIM authentication module for fast reauthentication, so that after a customer establishes a secure 802.1X network connection through the authentication and authorization processes, the server replaces the encryption keys used over the wireless link. Because fast reauthentication does

not require interaction between Steel-Belted Radius Carrier SIM authentication module and the HLR, this added security measure does not affect traffic loads on the network.

SIM Authentication Module Configuration

The SIM authentication module is configured by defining parameters in a number of Steel-Belted Radius Carrier configuration files. For complete details on configuring the SIM authentication see the *SBR Carrier Reference Guide*.

Special Attribute Handling Features

Two special attribute handling features are available to use with the authentication modules:

- [“Assigning IP Addresses Based on Access Point Name \(APN\)” on page 543](#)
- [“Adding Attributes to an Access-Accept” on page 544](#)

Assigning IP Addresses Based on Access Point Name (APN)

This feature enables Steel-Belted Radius Carrier to assign IP addresses to mobile devices based on the access point name (APN).

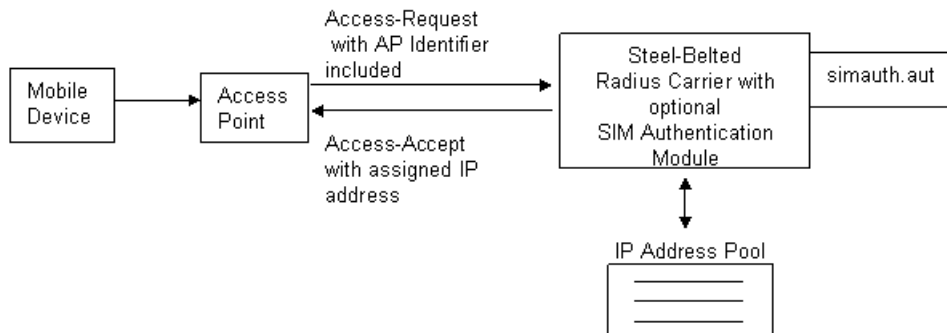
Overview

APN-based IP address assignment enables Steel-Belted Radius Carrier to perform the task of address assignment, rather than requiring the access point to assign addresses.

This feature works by configuring IP address pools, each of which consists of a set of IP addresses that can be assigned to a mobile device. You then configure an access point name (APN) to be associated with a particular pool. When an Access-Request is received, Steel-Belted Radius Carrier selects an IP address from the pool that is assigned to the APN handling the request.

[Figure 223 on page 544](#) shows the configuration of IP address assignment based on APN.

Figure 223: IP Address Assignment Based on Access Point Name



NOTE: APN-based IP address assignment takes precedence over all other methods of IP address assignment except when an IP address (or pool name) is added to an Access-Accept as the value of the Framed-IP-Address attribute.

For information about how to add any attribute from a subscriber database (such as a SQL database), see [“Adding Attributes to an Access-Accept” on page 544](#). For example, you can retrieve the IP address from a SQL database and include it as the value of Framed-IP-Address in an Access-Accept.

Configuration Tasks for Assigning IP Address Based on Access Point Name

Assigning IP addresses based on access point name involves the following main tasks:

- Configure simauth.aut. See the chapter on *Configuring the Special Attribute Handling Features for Use With the SIM Authentication Module* in the *SBR Carrier Reference Guide*.
- Create an address pool. See [“Administering Address Pools” on page 182](#).

Adding Attributes to an Access-Accept

This feature enables you to add attribute values retrieved from an external subscriber database to an Access-Accept message. For example, you might want to include the subscriber’s level of service in the Access-Accept as the value of the attribute Reply-Message. Another example might be retrieving the IP address to be assigned to a mobile node and returning it in the Access-Accept as the value of the attribute Framed-IP-Address.

Overview

An Access-Accept can include attribute values. Two authentication plug-ins are used to accomplish the tasks of authentication and adding attributes to an Access-Accept. The authentication plug-ins are:

- SIMAuth (acting as an *EAP helper*)
This authenticator provides EAP authentication.

- *Helped* authenticator (for example: **radsql.aut**)

This authenticator accesses the database, retrieves the specified attributes, and attaches them to the Access-Accept. In this situation, the *helped* authenticator does not perform any authentication tasks and its password-checking is suppressed. All authentication is performed by **SIMAuth.aut**, the EAP helper.

Data Flow

Authentication of the Access-Request and the addition of attributes to the Access-Accept is handled according to the following flow of data:

1. The mobile device sends an Access-Request to Steel-Belted Radius Carrier.
2. SIMAuth manages the EAP negotiation (challenge and response).
3. If SIMAuth authenticates the request, it attaches the IMSI and MSISDN of the mobile device, and sends the request to **radsql.aut**.
4. **radsql.aut** can use the IMSI or MSISDN as a key to query the database and request attribute values (as a separate step from the SIMAuth authentication).
5. *Helped* authenticator (for example the SQL plug-in: **radsql.aut**) returns the Access-Accept with attribute values attached.

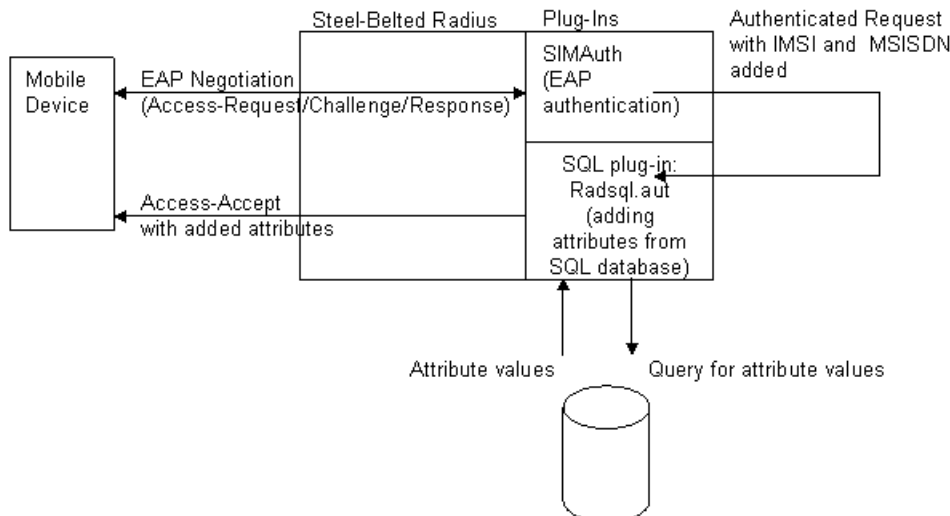
NOTE: SIMAuth is known as a Steel-Belted Radius Carrier *EAP helper* because it performs the EAP authentication for the *helped* authentication method (in this case, the SQL plug-in: **radsql.aut**). Although **radsql.aut** is usually used for authentication, in this case, it accesses the subscriber database, retrieves attributes, and returns them with the Access-Accept.

For complete information about EAP helpers, see [“Automatic EAP Helpers” on page 254](#).

[Figure 224 on page 546](#) shows an example data flow in which Steel-Belted Radius Carrier, SIMAuth, and **radsql.aut** work together to perform the following tasks:

- Access authentication (performed by SIMAuth)
- Addition of MSISDN and IMSI to the request (performed by SIMAuth)
- Database access and attribute retrieval (performed by the SQL plug-in: **radsql.aut**)
- Addition of retrieved attributes to the Access-Accept (performed by **radsql.aut**)

Figure 224: Example Data Flow for Addition of Attribute to Access-Accept



Configuration Tasks for Adding Attributes to Access-Accept

To add attributes to the Access-Accept, you need to perform the following main tasks:

- Configure the related files, listed in [“Files to Configure for Adding Attributes to Access-Accept”](#) on [page 546](#).
- Activate authentication, as described in [“Activating the Authentication Method”](#) on [page 546](#).

Files to Configure for Adding Attributes to Access-Accept

The following files require special configuration for the addition of attributes to the Access-Accept:

- **simauth.aut**
- **radsql.aut**, **radsqljdbc.aut**, or **ldapauth.aut**
- **eap.ini**

Refer to the chapter on *Configuring the Special Attribute Handling Features for Use With the SIM Authentication Module* in the *SBR Carrier Reference Guide*.

Activating the Authentication Method

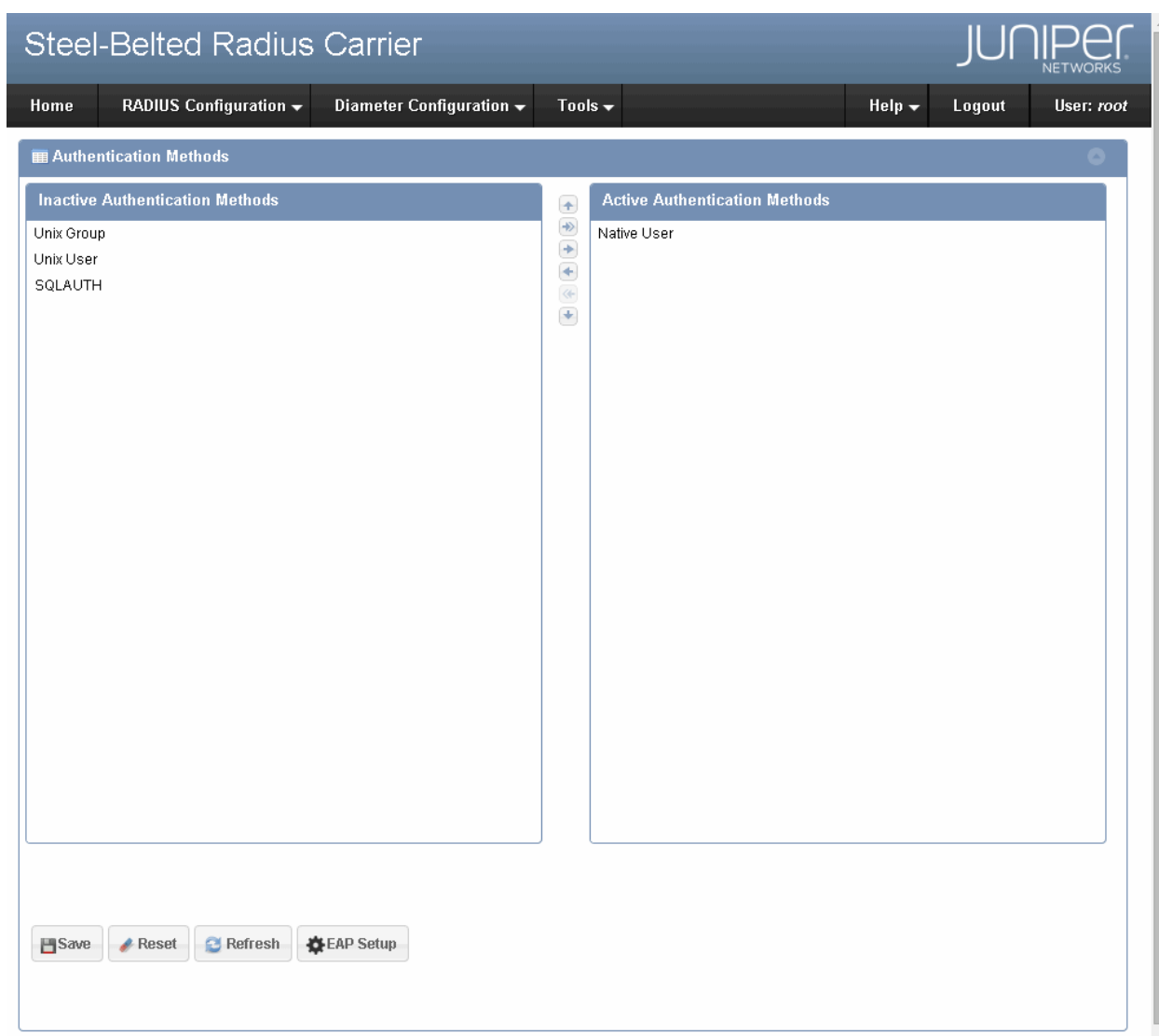
After you have configured the files described in the chapter on *Configuring the Special Attribute Handling Features for Use With the SIM Authentication Module* in the *SBR Carrier Reference Guide*, you need to activate the *helped* authentication method.

To activate the helped authentication using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > Order of Methods**. The **Authentication Methods** page ([Figure 225 on page 547](#)) appears.
2. Select the helped authentication method and click the **Right** arrow to place the authentication method in the **Active Authentication Methods** area.

In this case, the helped authentication method is named **SQLAUTH**, as shown in the **Inactive Authentication Methods** area ([Figure 225 on page 547](#)).

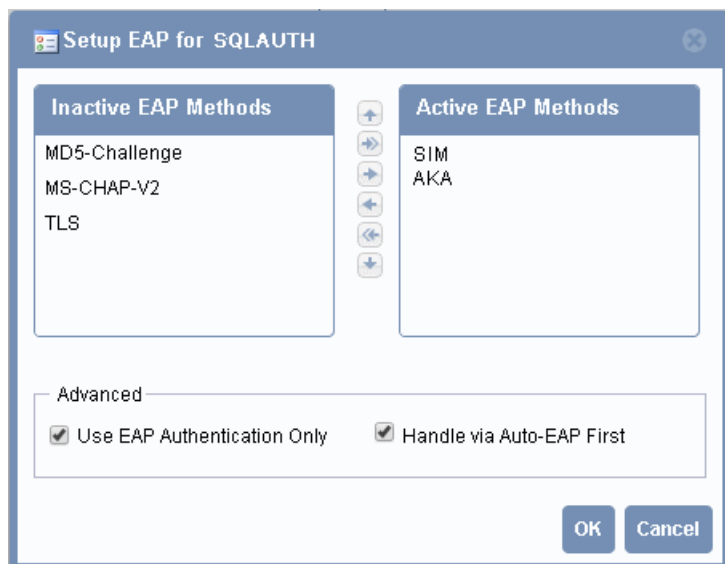
Figure 225: Activate the Helped Authentication Method



The name of the helped authentication method, or EAP helper, is user-defined. This name is specified in section on *Configuring Files for Adding Attributes to Access-Accept* in the *SBR Carrier Reference Guide*.

- After the helped authentication method has been moved to the **Active Authentication Methods** area, select the method and click **EAP Setup**. The **Setup EAP for SQLAUTH** dialog box (Figure 226 on page 548) appears.

Figure 226: Setup EAP for SQLAUTH Dialog



- Ensure that both the **Use EAP Authentication Only** and **Handle via Auto-EAP First** check boxes are selected.
- Ensure that both **SIM** and **AKA** are listed under the **Active EAP Methods** area.

NOTE: If you completed the steps correctly in the section *Configuring Files for Adding Attributes to Access-Accept* in the *SBR Carrier Reference Guide*, both Steps 4 and 5 are already completed.

- Click **OK**.
- Optionally, change the order of the authentication methods so the helped authentication method is first. Disable authentication methods that are no longer applicable. Refer to [“Order of Authentication Methods” on page 233](#) and [“Enabling EAP Methods” on page 235](#).

Kineto S1 Support

The Kineto INC (IP Network Controller) is the component of the Kineto UMA network Controller (UNC) that manages subscriber access to voice and data mobile services. The Kineto INC-AAA (S1) interface Protocol Specification defines the interface requirements for communication between an AAA server and the Kineto INC. Steel-Belted Radius Carrier complies with the requirements of the AAA server.

Steel-Belted Radius Carrier meets the Kineto S1 PS000021 specification.

For complete details on configuring the Steel-Belted Radius Carrier authentication modules to interface with the Kineto S1, see the section on *Interfacing with a Kineto IP Network Controller* in the *SBR Carrier Reference Guide*.

Summary of Configuration Tasks for the SIM Authentication Module

This chapter provides a summary of the mandatory and optional tasks associated with configuring the SIM authentication module.

[Table 63 on page 550](#) summarizes the configuration tasks associated with the SIM authentication module.

Summary of Configuration Tasks for the SIM Authentication Module

[Table 63 on page 550](#) summarizes the configuration tasks associated with the SIM authentication module.

Table 63: Summary of Configuration Tasks for SIM Authentication Module

Task	Location of Detailed Information
Install and start Signalware	<i>SBR Carrier Installation Guide</i>

Table 63: Summary of Configuration Tasks for SIM Authentication Module (continued)

Task	Location of Detailed Information
<p>Configure the SS7/IP network communication.</p> <p>This is a multi-step process that includes:</p> <ul style="list-style-type: none"> • Configuring the Signalware SIGTRAN protocol stack • Configuring the MAP gateway application, which acts as a link between Steel-Belted Radius Carrier and the SS7 network by formatting and transmitting Mobile Access Part (MAP) requests to the Home Location Register (HLR) over the SS7 network. This configuration process includes: <ul style="list-style-type: none"> • Configuring the authGateway routing location information • Configuring the authGateway.conf File. • Configuring the authGateway startup information. • Configuring the GWrelay.conf file • Starting the GWrelay process • Configuring the ulcmmg.conf File. • Loading the MML configuration settings 	<p><i>SBR Carrier Installation Guide</i></p>
<p>Define the settings for authenticating Access-Request messages by configuring the gsmmap.gen file.</p>	<p>See "gsmmap.gen File" section in the <i>SBR Carrier Reference Guide</i>.</p>
<p>Configure the EAP authentication methods you want to use.</p>	<p>See "SIM/AKA Authentication" chapter in the <i>SBR Carrier Reference Guide</i>.</p>
<p>Optionally, configure Call Detail Record (CDR) accounting (cdracct.acc file).</p>	<p>See "CDR Accounting Plug-Ins" chapter in the <i>SBR Carrier Reference Guide</i>.</p>
<p>Optionally, if you are using an LDAP directory or SQL database to store subscriber credentials, configure the data accessors.</p>	<p>See "SQL Plug-Ins" and "LDAP Plug-Ins" chapters and "Using SQL Accessors" and "Using LDAP Accessors" sections in the <i>SBR Carrier Reference Guide</i>.</p>

Table 63: Summary of Configuration Tasks for SIM Authentication Module (continued)

Task	Location of Detailed Information
Optionally configure the special attribute handling features.	See “Assigning IP Addresses Based on APN” and “Adding Attributes to an Access-Accept” sections in the <i>SBR Carrier Reference Guide</i> .
Optionally, if you are interfacing with a Kineto UMA Network Controller (UNC), configure the SIM authentication module to properly handling the Kineto attributes.	See "Interfacing with a Kineto IP Network Controller" section in the <i>SBR Carrier Reference Guide</i> .

SIM Authentication Module Configuration with a SIGHUP (1) Signal

Some configuration settings in Steel-Belted Radius Carrier are reloaded every time that Steel-Belted Radius Carrier receives a SIGHUP (1) signal. Other settings are loaded only when you stop and restart the Steel-Belted Radius Carrier server.

See the *SBR Carrier Reference Guide* for more information.

Overview of the WiMAX Mobility Module

IN THIS CHAPTER

- Supported Features of the WiMAX Mobility Module | 554
- WiMAX Network Reference Model | 555
- AAA-Generated Cryptographic Keys | 559
- EAP Authentication Methods and EAP-Derived Cryptographic Keys | 560
- WiMAX Vendor Specific Attribute (VSA) Format | 562
- WiMAX Capabilities Negotiation | 565
- Home Agent and DHCP Server Assignment | 567
- WiMAX Post-Paid (Offline) Accounting | 568
- WiMAX Prepaid Accounting | 570
- Categorizing Access-Requests from Different Devices | 578

Supported Features of the WiMAX Mobility Module

The WiMAX mobility module is an optional, license-enabled software module available for Steel-Belted Radius Carrier. Steel-Belted Radius Carrier fulfills all of the AAA requirements of WiMAX, with its support for multiple EAP methods, robust Mobile IP session management capabilities, and cryptographic key management. Steel-Belted Radius Carrier supports IEEE standards 802.16e-2005 for mobile WiMAX, and the 802.16-2004 for fixed WiMAX, as well as most features of revisions 1.0–1.3 of the WiMAX Forum Networking Group (NWG) specification.

The WiMAX mobility module of Steel-Belted Radius Carrier supports the following features:

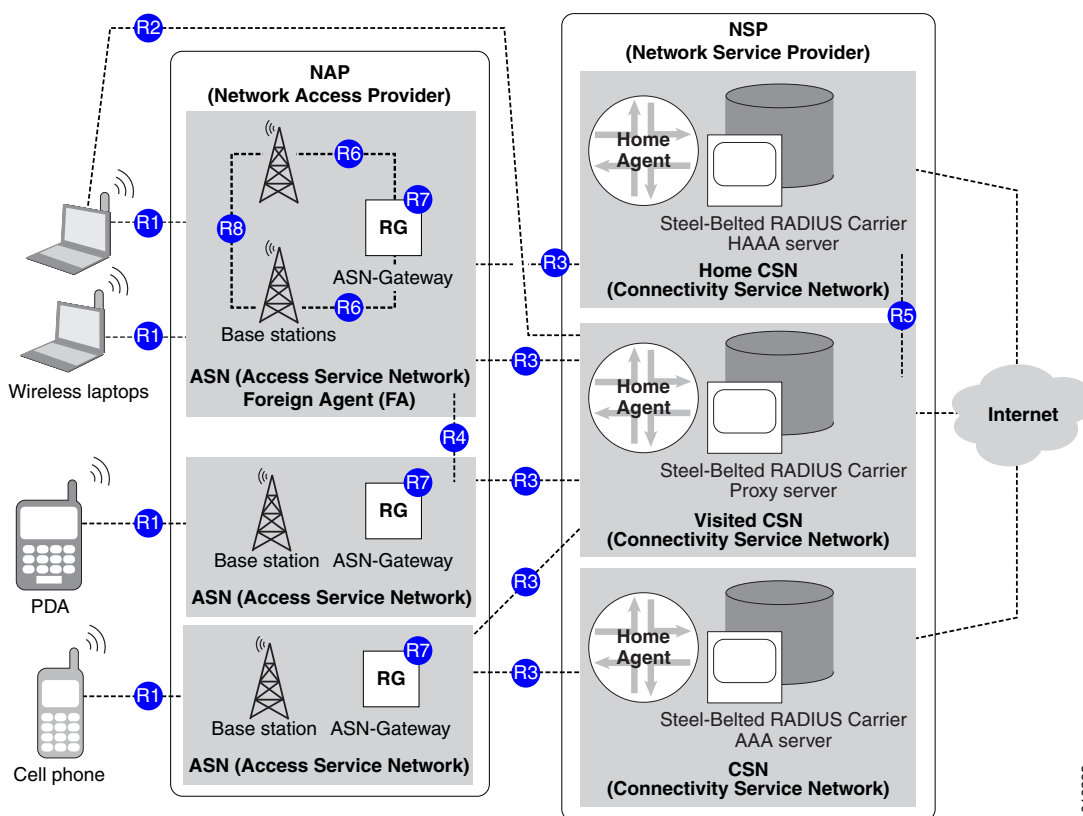
- Cryptographic key generation, management, and distribution
- EAP-TTLS, EAP-TLS, or EAP-AKA authentication methods
- WiMAX capabilities processing and negotiation
- Access-Request categorization by client type or realm or, both
- Home agent and DHCP server assignment, including HCSN and VCSN assignment

- Post-paid (offline) accounting, including per-flow accounting messages and continued sessions (Account-Stop/Account-Start pairs)
- Prepaid accounting solutions
- Reauthentication (authenticate-only service type) without handover
- Mobility and endeavor, including reauthentication and accounting with handover

WiMAX Network Reference Model

Figure 227 on page 555 shows the WiMAX network reference model containing links (interfaces or reference points) and functional entities.

Figure 227: WiMAX Network Reference Model



The WiMAX network reference model is composed of four logical parts:

- **Mobile Stations (MS)**—Comprises all user (subscriber) mobile devices, such as cell phones, PDAs, and wireless laptops, and software needed for communication with a wireless telephone network.
- **Network Access Provider (NAP)**—Provides radio access functionality. Contains the logical representation of the functions of a NAP. Some of the functions included in the NAP are: access service network (ASN), 802.16 interface with network entry and handover, ASN-GW (gateway), base stations (wireless towers), foreign agent (FA), QoS and policy enforcement, and forwarding to a selected CSN. A NAP may have contracts with multiple NSPs.
- **Network Service Provider (NSP)**—Provides IP connectivity services. Contains the logical representation of the functions of the NSP. Some of the functions included within the NSP are: connectivity service network (CSN), home agent, Visited and Home Authentication, Authorization, and Accounting servers (VAAA or HAAA), connectivity to the Internet, IP address management, authentication, authorization, and accounting, and mobility and roaming between ASNs. An NSP may have a contract with another NSP and may also have contracts between multiple NAPs.
- **Internet**—Provides Internet content to a user/subscriber and connectivity to a NSP.

Reference points (for example, R1 or R2) are conceptual links that connect two functional entities. Reference points represent a bundle of protocols between peer entities (similar to an IP network interface).

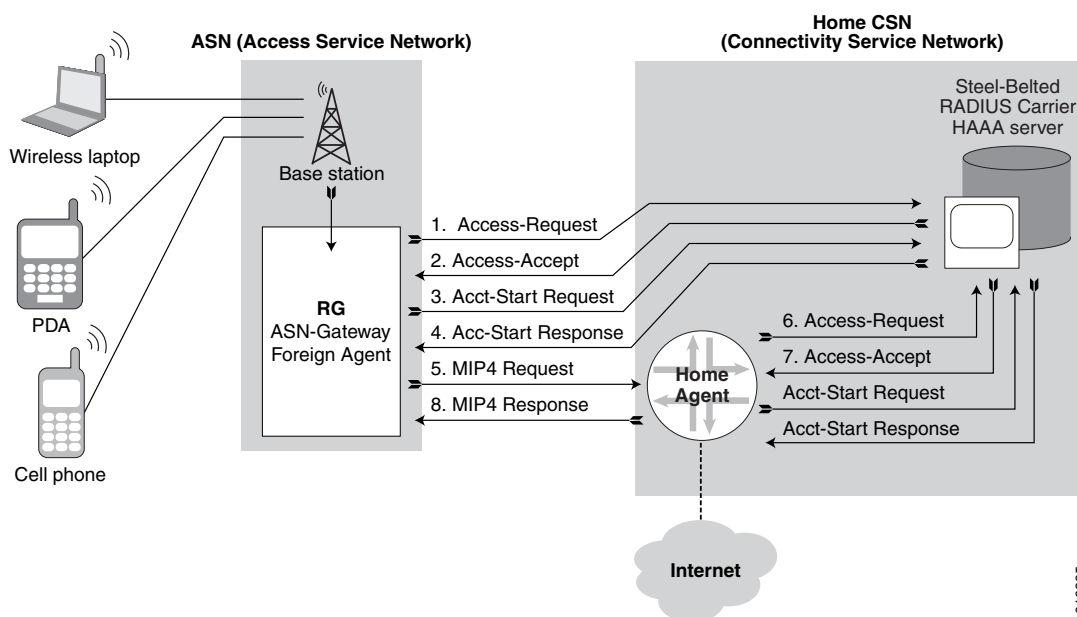
Interoperability is enforced through reference points without dictating how vendors implement the edges of those reference points.

- **R1**—Represents the interface between the wireless device and the base station.
- **R2**—Represents the link between the MS (mobile station) and the CSN (connectivity service network). EAP traffic from the mobile station to the AAA server traverses R2 and R3.
- **R3**—Represents the link between the ASN (access service network) and the CSN. RADIUS traffic between the ASN-GW and the AAA server traverses R3.
- **R4**—Represents the link between an ASN and another ASN.
- **R5**—Represents the link between a CSN and another CSN.
- **R6**—Located within an ASN and represents a link between the BS (base station) and the ASN-GW.
- **R7**—Located within the ASN-GW and represents internal communication within the gateway.
- **R8**—Located within an ASN and represents a link between two base stations.

Home Network Communication Flow Example

Figure 228 on page 557 shows a successful example of a WiMAX home network communication flow.

Figure 228: Home Network Communication Flow



For initial entry into the network, a mobile station uses a base station to attach itself to the network through the ASN-GW in the access service network (ASN). After the mobile station is attached to the network, the following communication flows occur:

1. The ASN-GW acts as the foreign agent and must authenticate the mobile device and its user by using an EAP-specific method (EAP-TTLS, EAP-TLS, or EAP-AKA). The ASN-GW also needs to obtain cryptographic keys. To retrieve the cryptographic keys, it sends a RADIUS Access-Request message to the SBR Carrier HAAA server in the home CSN (connectivity service network).
2. The SBR Carrier HAAA server receives the Access-Request message from the ASN-GW. Then, the HAAA server sends successful RADIUS Access-Accept and EAP-specific method messages back to the ASN-GW. Additionally, it sends the AAA-session-ID to use for the session (assigning the home agent) and the following cryptographic keys: MSK (master session key), MN-HA-MIP4-KEY, and MN-HA-MIP4-SPI. For more details about the EAP methods and cryptographic keys, see [“EAP Authentication Methods and EAP-Derived Cryptographic Keys” on page 560](#).
3. To start the accounting process for the session, the ASN-GW sends an Acct-Start Request message to the HAAA server. The accounting may be IP-session-based or flow-based. For more details about the supported WiMAX accounting methods, see [“WiMAX Post-Paid \(Offline\) Accounting” on page 568](#).

NOTE: Step 3 (Acct-Start Request) and Step 4 (Acct-Start Response) in [Figure 228 on page 557](#) are meant to show that accounting requests *may* occur at this stage. The precise ordering of the accounting messages is variable. Step 3 occurs sometime after Step 2, and Step 4 must occur sometime after Step 3. However, after Step 2, the ASN-GW can potentially send the Acct-Start Request (Step 3) or may send the MIP4 request (Step 5), or can even send both requests at the same time (Step 3 and Step 5).

4. To indicate the type of accounting it will use for the session, the HAAA server sends an Acct-Start Response message back to the ASN-GW to start the accounting session using the cryptographic keys and AAA-session-ID that were sent in the Access-Accept message.
5. After the ASN-GW receives the cryptographic keys from the HAAA server, it must send a separate Mobile IP registration (MIP4) request message to the home agent in the home CSN (connectivity service network).
6. The home agent performs an authentication check by sending the HAAA server an Access-Request message requesting its cryptographic keys for the Mobile IP session. The Access-Request message contains the home agent's cryptographic keys (MN-HA-MIP4-SPI and HA-RK-SPI).

NOTE: Optionally, an accounting session may also be started between the home agent and the HAAA server. The home agent sends an Acct-Start Request message to the HAAA server and assigns the Acct-session-ID. The HAAA server acknowledges the request and sends an Acct-Start Response message back to the home agent.

7. The HAAA server responds to the Access-Request message by sending the home agent an Access-Accept message containing its cryptographic keys: MN-HA-MIP4-KEY, MN-HA-MIP4-SPI, HA-RK-KEY, HA-RK-SPI, and HA-RK-Lifetime. For more details about these cryptographic keys, see [“EAP Authentication Methods and EAP-Derived Cryptographic Keys” on page 560](#).
8. After receiving the cryptographic keys from the HAAA server, the home agent sends the ASN-GW a MIP4 registration response message creating a Mobile IP binding used to set up a router path/connection to the Internet.

For this example, the router path/connection for the user/subscriber on a wireless laptop, cell phone, or PDA is now secure and complete. The connection follows this path: mobile station -> base station -> ASN-GW -> home agent -> Internet.

AAA-Generated Cryptographic Keys

Two cryptographic keys are generated by Steel-Belted Radius Carrier:

- Home agent root key (HA-RK)
- DHCP server root key (DHCP-RK)

The Steel-Belted Radius Carrier generates the keys and their associated security parameter indexes (SPI) and determines the lifetime of both keys.

Home Agent Root Key (HA-RK)

At least one HA-RK must be generated for each home agent. The HA-RK is a random number generated by the Steel-Belted Radius Carrier. The attribute used for the key is WiMAX-hHA-RK-KEY or WiMAX-vHA-RK-KEY.

When the Access-Accept message is sent to the access services network gateway (ASN-GW) the Steel-Belted Radius Carrier queries its internal HA-RK list for the home agent and selects the HA-RK with the longest lifetime. If the remaining lifetime of all HA-RKs is less than the master session key (MSK) lifetime (value of the Session-Timeout attribute), then the Steel-Belted Radius Carrier server generates a new HA-RK with a longer lifetime than the MSK lifetime.

The Steel-Belted Radius Carrier generates a unique SPI for each home agent (HA-RK-SPI attribute). The SPI is a 32-bit integer that is randomly generated. The home agent is identified by a combination of its IPv4 address and the HA-RK-SPI attribute.

The HA-RK-Lifetime attribute represents the lifetime of HA-RK-KEY. The Session-Timeout attribute is sent by the Steel-Belted Radius Carrier to the ASN-GW in the Access-Accept message. It specifies the lifetime of the MSK and all extended master session key (EMSK) derived keys, and indicates the maximum number of seconds of service to be provided to the user before termination of the session. The Session-Timeout attribute is configurable using Web GUI or through an external SQL or LDAP database.

Make sure to configure the HA-RK-Lifetime attribute significantly larger than the Session-Timeout attribute.

The HA-RK key is destroyed *only* when its lifetime has expired.

After the HA-RK key has been generated, it is stored for possible future retrieval. One or more keys are stored for each home agent.

Allowing the VAAA to Assign the HA-RK

If the hHA-IP-MIP4 attribute is received from the ASN-GW in an Access-Request, it indicates that the VCSN proxy AAA server (VAAA) is capable of assigning the home agent IP address, vHA-RK-Key, vHA-RK-SPI, and vHA-RK-Lifetime values. If the **Allow-VAAA-To-Assign-Home-Agent-And-DHCP-Server** parameter is set to 1 in the **wimax.ini** file, Steel-Belted Radius Carrier allows the VAAA to set these values, echoes the received hHA-IP-MIP4 as vHA-IP-MIP4 and attaches the vHA-IP-MIP4, MN-vHA-MIP4-KEY,

and MN-vHA-MIP4-SPI attributes to the Access-Accept message. For more information see [“Home Agent and DHCP Server Assignment” on page 567](#).

DHCP Server Root Key (DHCP-RK)

The DHCP-RK is very similar to the HA-RK. However, instead of an SPI, the DHCP-RK is identified by the DHCP-RK-Key-ID attribute.

If the DHCP-RK-Key-ID attribute is received in the Access-Request from the DHCP server, it contains the identifier of one of the DHCP-RK keys for the DHCP server.

The DHCP-RK is a random number generated by the Steel-Belted Radius Carrier server. At least one DHCP-RK must be generated for each DHCP server. The DHCP server is identified by the DHCPv4-Server attribute. The [DHCP-Servers] section of **wimax.ini** lists the IPv4 addresses of all allowed DHCP servers.

When Steel-Belted Radius Carrier sends the Access-Accept message to the ASN-GW, the server queries the DHCP-RK list for that DHCP server and selects the DHCP-RK with the longest lifetime. If the remaining lifetime of all DHCP-RKs is less than the MSK lifetime (value of the Session-Timeout attribute), then Steel-Belted Radius Carrier generates a new DHCP-RK with a longer lifetime than the MSK lifetime.

The DHCP-RK-Lifetime attribute represents the lifetime of DHCP-RK. This attribute is attached to the Access-Accept by SBR Carrier, and sent to the ASN-GW. Make sure to configure the DHCP-RK-Lifetime attribute with a value significantly larger than the Session-Timeout attribute. The DHCP-RK lifetime configuration is added to **wimax.ini**. The configuration is global. It applies to all DHCP-RKs.

The Session-Timeout attribute specifies the lifetime of MSK and all EMSK derived keys, and is the maximum number of seconds of service to be provided to the user before termination of the session. The Session-Timeout attribute is configurable using the Web GUI or through an external SQL or LDAP database. Make sure to configure the Session-Timeout attribute value significantly smaller than the DHCP-RK-Lifetime attribute.

The DHCP-RK key is destroyed *only* when its lifetime has expired.

After the DHCP-RK key has been generated at the time of ASN-GW authentication, the key is stored for possible future retrieval. One or more keys are stored for each DHCP server, where the DHCP server is identified by its IPV4 address.

EAP Authentication Methods and EAP-Derived Cryptographic Keys

Both the MSK and EMSK are generated as by-products of successful EAP authentication. Various WiMAX cryptographic keys are derived from the MSK and EMSK. The following EAP authentication protocols can be used with the WiMAX mobility module:

- EAP-TLS (Transport Layer Security) protocol

- EAP-TTLS (Tunneled Transport Layer Security) protocol
- EAP-AKA (Authentication and Key Agreement) protocol.

NOTE: The EAP-AKA protocol requires a license for the optional SIM authentication module.

Master Session Key (MSK)

The MSK attribute is derived as the result of successful EAP authentication. It may be sent to the ASN-GW by Steel-Belted Radius Carrier in the Access-Accept message. It is not sent to the home agent. No SPI is associated with the MSK.

Steel-Belted Radius Carrier uses the first 64 bits of keying material as the MSK.

The Session-Timeout attribute specifies the MSK lifetime. It is sent to the ASN-GW by Steel-Belted Radius Carrier in the Access-Accept message, and it indicates the maximum number of seconds of service to be provided to the user before termination of the session. You can configure the Session-Timeout attribute using Web GUI or through an external SQL or LDAP database.

It is the responsibility of the ASN-GW to reauthenticate before the MSK expires.

Extended Master Session Key (EMSK)

Steel-Belted Radius Carrier generates the Mobile IP-root key (MIP-RK) from the EMSK and then uses the MIP-RK to generate the following additional keys:

- MN-HA CMIP4—Mobile node-home agent client Mobile IP for IPv4 key.
- MN-HA PMIP4—Mobile node-home agent proxy Mobile IP for IPv4 key.
- FA-RK—Foreign agent-root key.
- RRQ-MN-HA—Registration request (part of the MIP protocol) mobile node home agent key.

Steel-Belted Radius Carrier sends these EMSK-derived keys to both the ASN-GW and the home agent.

EMSK-Derived Key Generation and Identification

The MN-HA CMIP4, MN-HA PMIP4, and FA-RK keys are each identified by a unique SPI value. SPI values relating to the current Mobile IP (MIP) session are unique, where the current MIP session is identified by the pseudo-identifier (outer username) associated with the session.

NOTE: The MIP session is not the same as the RADIUS session. The RADIUS session can be identified by its associated AAA-session-ID value.

Therefore, each MN-HA CMIP4, MN-HA PMIP4, and FA-RK key is identified by a combination of the pseudo-identifier and SPI.

MSK and EMSK-Derived Key Lifetime and Deprecation

When a final Account-Stop is received for the pseudo-identity from the ASN-GW, the MSK and EMSK-derived key set is destroyed. The lifetime of the keys is specified by the Session-Timeout attribute sent to the ASN-GW. It is the responsibility of the ASN-GW to reauthenticate with Steel-Belted Radius Carrier before the MSK and EMSK-derived key set expires. After the ASN-GW obtains the new keys, it communicates with the home agent using the MIP protocol enabling the home agent to obtain the new keys from Steel-Belted Radius Carrier.

After the home agent has requested the new key set (by supplying the new MN-HA-SPI value), the old key set is destroyed.

NOTE: For a short period of time, two key sets are active. During this time, when a new key set is used, all other older key sets are destroyed.

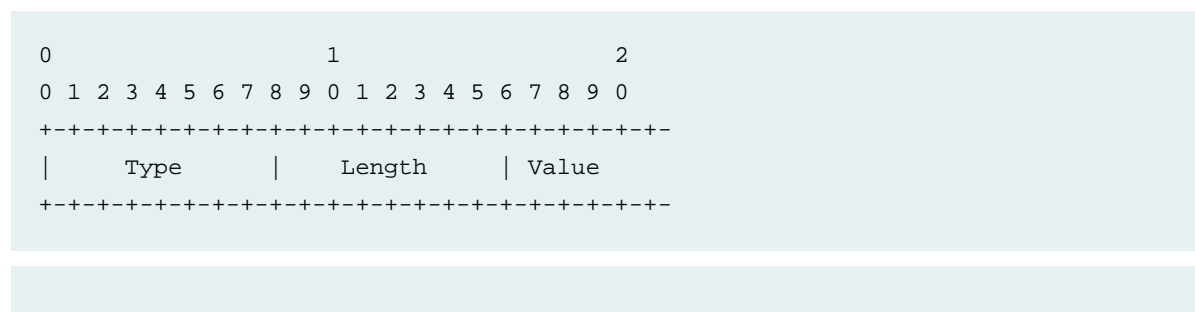
EMSK-Derived Key Storage and Retrieval

After the key is generated, it is stored for possible later retrieval. One or more key sets are stored for each MIP session, where a MIP session is identified by the pseudo-identifier.

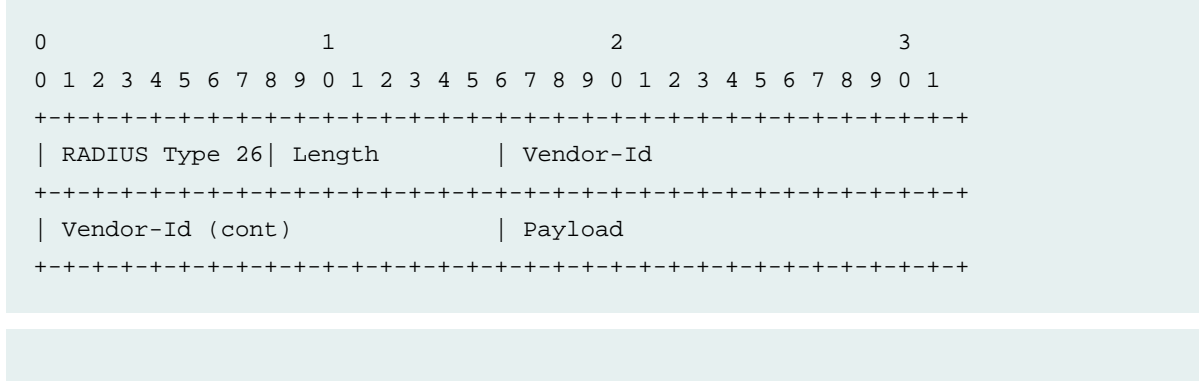
WiMAX Vendor Specific Attribute (VSA) Format

To support the length of WiMAX attributes, Steel-Belted Radius Carrier natively supports structured attributes. These WiMAX RADIUS VSAs are transported in a RADIUS VSA. Each attribute format contains the following bit definitions:

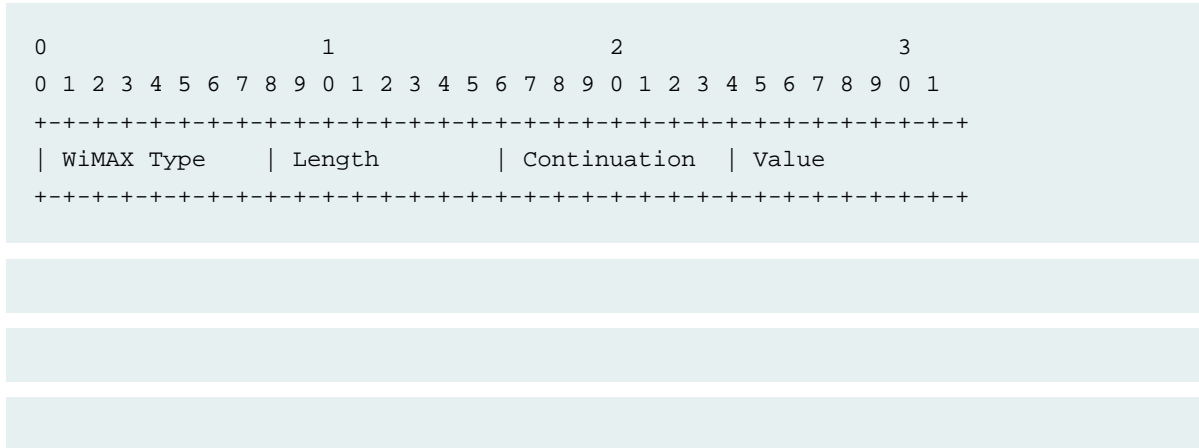
- The RADIUS attribute format is:



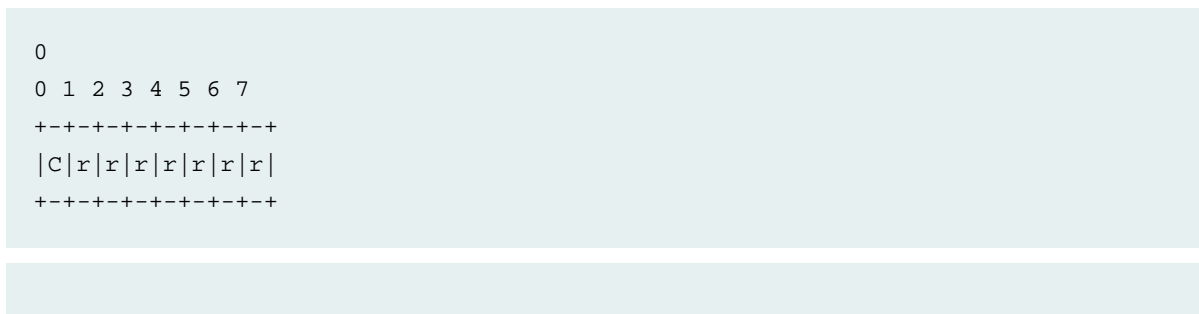
- The RADIUS VSA format is:



- The WiMAX VSA Payload format is:



The **Continuation** field of the WiMAX VSA format is:



If the WiMAX VSA value is larger than 246 bytes, then the value must be fragmented into a series of WiMAX VSAs. The C-bit of the **Continuation** field indicates whether a WiMAX attribute is fragmented.

- When the C-bit=0, the WiMAX attribute is not fragmented.
- When the C-bit=1, the WiMAX attribute is fragmented. The next WiMAX VSA of the same WiMAX type is appended to this attribute.

A fragmented WiMAX attribute has its C-bit set to 1 for all of its fragments except for its last fragment. To indicate that the fragment is the last fragment of the attribute, its C-bit is set to 0.

When using XML to specify the structured attribute format, the `hasContinuationFlag=true` attribute represents the C-bit. This element indicates that the attribute starts with a continuation octet, where the payload of the attribute may be split over multiple VSAs.

The r-bits are reserved for future use. They are set to 0 by the sender and are ignored by the receiver.

Structured Attributes

SBR Carrier natively supports structured attributes that contain subattributes. Subattributes like normal RADIUS attribute-value pair (AVPs), consist of the raw encoding of a type field (such as 1 for WiMAX-Release, within the WiMAX-Capability VSA) followed by a length value (such as 5) followed by the value of the attribute (such as 1.2).

Subattributes are values in a RADIUS packet that are not stored as a RADIUS AVP, or vendor-specific-attribute (VSA), but rather are packed with other subattributes into a RADIUS VSA. In a RADIUS packet, multiple RADIUS VSAs might contain subattributes. The RADIUS VSA, which consists of multiple subattributes, is sometimes referred to as a *structured attribute* because it contains structured data.

NOTE: You specify structured attributes using the “.” notation. For example:

“A.b.c”

Where attribute “A” is a group attribute containing a sequence subattribute “b”, which contains a simple attribute “c”.

Structured attributes are addressable only by their full *pathname*, which must include all interim group or sequence attributes.

You do not need to add every subattribute in a tree. Steel-Belted Radius Carrier handles partially defined trees by automatically supplying subattributes that have a default value specified in the **.jdict** definition files. Additionally, if a subattribute is added that does not have a suitable parent or group defined, Steel-Belted Radius Carrier automatically creates one.

For information about specifying structured attributes, see the section on *Attribute Processing Files* of the *SBR Carrier Reference Guide*.

WiMAX Capabilities Negotiation

The ASN-GW and home agent each send a WiMAX-Capability attribute listing the capabilities they want to use in the session. The HAAA server selects from those received capabilities and responds with a WiMAX-Capability attribute listing the capabilities it can support for the session. The HAAA capabilities configuration (policy) may differ for the ASN-GW and home agent.

WiMAX-Capability Attribute

The WiMAX-Capability attribute is a structured VSA that contains subattributes that specify different capabilities that can be used in the session. If a subattribute (capability) was sent in the Access-Request, and the HAAA (Steel-Belted Radius Carrier) can support the capability, then the subattribute is returned in the Access-Accept message. If a subattribute (capability) is not sent in the Access-Request, it is not returned in the Access-Accept. Absence of a subattribute in the Access-Request, indicates the device (ASN-GW or home agent) does not support the capability.

WiMAX-Capability Structured Attribute

The WiMAX-Capability structured attribute may contain the following subattributes, each representing a specific WiMAX capability:

- WiMAX-Release attribute
- Accounting-Capabilities attribute
- Hotlining-Capabilities attribute
- Idle-Mode-Notification-Capabilities attribute

WiMAX-Release Attribute

The WiMAX-Release subattribute indicates the release of the WiMAX Forum Networking Group (NWG) specification supported by the device. Steel-Belted Radius Carrier supports releases 1.0, 1.1, 1.1.1, 1.1.2, and 1.2. The WiMAX-Release subattribute is sent in the Access-Request by both the ASN-GW and home agent. If Steel-Belted Radius Carrier supports the requested release number, it returns that same release number with the Access-Accept. If Steel-Belted Radius Carrier does not support the requested release number it rejects the request.

WiMAX-Accounting-Capabilities Attribute

The WiMAX-Accounting-Capabilities subattribute specifies the type of accounting to be used for the session. It must be sent in the Access-Request by both ASN-GW and home agent, and it must be returned to both the ASN-GW and home agent by the HAAA server in the Access-Accept.

Steel-Belted Radius Carrier supports flow-based and IP-session-based accounting. A value of *None* is allowed only at the home agent. If the ASN-GW indicates a value of *None*, the request is rejected.

For more details about accounting, see [“WiMAX Post-Paid \(Offline\) Accounting” on page 568](#).

Hotlining-Capabilities Attribute

Hotlining is the practice of diverting a subscriber from their desired destination to a destination controlled by the service provider, in order to rectify specific circumstances related to the subscriber account. Hotlining is typically used to reconcile issues such as prepayment of services, delinquent account issues, account updates, and administrative functions (for example IP address renewal). The subscriber is typically redirected to a webpage to reconcile the issue. After the issue is resolved, they are redirected to their desired destination.

There are two types of hotlining:

- New session hotlining
- Active session hotlining

New session hotlining is the process of returning hotlining parameters attached to the Access-Accept. The hotlining parameters are returned in the Access-Accept message based on the value of the Hotlining-Capabilities subattribute in the WiMAX-Capability attribute. The hotlining parameters are returned as WiMAX hotlining attributes. The values of these attributes can be stored in either an SQL or LDAP database, and are retrieved using the SQL or LDAP authentication files (`sqlauth.aut` or `ldapauth.aut`).

Active session hotlining is the process of returning hotlining parameters through Change of Authorization (CoA) messages. For more information about SBR Carrier's support for CoA messages, see [“Managing and Controlling Sessions” on page 694](#).

For more information about configuring new session hotlining, see [“Example Configuration for New Session Hotlining” on page 595](#).

Idle-Mode-Notification-Capabilities Attribute

Idle mode notification can be negotiated at the time of network access. During network access, if the ASN-GW sends the WiMAX-Capabilities attribute using the Idle-Mode-Notification-Capability sub-attribute, it indicates the ASN-GW supports idle mode notification. Absence in the Access-Request means that idle mode notification is not supported by the ASN-GW.

The HAAA server indicates it requires idle mode notification by returning this subattribute to the ASN-GW in the Access-Accept. Absence of this subattribute in the WiMAX-Capabilities attribute in the Access-Accept indicates the HAAA server does not require idle mode notification.

The Idle-Mode-Notification-Capabilities subattribute is never sent to the home agent.

Idle mode notification is negotiated at network access and occurs when the mobile station (for example, cell phone) enters or exits the idle mode. The accounting server at the CSN (connectivity service network) is notified about the idle mode transition through accounting messages.

Enabling WiMAX Capabilities Negotiation

WiMAX capabilities negotiation may be performed by adding capabilities subattributes to the user or profile return list and enabling the Echo option for each capability you want Steel-Belted Radius Carrier

to support. When using the SQL or LDAP authentication plug-ins (for EAP-AKA), this is done by defining the attributes in the return list of a profile.

If you want Steel-Belted Radius Carrier to support all requested WiMAX capabilities defined in an Access-Request message, enable the Echo option on the parent WiMAX-Capability attribute. If you want to selectively choose what capabilities Steel-Belted Radius Carrier supports, enable the Echo at the subattribute level only.

For example, if you do not want to support idle mode notification, do not enable Echo for the Idle-Mode-Notification-Capability subattribute.

Home Agent and DHCP Server Assignment

The home agent IP address is assigned with the WiMAX-HA-IP-MIP4 attribute. The DHCP server IP address is assigned with the WiMAX-DHCPv4-Server attribute. This section provides a brief description on how Steel-Belted Radius Carrier can assign these addresses.

Assignment Using Return List Attributes

SBR Carrier can assign the home agent and DHCP server addresses by returning the WiMAX-HA-IP-MIP4 and WiMAX-DHCPv4-Server attributes in the Access-Accept message. This is configured by defining these attributes in the return list of a user or profile entry.

The address assignment process and configuration differs depending on whether the server is acting as the HAAA or VAAA. Steel-Belted Radius Carrier can be configured to perform either role. For details on configuring this method, see [“Configuring Return List Attributes to Assign the Home Agent and DHCP Server” on page 585](#).

Assignment Using Statically Weighted Round-Robin Groups

By using statically weighted round-robin groups, SBR Carrier can load balance the IP address assignment of multiple home agents and DHCP servers in the network. The round-robin feature enables you to configure a home agent round-robin group and assign fixed weights to each home agent in the group. SBR Carrier then assigns the home agent to a session based on a weighted round-robin method. For details on configuring this method, see [“Configuring Statically Weighted Round-Robin Groups to Assign the Home Agent and DHCP Server” on page 586](#).

Assignment Using the Smart Dynamic Home Agent Assignment Feature

The smart dynamic home agent assignment feature uses several configuration files that define both the round-robin groups, and their associated weights. The weight file is dynamically updated. When SBR Carrier receives a SIGHUP (1) signal, it reads these configuration files and updates the round-robin groups. You

can assess the load status of the home agents in your network and populate the associated configuration files in a way that balances the load across home agents. For details on configuring this method, see [“Configuring the Smart Dynamic Home Agent Assignment Feature” on page 588](#).

WiMAX Post-Paid (Offline) Accounting

Accounting in WiMAX is based on a subscription that is identified through the subscription’s NAI. A single subscriber can have multiple subscriptions. The WiMAX mobility module supports the following post-paid (offline) accounting features:

- Flow-based accounting
- IP-session-based accounting

For a diagram depicting accounting events within a home network, see [Figure 228 on page 557](#).

Accounting events for the ASN-GW, home agent, and DHCP server in either the home network or visited network are recorded in accounting log files. Accounting log files are named **yyyymmdd.act**, where **yyyy** is the 4-digit year, **mm** is the month, and **dd** is the day on which the log file was created. Accounting events include START messages, which indicate the beginning of a connection; STOP messages, which indicate the termination of a connection; and INTERIM messages, which indicate a connection is ongoing.

For more details about logging and reconciling accounting records, see [“Using the Accounting Log File” on page 854](#).

Flow-Based Accounting

Flow-based accounting sessions are defined by starting and stopping flows. An Account-Start (defined with its proper attributes) marks the start of a flow and an Account-Stop (defined with its proper attributes) marks the end of a flow.

An Account-Start is received for each WiMAX packet flow. Flows are identified at the PDSN (Packet Data Serving Node) using packet filters. Packet filters are used to map forward traffic and for per-flow accounting in the forward and reverse direction.

Flow-based accounting messages do not create or destroy sessions, but will create entries in the accounting log file. These messages are received by the SQL accounting plug-in. For more details about using the SQL accounting plug-in, see [“Configuring SQL Accounting” on page 458](#).

The attributes in [Table 64 on page 569](#) are used to determine the state of the flow:

Table 64: WiMAX Packet Flow Attributes

Attribute	Description
WiMAX-Session-Continue	<p>Specifies whether the WiMAX packet flow continues or reaches the final stop of an accounting session.</p> <ul style="list-style-type: none"> • If set to 0 (False) or if the attribute is missing, indicates that the WiMAX packet flow has reached the final stop of the accounting session. • If set to 1 (True), indicates that the WiMAX packet flow is not at the end of an accounting session (an Account-Stop is immediately followed by another Account-Start).
WiMAX-Beginning-Of-Session	<p>Specifies whether the WiMAX packet flow is starting a new accounting flow or continues with a previous accounting flow.</p> <ul style="list-style-type: none"> • If set to 0 (False) or if the attribute is missing, indicates that the WiMAX packet flow is a continuation of a previous accounting session. • If set to 1 (True), indicates that the WiMAX packet flow is starting a new flow of an accounting session.

Flows are preprovisioned and are described in the Packet-Flow-Descriptor attribute. A packet flow may describe a unidirectional flow and bidirectional flow. The Packet-Flow-Descriptor attribute is attached to the initial Access-Accept. Each (potential) flow is identified by a package data flow ID (PDFID) and announces when a flow has started. Its value matches all records from the same packet data flow. The PDFID correlates all accounting records, and is assigned by the connectivity services network (CSN) and remains constant through all handover scenarios.

For each flow that has started within an accounting session, before the accounting session can be closed (stopped), each flow within that accounting session must be stopped. A CSN can restart a flow by sending an existing PDFID in a new Account-Start.

IP-Session-Based Accounting

The WiMAX-Session-Continue attribute (as described in [Table 64 on page 569](#)) defines when an accounting session actually ends. Each Accounting Start/Stop packet delineates the following:

- A complete session or flow
- A segment of a session or flow

A session or flow may consist of several accounting segments. Accounting segmentation occurs due to:

- Handover with accounting client relocation

- Change in status such as hot-line state

The accounting attributes WiMAX-Beginning-Of-Session and WiMAX-Session-Continue, determine the interpretation of the Accounting-Request packets as shown in [Table 65 on page 570](#):

Table 65: Interpretation of Accounting-Request Packets

Acct-Status-Type	WiMAX-Beginning-Of-Session	WiMAX-Session-Continue	Description
Start	True	N/A	Beginning of the first accounting segment for a session or flow.
Start	False (or missing)	N/A	Beginning of a subsequent accounting segment of a session or flow.
Start	N/A	True	The end of the accounting segment. Another accounting segment is starting; expect an Accounting-Request (Start).
Stop	N/A	False	This is the end of the session or flow.

Each accounting session is identified by an Acct-Multi-Session-Id attribute. Flows within an accounting session are identified by the Acct-Multi-Session-Id. The Acct-Multi-Session-Id attribute contains the same value as the AAA-Session-Id attribute. The identifier is set to the value of the AAA-Session-Id which is generated by the AAA server after successful authentication and delivered to the ASN-GW in an Access-Accept message. The AAA-Session-Id is unique per AAA server, but is used for all flows.

The Acct-Multi-Session-Id correlates accounting records for a device session on a particular device for a given subscription.

WiMAX Prepaid Accounting

Prepaid solutions are generally either single-service or multi-service solutions. This section provides a description of these solutions and describes how to configure SBR Carrier to support them.

Prepaid Scenarios

Single-Service Prepaid Solution

This is a prepaid solution that manages a single service, hence the quota in this solution is only consumed by a single service. Single-service prepaid services range from true prepaid accounts to one-time models like scratch-card and credit card one-time services. This type of solution typically does not require a full-feature prepaid server (PPS), thus greatly simplifying deployment of the solution and reducing your cost. Since only a single service consumes the prepaid quota, in-session quota management is less complex than a multi-service solution. In this case SBR Carrier retrieves the complete quota from the subscriber database on session start, and returns that volume or time based quota to the NAS on authentication. The NAS reports the quota usage through interim accounting and SBR Carrier updates the quota in the subscriber database accordingly. When the session ends, SBR Carrier performs the final update, and any remaining quota is available for the next session.

Multi-Service Prepaid Solution

In true prepaid multi-service scenarios, the subscriber's quota is consumed by multiple services which requires more sophisticated in-session quota management, as well as a prepaid server to manage that quota. In this scenario, the PPS assigns small portions of the quota when a service is initiated, and the service enforcement point then requests additional quota during the lifetime of that service. There are basically three multi-service prepaid models:

PPS Controlled Prepaid Model

In this model the PPS manages the subscriber's quota. The NAS has no quota management capabilities at all. The NAS authenticates the subscriber with SBR Carrier, which then triggers the PPS server to start quota management for the session. When the subscriber's quota on the PPS is depleted, the PPS can trigger SBR Carrier to either:

- Issue a RADIUS Change of Authorization (CoA) request that redirects the subscriber to a prepaid web portal where they can purchase more time or data for their account.
- Issue a RADIUS Disconnect Message to disconnect the subscriber from the network.

AAA Proxy Model

In this model the NAS initiates RADIUS Authorize-Only requests or standard Authentication-Requests to obtain portions of quota during the session lifetime. SBR Carrier forwards these requests to the PPS, either by RADIUS proxy or by using one of SBR Carrier's authentication methods. In this use model the NAS manages the quota which is assigned by the PPS.

Direct PPS Connection

In this model the NAS directly manages the quota assigned by the PPS; the AAA server is not in the quota request path. In this scenario, SBR Carrier identifies the subscriber as being prepaid in their subscriber profile, and assigns a PPS address (or addresses) that the NAS uses to perform quota management for the specific subscriber.

Data Flow for Prepaid Accounting in SBR Carrier

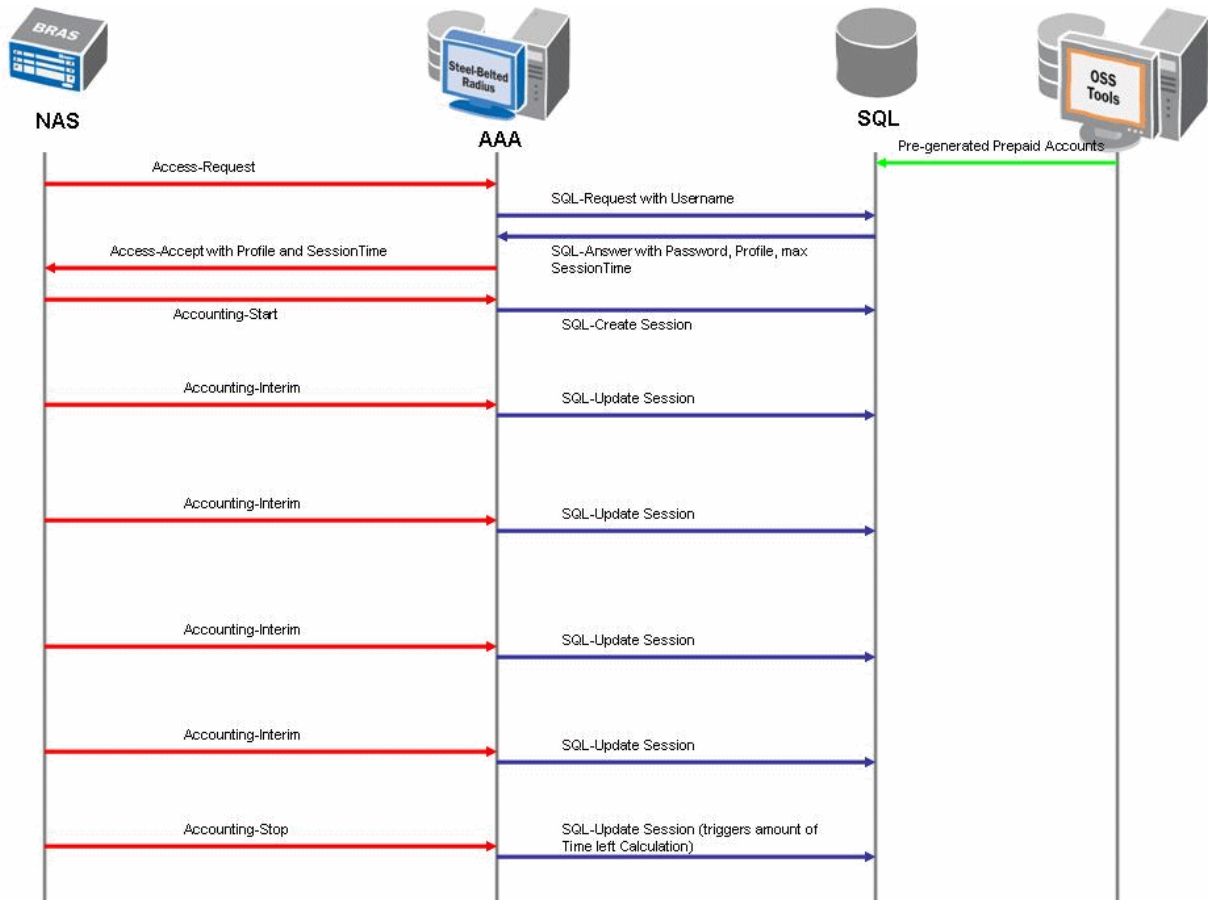
This section provides an overview of the data flow for the prepaid accounting solutions.

Data Flow for Single-Service Prepaid Accounting Model

This option is a simple and cost effective way of implementing a single prepaid service, since it does not require quota management capabilities on the NAS or a PPS system. The NAS receives the entire quota for the subscriber account and can completely consume it without requesting additional quota.

Figure 229 on page 572 shows an example data flow for the WiMAX single-service prepaid accounting solution using SQL authentication and accounting.

Figure 229: Example Data Flow for WiMAX Single-Service Prepaid Accounting Model



The following section provides a more detailed discussion of the data flow for the WiMAX single-service prepaid accounting shown in Figure 229 on page 572.

1. An operations support systems (OSS) generates subscriber accounts into the subscriber and billing database. These accounts have a predefined quota. There may also be a self-service application in which subscribers are able to increase their quota.
2. The NAS sends an Access-Request containing the Username the subscriber entered (and the password depending on the authentication method).
3. SBR Carrier sends an SQL-Request to the subscriber and billing database requesting the subscriber credentials, the subscriber profile (QoS, bandwidth, and so forth) and the available quota. Refer to [“Configuring SQL Authentication” on page 442](#) for information about how to configure SQL authentication in SBR Carrier.
4. The subscriber and billing database sends back the credentials, profile information and quota.
5. SBR Carrier compares the credentials and if they are acceptable, a final Access-Accept message is sent to the NAS. The Access-Accept message contains the subscriber profile information that is enforced for the lifetime of the subscriber session. SBR Carrier also sends the time or volume based quota to the NAS.
6. The NAS starts a session for the subscriber and sends an initial Accounting-Start message to SBR Carrier.
7. SBR Carrier conveys the Accounting-Start information to the subscriber and billing database, updating the session in the database through SQL accounting. Refer to [“Configuring SQL Accounting” on page 458](#) for information about how to configure SQL accounting in SBR Carrier.
8. The NAS sends an Accounting-Interim message to SBR Carrier, containing the time and volume the subscriber has used since starting the session.
9. SBR Carrier updates the relevant information in the subscriber and billing database.
10. Steps 8 and 9 are repeated until the NAS detects that either the quota limit is reached for the session, or the subscriber has disconnected. If the subscriber does not disconnect, the NAS disconnects or redirects the subscriber and sends a final Accounting-Stop to SBR Carrier.
11. SBR Carrier updates the subscriber and billing database using SQL accounting, including the subscriber's remaining quota.

Data Flow for Multi-Service Prepaid Accounting Models

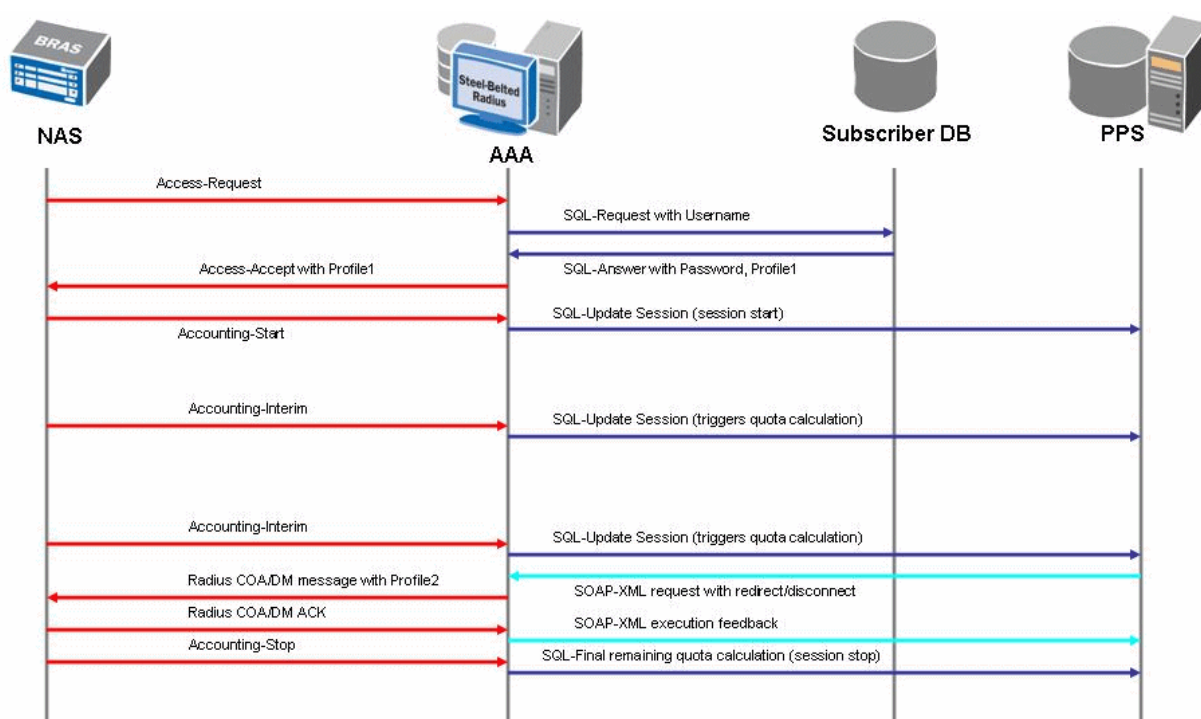
This section provides an overview of the data flow and configuration for the PPS controlled and AAA proxy prepaid accounting models.

PPS Controlled Prepaid Model

The PPS controlled prepaid accounting model uses a RADIUS extension for dynamic changes during the session called Change of Authorization (CoA) according to RFC 3576. To support this model, the NAS must support the RFC 3576 RADIUS extensions. To enable the RADIUS CoA/DM functionality in SBR Carrier, the optional Session Control module is required on each SBR Carrier instance in a clustered installation requiring the SSR component. For more information about the optional Session Control module, see [“Managing and Controlling Sessions” on page 694](#).

[Figure 230 on page 574](#) shows an example data flow for the WiMAX PPS controlled prepaid model using SQL authentication and accounting. This example uses SQL to convey session update information to the PPS. Proxy RADIUS can also be used to convey this information.

Figure 230: Example Data Flow for WiMAX PPS Controlled Prepaid Accounting Model



The following section provides a more detailed discussion of the data flow for the WiMAX PPS controlled prepaid accounting shown in [Figure 230 on page 574](#).

1. The NAS sends an Access-Request containing the Username the subscriber entered (and the password depending on the authentication method).
2. SBR Carrier sends an SQL-Request to the subscriber database requesting the subscriber credentials and profile (QoS, bandwidth, and so forth). Refer to [“Configuring SQL Authentication” on page 442](#) for information about how to configure SQL authentication in SBR Carrier.

3. The subscriber database sends back the credentials and profile information.
4. SBR Carrier compares the credentials and if they are acceptable, a final Access-Accept message is sent to the NAS. The Access-Accept message contains the subscriber profile information that is enforced for the lifetime of the subscriber session.
5. The NAS starts a session for the subscriber and sends an initial Accounting-Start to SBR Carrier.
6. SBR Carrier conveys the Accounting-Start information to the PPS, updating the session in the database through SQL accounting. The PPS starts counting down the quota. Refer to [“Configuring SQL Accounting” on page 458](#) for information about how to configure SQL accounting in SBR Carrier.
7. The NAS sends an Accounting-Interim message to SBR Carrier, containing the time and volume the subscriber has used since starting the session.
8. SBR Carrier updates the relevant information in the PPS through SQL accounting.
9. Steps 7 and 8 are repeated until the subscriber runs out of quota in the PPS.
10. When the quota is exhausted, the PPS send's an HTTPS-XML message to SBR Carrier indicating what to do next. For example, the PPS may send a redirect message to SBR Carrier. The redirect message triggers the server to send a CoA message to the NAS which includes new profile information for the subscriber session. The new profile information may reduce in the subscriber's bandwidth, or may redirect the subscriber to a web portal where they can renew their quota. Alternatively, when the quota is exhausted, the PPS can send an HTTPS-XML message to SBR Carrier requesting the subscriber be disconnected (by means of a Disconnect-Message (DM)). For more information about how to trigger CoA/DM requests through the HTTPS-XML interface of the Session Control module, see [“Managing and Controlling Sessions” on page 694](#).
11. SBR Carrier sends the change request from the PPS as either a RADIUS CoA-Message or Disconnect-Message to the NAS with the new profile. To learn how to configure the Session Control module to emit RADIUS CoA/DM requests, see [“Managing and Controlling Sessions” on page 694](#).
12. The NAS changes the profile for the subscriber, and on success, sends an ACK-Message back to SBR Carrier.
13. SBR Carrier sends a result code back to the PPS over the HTTPS-XML interface indicating whether the request was successful or not.

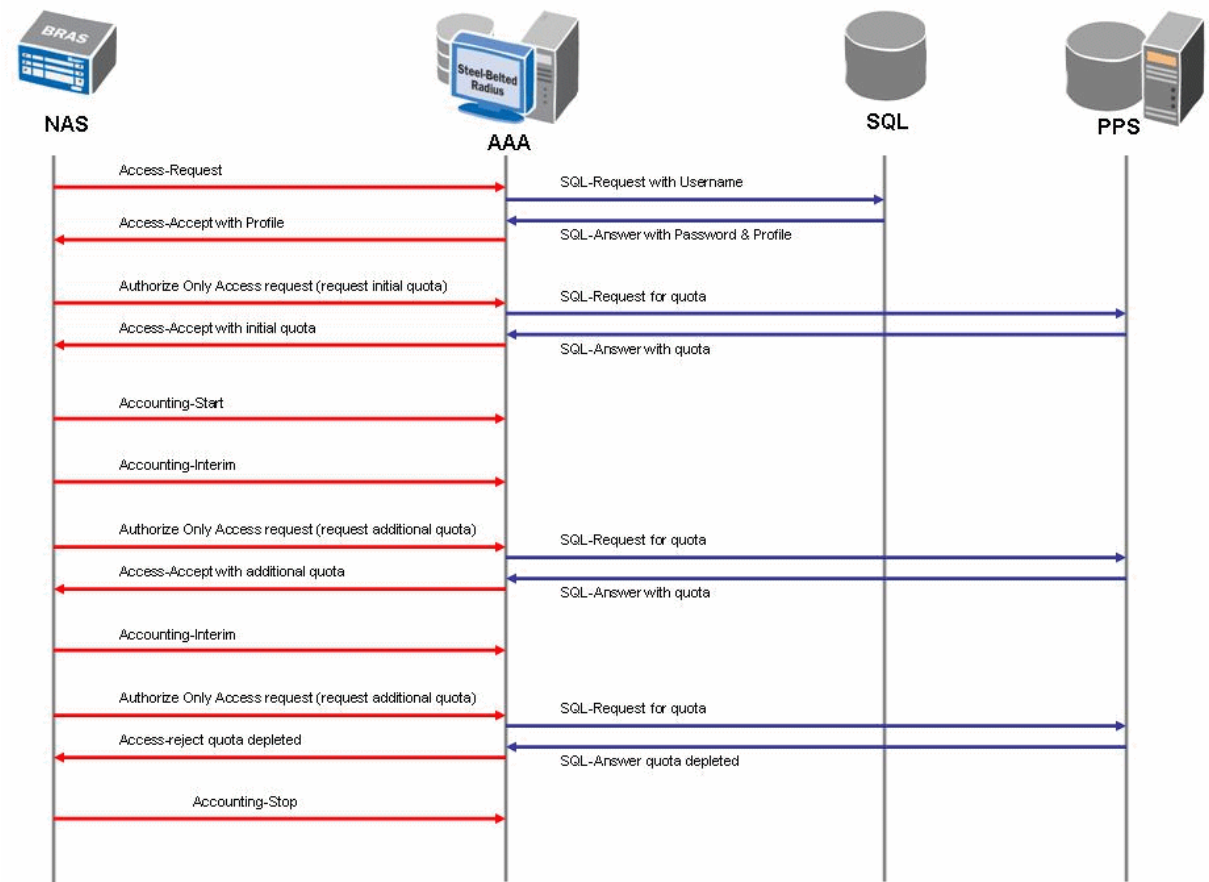
14. The NAS sends an Accounting-Stop message to SBR Carrier to terminate the session.
15. SBR Carrier sends the Accounting-Stop message as SQL-Update to the PPS so it can calculate the final amount of quota left for the old session. For more information about SQL accounting, see [“Configuring SQL Accounting” on page 458](#).

AAA Proxy Model

In the AAA proxy prepaid accounting model, the NAS manages time or volume based quota which is updated by periodic reauthorization. The reauthorization is handled through RADIUS authentication or RADIUS Authorize-Only messages to the AAA server, which in turn, retrieves portions of the quota from the PPS and encapsulates them in a RADIUS attribute in the authentication/authorization reply.

[Figure 231 on page 576](#) shows an example data flow for the WiMAX AAA proxy prepaid accounting model. This example uses SQL as a method for communication with a prepaid system, however proxy RADIUS could also be used.

Figure 231: Example Data Flow for WiMAX AAA Proxy Prepaid Accounting Model



The following section provides a more detailed discussion of the data flow for the WiMAX AAA proxy prepaid accounting shown in [Figure 231 on page 576](#).

1. The NAS sends an Access-Request containing the Username the subscriber entered (and the password depending on the authentication method).
2. SBR Carrier sends an SQL-Request to the subscriber database requesting the subscriber credentials and profile (QoS, bandwidth, and so forth). Refer to [“Configuring SQL Authentication” on page 442](#) for information about how to configure SQL authentication in SBR Carrier.
3. The subscriber database sends back the credentials and profile information.
4. SBR Carrier compares the credentials and if they are acceptable, a final Access-Accept message is sent to the NAS. The Access-Accept message contains the subscriber profile information that is enforced for the lifetime of the subscriber session.
5. Before starting the session, the NAS requests the initial quota by sending a RADIUS Authorize-Only message to SBR Carrier.
6. SBR Carrier conveys the request for initial quota to the PPS, updating the session in the database through SQL accounting (this can also be performed through proxy RADIUS). Refer to [“Configuring SQL Accounting” on page 458](#) for information about how to configure SQL accounting in SBR Carrier.
7. The PPS returns a portion of quota in the SQL response.
8. SBR Carrier conveys that quota to the NAS in the Access-Accept.
9. The NAS starts the session for the subscriber and sends an initial Accounting-Start to SBR Carrier.

NOTE: Steps 5 through 9 can be skipped if initial quota is returned through SQL in the authentication process (steps 1 through 4) and returned in the initial access accept.

10. The NAS sends interim accounting to SBR Carrier.
11. When the initial quota is depleted the NAS requests additional quota through a RADIUS Authorize-Only message to SBR Carrier.
12. SBR Carrier conveys the request for additional quota to the PPS, updating the session in the database through SQL accounting (this can also be performed through proxy RADIUS). For more information about SQL accounting, see [“Configuring SQL Accounting” on page 458](#).

13. The PPS returns a portion of quota in the SQL response.
14. SBR Carrier conveys the additional quota to the NAS in the Access-Accept.
15. The NAS updates the available quota and continues the subscriber session.
16. The NAS sends interim accounting to SBR Carrier.
17. When the additional quota is depleted the NAS requests more quota through a RADIUS Authorize-Only message to SBR Carrier.
18. SBR Carrier conveys the request for additional quota to the PPS, updating the session in the database through SQL accounting (this can also be performed through proxy RADIUS).
19. The PPS returns a portion of the quota in the SQL response, or as in this case if there is no more quota, the PPS returns no quota.
20. SBR Carrier subsequently sends a reject message to the NAS.
21. The NAS terminates (or redirects) the session and sends an Accounting-Stop to SBR Carrier.

Direct PPS Connection Model

In the direct PPS connection model, the NAS performs quota management directly with the PPS; SBR Carrier is not involved in quota management. In this model, SBR Carrier's involvement is limited to assigning the IP-address of the prepaid server responsible for quota management for the subscriber based on the subscriber's profile. For more information about managing profiles in, see ["Administering Profiles" on page 144](#).

Categorizing Access-Requests from Different Devices

Within a WiMAX network, Steel-Belted Radius Carrier may communicate with several types of clients including an ASN-GW, home agent, DHCP server or something else. Steel-Belted Radius Carrier must determine what type of client it is communicating with in order to process the request and send the appropriate response. Access-Requests from each client type must be handled differently and reply attributes on the Access-Accept may differ based on client type. For example, the Access-Request from the home agent contains the **User-Name** attribute, which identifies a Mobile IP session instead of a user. However, the Mobile IP session is associated (paired) with the user. Therefore, when the Access-Request from the home agent arrives, the username is looked up and the reply attributes are processed correctly.

The [RADIUS client-Access-Request-Required-Attributes] section of the **wimax.ini** file list the attributes that must be present in an Access-Request to classify the RADIUS client as a WiMAX ASN-GW, home agent, DHCP server, or something else (Other).

Access-Request from the ASN-GW

An Access-Request from the ASN-GW is a device or user EAP authentication request. This request may contain attributes that indicate, for example, whether the VAAA is assigning the home agent or whether the device authentication phase of EAP has succeeded. Returned attributes may contain, for example: Session-Timeout, Packet-Flow-Descriptor, and QoS-Descriptor.

Access-Request from the Home Agent

An Access-Request from the home agent is a device request, not an authentication request. However, if no mobile session has been established for the pseudo-identifier, then this request is rejected. The RRQ-HA-IP attribute may be received, and if so, then the RRQ-MN-HA-KEY is returned. The Framed-IP-Address attribute may also be returned to the home agent.

Access-Request from the DHCP Server

An Access-Request from the DHCP server is a device request, not an authentication request. However, if no mobile session has been established for the pseudo-identifier, then this request is rejected. The DHCP-RK attribute is returned to the DHCP server.

Categorization Rules

To determine the WiMAX client type, the Access-Request categorization rules are as follows:

- If the Access-Request contains *all* mandatory attributes and no attributes that can *only* be attached to either a home agent or DHCP server, then the WiMAX client type is ASN-GW.
- If the Access-Request contains *all* mandatory attributes and no attributes that can only be attached to either an ASN-GW or DHCP server, then the WiMAX client type is home agent.
- If the Access-Request contains *all* mandatory attributes and no attributes that can only be attached to either an ASN-GW or home agent, then the WiMAX client type is DHCP server.
- If the Access-Request comes from some other client type, it may or may not be allowed. If the Access-Request is allowed, then the WiMAX client type is Other. However if the Access-Request is not allowed, then it is rejected.

For more details about configuring the list of required attributes based on client type, see the [RADIUS client-Access-Request-Required-Attributes] section in the *SBR Carrier Reference Guide*.

Configuring the WiMAX Mobility Module

IN THIS CHAPTER

- Before You Begin | 580
- Configuring the radius.ini File for WiMAX | 581
- Configuring the Home Agent and DHCP Server Assignment | 585
- Configuring WiMAX Clients | 592
- Configuring WiMAX Users and Profiles | 593
- Configuring the EAP Methods for WiMAX | 607

This chapter describes specific steps for configuring Steel-Belted Radius Carrier to support WiMAX requests and points out any specific parameters that must be configured for WiMAX users, profiles and EAP methods. This chapter contains these topics:

Before You Begin

You must install the Steel-Belted Radius Carrier software on a Solaris server and add the license key for the optional WiMAX module before you can configure the server to support WiMAX requests.

NOTE: For details about installing Steel-Belted Radius Carrier software and WiMAX licensing, see the *SBR Carrier Installation Guide*.

NOTE: The **wimax.dct**, **radius.dct**, and **.jdict** dictionary files shipped with Steel-Belted Radius Carrier are configured with the attributes and subattributes necessary for supporting WiMAX in compliance with the WiMAX Forum Network Working Group standards. For more information about dictionary files, see the *SBR Carrier Reference Guide*.

Configuring the radius.ini File for WiMAX

The [Configuration] section of the **radius.ini** file contains parameters that control the basic behavior of Steel-Belted Radius Carrier. For WiMAX, you must edit the parameter listed in [Table 66 on page 581](#).

Table 66: radius.ini [Configuration] Section WiMAX-specific Parameter

Parameter	Function
SendOnlyOneClassAttribute	<p>When a user's identity information is encrypted during authentication, Steel-Belted Radius Carrier uses a special class attribute to pass the user's encrypted identity to an accounting server. Because this typically requires more than one class attribute to be included in the accept response, and because some access points do not support echoing more than one class attribute, you can use the SendOnlyOneClassAttribute parameter to specify how you want Steel-Belted Radius Carrier to forward encrypted user identity information.</p> <p>For WiMAX, set to 1 so Steel-Belted Radius Carrier creates a class attribute containing a class attribute flag, a server identifier, and a transaction identifier. The user identification data that normally is stored in the class attributes is stored in the current sessions table.</p>
AckOnCookieFailure	<p>SBR Carrier does not acknowledge accounting requests for WiMAX when the session does not exist in the current sessions table (CST). To work around this problem, set the AckOnCookieFailure parameter to yes, and SBR Carrier sends an acknowledgement back for every accounting request it receives.</p>

Table 66: radius.ini [Configuration] Section WiMAX-specific Parameter (continued)

Parameter	Function
EnableWiMAXUniqueSessionIdFromNAI	<p>This parameter provides improvements to WiMAX performance and scalability. The improvements include different logic for assigning primary keys to WiMAX tables and for generating the Class attribute in the Access-Accept response.</p> <ul style="list-style-type: none"> • If set to 1, the EnableWiMAXUniqueSessionIdFromNAI parameter is enabled. • If set to 0, the EnableWiMAXUniqueSessionIdFromNAI parameter is disabled. <p>NOTE: When the EnableWiMAXUniqueSessionIdFromNAI parameter is enabled, new session records in the database and the Class attribute in Access-Accept messages are incompatible with the WiMAX logic in previous releases of SBR Carrier. For compatibility with SBR Carrier 7.2.1 and earlier, set EnableWiMAXUniqueSessionIdFromNAI = 0.</p> <p>By default this parameter is enabled.</p> <p>For details on migrating from existing SBRC WiMAX installations and new installations using WiMAX, see the section on <i>Migration and New Installations of SBR Carrier with WiMAX</i> in the <i>Migrating from Previous SBR Releases</i> section of the <i>SBR Carrier Installation Guide</i>.</p>

Configuring Support for Authorize-Only Requests

Authorize-Only requests are supported for WiMAX sessions. You can use Authorize-Only requests to support control of prepaid services. For example, where subscriber usage is metered by time or traffic volume. When subscribers exhaust their prepaid service quota, they can be redirected in mid-session to a prepaid web portal where they can purchase more time or data for their account.

To accept Authorize-Only requests, all of the following conditions must be true:

- The **AuthorizeOnly** parameter in the [Configuration] section of **radius.ini** must=1.
- The **AcceptsAuthorizeOnly** parameter in the [Bootstrap] section of the authentication plug-in you are using (either SQL, LDAP or other) must=1.
- The Access-Request must contain the Service-Type attribute with a value=Authorize-Only.
- The Message-Authenticator must be present and valid in the request.
- A session must already exists in SBR Carrier for the requested AAA session ID (WiMAX).

NOTE: Care must be taken to ensure the **.aut** file used for authentications is separate from the **.aut** file used for Authorize-Only requests, even though the two files may be using the same database table. Also the AuthorizeOnly **.aut** file should not be able to handle or pass any authentications.

For complete details on the **radius.ini** file and its parameter settings, see the *SBR Carrier Reference Guide*.

Enabling the WiMAX Module and Configuring What Request Types Are Supported

This section describes the basic parameters for enabling the WiMAX module and for enabling processing of Access-Requests from the ASN-GW, DHCP server and home agent. For complete details, see the section on *WiMAX Mobility Module Configuration File* in the *SBR Carrier Reference Guide*.

The **wimax.ini** configuration file contains parameters that control basic behavior of the WiMAX mobility module of Steel-Belted Radius Carrier. Depending on your WiMAX deployment and the client types, you can enable which Accept-Requests you want Steel-Belted Radius Carrier to process. [Table 67 on page 583](#) describes the basic parameters for enabling WiMAX and configuring the requests you want the server to process.

Table 67: wimax.ini Parameters

[Section]	Parameter	Function
[Settings]	Enable	<p>Specifies whether the WiMAX mobility module is enabled.</p> <p>Set to 1 to enable WiMAX. This setting is <i>required</i> to use WiMAX.</p> <p>Default value is 0.</p>

Table 67: wimax.ini Parameters (continued)

[Section]	Parameter	Function
[Settings]	AddKeysToAccessAccept	<p>Specifies whether to add the WiMAX-MSK to the Access-Accept.</p> <ul style="list-style-type: none"> • If set to 1, WiMAX-MSK is added to the Access-Accept. • If set to 0, WiMAX-MSK is not added to the Access-Accept. <p>The default value is 0.</p>
[ASNGW-Requests]	Accept-ASNGW-Requests	<p>Specifies whether ASN-GW request processing is enabled.</p> <ul style="list-style-type: none"> • If set to 0, ASN-GW request processing is disabled. If an Access-Request is received from an ASN-GW, the request is rejected. • If set to 1, ASN-GW request processing is enabled. If an Access-Request is received from an ASN-GW, the request is processed. <p>Default value is 0.</p>
[Home-Agent-Requests]	Accept-Home-Agent-Requests	<p>Specifies whether home agent Access-Request processing is enabled.</p> <ul style="list-style-type: none"> • If set to 0, home agent request processing is disabled. If an Access-Request is received from a home agent, the request is rejected. • If set to 1, home agent request processing is enabled. If an Access-Request is received from a home agent, the request is processed. <p>Default value is 0.</p>
[DHCP-Server-Requests]	Accept-DHCP-Server-Requests	<p>Specifies whether DHCP server request processing is enabled.</p> <ul style="list-style-type: none"> • If set to 0, any DHCP server request is rejected. • If set to 1, DHCP server request processing is enabled. <p>Default value is 0.</p>

Configuring the Home Agent and DHCP Server Assignment

There are several ways SBR Carrier can assign the home agent and DHCP server. This section provides a configuration overview for each method.

NOTE: Only IPv4 addresses are supported.

Define the List of Home Agents and DHCP Servers

You need to define a list of NAS-Identifiers for each home agent and DHCP server that sends Access-Requests to Steel-Belted Radius Carrier. These are defined in the [HAs] section and [DHCPServers] section of the **wimax.ini** file. If these lists are not defined, Access-Requests from *any* home agents and DHCP servers are processed.

Configuring Return List Attributes to Assign the Home Agent and DHCP Server

SBR Carrier can assign the home agent and DHCP server addresses by returning the WiMAX-HA-IP-MIP4 and WiMAX-DHCPv4-Server attributes in the Access-Accept message. This is configured by defining these attributes in the return list of a user or profile entry.

The address assignment process and configuration differs depending on whether the server is acting as the HAAA or VAAA. Steel-Belted Radius Carrier can be configured to perform either role.

NOTE: For DHCP server keys to be generated, the DHCP server IP address needs to be returned as part of the ASNGW Access-Accept either through a profile or filter. We recommend using a filter so that based on the user@NAI, the appropriate DHCP server IP address is returned.

Assignment When Acting as the HAAA Server

When you want Steel-Belted Radius Carrier (acting as the HAAA) to assign the home agent and DHCP server IP addresses, you need to configure it to return the WiMAX-hHA-IP-MIP4 and WiMAX-hDHCPv4-Server attributes in the Access-Accept message. This is configured using Web GUI and defining these attributes in the return list of either a user or profile entry. When Steel-Belted Radius Carrier HAAA server receives an Access-Request, it returns these attributes (containing the IP addresses) in the Access-Accept message. For more information about adding these attributes to the return list, see [“Adding a Profile” on page 146](#).

Assignment When Acting as the VAAA Server

Optionally, when Steel-Belted Radius Carrier is acting as the VAAA server, it can assign the IP addresses of the home agent and DHCP server. This requires configuration on both the VAAA and HAAA Steel-Belted Radius Carrier servers. The Steel-Belted Radius Carrier acting as the VAAA must be configured to *add* the WiMAX-hHA-IP-MIP4 and WiMAX-hDHCPv4-Server attributes to the Access-Request before it proxies the request to the Steel-Belted Radius Carrier HAAA. The Steel-Belted Radius Carrier acting as the HAAA must be configured to *allow* the VAAA to assign the home agent and DHCP server.

Configuring the VAAA

To configure the Steel-Belted Radius Carrier VAAA to add the WiMAX-hHA-IP-MIP4 and WiMAX-hDHCPv4-Server attributes to Access-Request messages, you need to specify an attribute filter to be applied to authentication requests before the VAAA proxies the requests to the Steel-Belted Radius Carrier HAAA. Defining this filter is a multi-step process:

1. Define a filter using the **Filters List** page in Web GUI:
 - a. Filter name=wimaxVisitedFilter
 - a. Set the filter Default Rule=Allow
 - b. Add WiMAX-hHA-IP-MIP4 attribute
2. In the **proxy.ini** file, define a proxy realm < the realm name should be the name of the *.pro file> in the [Realms] section.
3. In *.pro file, assign the FilterOut = wimaxVisitedFilter in the [Auth] section.
4. Define the proxy target (HAAA server) using the **Proxy Targets List** page in Web GUI:
5. In *.pro file, set the proxy target name to 1 in the [AuthTargets] section.
6. In *.pro file, set the proxy target name to 1 in the [AcctTargets] section

Configuring the HAAA

You must the configure Steel-Belted Radius Carrier (HAAA) to allow the VAAA to assign the home agent and DHCP server addresses. To do this you set **Allow-VAAA-To-Assign-Home-Agent-And-DHCP-Server=1** in the [ASNGW-Requests] section of the **wimax.ini** file. If this parameter is set to 1 and the VAAA server attaches the WiMAX-hHA-IP-MIP4 attributes to the Access-Request, then the HAAA server will echo the WiMAX-hHA-IP-MIP4 as WiMAX-vHA-IP-MIP4 attribute to the Access-Accept, along with the following additional attributes: vHA-IP-MIP4, MN-vHA-MIP4-KEY, and MN-vHA-MIP4-SPI.

Configuring Statically Weighted Round-Robin Groups to Assign the Home Agent and DHCP Server

The round-robin feature enables you to configure a home agent round-robin group and assign fixed weights to each home agent or DHCP server in the group. SBR Carrier then assigns the home agent or DHCP server to a session based on a weighted round-robin method. This feature load-balances the assignment of home agents in a round-robin fashion by selecting the home agent from a pool of IP addresses.

NOTE: Separate round-robin groups are required to load balance the assignment of both the home agent and DHCP server IP addresses.

You can only use round-robin groups to assign the home agent or DHCP server when Steel-Belted Radius Carrier is acting as the HAAA.

To enable this capability, you need to configure attribute value pools. Attribute value pooling allows for dynamic allocation of attribute values sets, so that attributes needed to configure changeable and complex situations do not have to be assigned in static profiles. Attribute value pools enable Steel-Belted Radius Carrier to assign and return attribute sets dynamically when an Access-Request is processed. Attribute value pooling is configured by using the VSA called Funk-Round-Robin-Group. This attribute is placed in the return list of a user or profile entry to dynamically assign an attribute set from an attribute value pool at log in time.

The value of this attribute must be set to the name of the **.rr** file which defines the attribute value pool. This value is set for a user or profile by using the Web GUI or LDAP Configuration Interface (LCI), or by any other return list mechanism (such as database retrieval).

Each home agent (and DHCP server) is identified by its IP address. The round-robin group functionality used to assign the home agent is based on weights specified in the *round-robin* (***.rr**) file. A *round-robin group* is a group of RADIUS attributes listed under a section in the **.rr** file. Each section is associated with a weight. Selection of a particular section is based on the section's relative weight as shown in this example **.rr** file:

```
[Sets]
Set1=5
Set2=1
Set3=0
Set4=

[Set1]
;<Attribute_Name>=<Attribute_Value>
WiMAX-hHA-IP-MIP4=71.14.1.4

[Set2]
WiMAX-hHA-IP-MIP4=71.14.1.5

[Set3]
WiMAX-hHA-IP-MIP4=71.14.1.6
[Set4]
WiMAX-hHA-IP-MIP4=71.14.1.7
```

For more information, see [“Attribute Value Pooling” on page 81](#) in this guide, and the **sample.rr** file, in the *SBR Carrier Reference Guide*.

NOTE: You cannot use round-robin group load balancing to assign IP addresses to the home agent or DHCP server when SBR Carrier is acting as the VAAA.

On receiving the SIGHUP (1) signal, SBR Carrier reads the configuration files and updates the round-robin groups. To use the smart dynamic home agent assignment feature, you create a file that defines the round-robin groups and another file that defines the weights for the round-robin group. The smart dynamic home agent assignment feature works by creating a configuration file for the round-robin group and creating round-robin groups as specified in those files. You can assess the load status of the home agents in your network and populate the associated configuration files in a way that balances the load across home agents. By sending a SIGHUP (1) signal to SBR Carrier, you can dynamically update the configuration. On receiving the SIGHUP (1) signal, SBR Carrier reads the configuration files and updates the round-robin groups.

By using dynamically updated round-robin groups, SBR Carrier can load balance the IP address assignment of multiple home agents and DHCP servers are in the network.

NOTE: You can only use these round-robin methods when SBR Carrier is acting as the HAAA.

Configuring the Smart Dynamic Home Agent Assignment Feature

The smart dynamic home agent assignment feature works by reading various configuration files and creating round-robin groups as specified in those files. You can assess the load status of the home agents in your network and populate the associated configuration files in a way that balances the load across home agents. When SBR Carrier receives a SIGHUP (1) signal, it reads these configuration files and updates the round-robin groups.

Smart Dynamic Home Agent Assignment Configuration Overview

SBR Carrier uses two files to configure the smart dynamic home agent assignment feature. The first file (**dynamic_ha.ini**) defines the home agent IP addresses for the round-robin groups. A second file defines, and dynamically updates the weights for the round-robin groups.

dynamic_ha.ini File

The **dynamic_ha.ini** file defines the home agent IP addresses for the round-robin groups. For example:

```
[Osaka]
71.14.1.6
71.14.1.22

[Tokyo]
71.14.1.7
71.14.1.22
```

Each section in the **dynamic_ha.ini** file contains a list of IP addresses that form a round-robin group. The file can contain any number of sections and therefore can define any number of round-robin groups. This file is *not* dynamically updated. A single IP address *may* be contained in more than one section.

The Funk-hHA-IP-MIP4-Group and Funk-vHA-IP-MIP4-Group attributes indicate which group (visited or home) assigns the home agent. These attributes contain the name of the section in **dynamic_ha.ini** file that is used. The SBR Carrier WiMAX subsystem determines whether to use the visited or the home attribute. This attribute is available to filters and profiles. These attributes are stripped and do not go out over the network.

Customer-Written Dynamically Updated File

A customer-written file with a format similar to the **.rr** file is dynamically updated and contains pairs of IP addresses and weights. The file is read by SBR Carrier upon the receipt of a signal (either SIGHUP (1) or SIGUSR2 (17), as defined in **update.ini** file).

The **HA-Dynamic-Addr-Weight-File** parameter in the [Settings] section of the **wimax.ini** defines the name and path to this file as follows:

```
[Settings]
...
HA-Dynamic-Addr-Weight-File = <path/filename>
...
```

Use of the smart dynamic home agent assignment feature requires you to write this file and the application that:

- Monitors home agents
- Based on load and availability, determines the list of IP addresses and their associated weights
- Writes to the file
- Immediately signals SBR Carrier servers when a home agent does down

This file uses the following format:

```
[IPv4]
<HA IP address> = <Weight>
```

For example:

```
[IPv4]
71.14.1.4 = 5
71.14.1.5 = 1
71.14.1.6 = 0
71.14.1.7
```

This example contains four *round-robin sets*, where each *set* contains an IP address. The attribute that carries that IP address is not defined in this file, but is either **WiMAX-hHA-IP-MIP4** or **WiMAX-vHA-IP-MIP4**, as determined by the SBR Carrier WiMAX subsystem. Because the WiMAX subsystem determines which attribute to use (home or visited), this mechanism is used only for WiMAX home agent assignment and is not a general alternative to **.rr** files.

A '0' as the weight indicates that the home agent is offline and not to use it. The default weight is '0', so an IP address with an empty weight field is not used.

[HAs] Section of the wimax.ini File

The [HAs] section of the **wimax.ini** file contains a list of IP addresses for allowed home agents (home or visited).

Access-Requests from home agents not listed in the [HAs] section of **wimax.ini** file are rejected. Values under the [HAs] section can be *either*:

- All NAS-Identifiers
- All IPv4 addresses

When using the smart dynamic home agent assignment feature, the values under the [HAs] section *must* be IPv4 addresses.

If the [HAs] section contains one or more IPv4 addresses, and if an IP address in the dynamic file is not in the [HAs] section, then the weight of that IP address in the dynamic file will be forced to zero. In other words, if the [HAs] section contains one or more IPv4 addresses then any IP address *not* under the [HAs] section will *not* be assigned.

If the [HAs] section contains no entries then the weights are read from the file without modification.

Operation of the Smart Dynamic Home Agent Assignment Feature

This section describes the processing that occurs in Steel-Belted Radius Carrier for the Smart Dynamic Home Agent Assignment feature.

Processing on Startup

On startup, SBR Carrier:

1. Reads the **dynamic_ha.ini** file.
2. Reads the dynamically updated file as specified in the **HA-Dynamic-Addr-Weight-File** parameter in the [Settings] section of the **wimax.ini** file.
3. Reads the [HAs] section of the **wimax.ini** file.
4. If an IP address specified in the **HA-Dynamic-Addr-Weight-File** is not listed in the **dynamic_ha.ini** file, or is not in the [HAs] section of the **wimax.ini** file, then it is ignored.
5. One *round-robin processing object* is created for each group that contains one or more addresses. If a group contains IP addresses, a warning is logged.

Processing on a Signal

Upon receiving a signal from the customer-written application, SBR Carrier:

1. Reads the dynamically updated file as specified in the **HA-Dynamic-Addr-Weight-File** parameter in the [Settings] section of the **wimax.ini** file.
2. Reads the [HAs] section of the **wimax.ini** file.
3. If an IP address specified in the **HA-Dynamic-Addr-Weight-File** is not listed in the **dynamic_ha.ini** file, or is not in the [HAs] section of the **wimax.ini** file, then it is ignored.
4. One *round-robin processing object* is created for each group that contains one or more addresses. If a group contains IP addresses, a warning is logged. The *round-robin processing objects* are reference-counted so that any object in use at the time of the signal is not destroyed until after it is no longer in use.

Access-Request Processing

This section describes the Access-Request processing when SBR Carrier is acting as either the VAAA or HAAA server.

Access-Request Processing When Acting as the VAAA

If SBR Carrier determines that it must proxy the Access-Request, then it is acting as a VAAA.

When acting as the VAAA and processing the initial Access-Request:

1. The **dynamic_ha.ini** section name (group name) is read from the Funk-vHA-IP-MIP4-Group attribute. The Funk-vHA-IP-MIP4-Group needs to be attached using the SBR Carrier filter capability.
2. The value of the WiMAX-hHA-IP-MIP4 attribute is obtained from the named *round-robin processing object*.

3. WiMAX-hHA-IP-MIP4 is attached to the Access-Request.
4. The Funk-vHA-IP-MIP4-Group attribute is stripped from the Access-Request.

When processing the final Access-Accept:

1. *If a WiMAX-vHA-IP-MIP4 and WiMAX-MN-vHA-MIP4-Key are attached to the Access-Accept message returned from the downstream HAAA server, and if the WiMAX-vHA-IP-MIP4 value is the same as the value assigned by the VAAA server, then:*
 - a. The VAAA server allows WiMAX-vHA-IP-MIP4 and WiMAX-MN-vHA-MIP4-Key to be passed back to the ASN-GW in the Access-Accept.
 - b. The VAAA server attaches WiMAX-vHA-RK and related attributes to the Access-Accept.
2. *Else WiMAX-vHA-IP-MIP4 and WiMAX-MN-vHA-MIP4-Key are stripped from the Access-Accept.*

Access-Request Processing When Acting as the HAAA

If the SBR Carrier determines that it does *not* need to proxy the Access-Request then it is acting as a HAAA.

When processing the final Access-Accept:

1. *If WiMAX-hHA-IP-MIP4 was attached to the Access-Request and if configured to allow the VAAA to assign the home agent then attach the WiMAX-vHA-IP-MIP4 and WiMAX-MN-vHA-MIP4-Key attribute to the Access-Accept.*
2. The **dynamic_ha.ini** section name (group name) is read from the Funk-hHA-IP-MIP4-Group attribute (which is attached to the Access-Accept).
3. The value of the WiMAX-hHA-IP-MIP4 attribute is obtained from the named *round-robin processing object*.
4. WiMAX-hHA-IP-MIP4 is attached to the Access-Accept.
5. The Funk-hHA-IP-MIP4-Group attribute is stripped from the Access-Accept.

Configuring WiMAX Clients

You must identify the devices in your network that are considered WiMAX clients to the Steel-Belted Radius Carrier and define them as clients using Web GUI. The following devices are considered clients:

- ASN-GW
- Home agent
- DHCP server

The only unique parameter used when configuring WiMAX clients is setting the Make or model option=WiMAX. For complete details on configuring clients in Steel-Belted Radius Carrier, see [“Administering RADIUS Clients and Client Groups” on page 104](#).

Configuring WiMAX Users and Profiles

To support WiMAX you need to configure a return list for either a user entry or profile entry that includes the attributes in [Table 68 on page 593](#):

Table 68: Mandatory Return List Attributes for WiMAX

Attribute	Description
WiMAX-hHA-IP-MIP4	Specifies the IP address for the home agent, and is also used as input to the formula for generating the keys associated with the session.
Session-Timeout	The session-timeout attribute is used as the lifetime for the keys.
WiMAX-Capabilities	Specifies the WiMAX capabilities the server supports for the session. You must also specify the associated subattributes for each capability you want to support, see “Configuring the WiMAX-Capabilities Negotiation” on page 594 .

You can optionally specify the attribute described in [Table 69 on page 594](#) in the return list.

Table 69: Optional Return List Attribute for WiMAX

Attribute	Description
WiMAX-hDCHP-Server	<p>Optionally, you can add the WiMAX-hDCHP-Server attribute to specify the IP address for DHCP server in the return list.</p> <p>If the WiMAX-hDCHP-Server attribute is attached to the Access-Accept, then Steel-Belted Radius Carrier generates and attaches the following additional attributes to the Access-Accept:</p> <ul style="list-style-type: none"> • Wi-MAX-hDHCP-RK • Wi-MAX-RK-Key-ID • Wi-MAX-RK-Lifetime

For complete details on configuring user and profile entries with return list attributes, see [“Configuring the WiMAX Mobility Module” on page 580](#) and [“Administering Profiles” on page 144](#).

Configuring the WiMAX-Capabilities Negotiation

To configure WiMAX capabilities negotiation, you need to add the WiMAX-Capabilities attribute and subattributes to the return list of a user entry or profile entry. You can define the following subattributes (capabilities):

- WiMAX-Release attribute
- Accounting-Capabilities attribute
- Hotlining-Capabilities attribute
- Idle-Mode-Notification-Capabilities attribute

To enable support for a particular capability, add the subattribute to the return list, and enable the Echo option for the subattribute. When Steel-Belted Radius Carrier receives the subattribute (capability) in the Access-Request, it returns the subattribute in the Access-Accept indicating the capability is supported for the session. If you do not want Steel-Belted Radius Carrier to support a particular capability, do not enable the Echo option for it. If Steel-Belted Radius Carrier receives an Access-Request with the subattribute, it does not return the subattribute in the Access-Accept, indicating the capability is not be supported for the session. If a subattribute (capability) was never sent in the Access-Request, then it cannot be returned in the Access-Accept. Absence of a subattribute in the Access-Request indicates the device (ASN-GW or home agent) does not support the capability.

If you enable Echo on the WiMAX-Capability parent attribute, you cannot add subattributes. The Add Child button is disabled. In this case, Steel-Belted Radius Carrier echoes back whatever WiMAX capabilities it receives in the Access-Request message.

For more details on each of the WiMAX capabilities, see [“WiMAX-Capability Attribute” on page 565](#).

For complete details on adding subattributes to the return list, see [“Adding Subattributes to a Structured Attribute” on page 128](#).

Example Configuration for New Session Hotlining

This section provides an example configuration for new session hotlining. Because this example uses the EAP-TTLS authentication method, you need to create both a request and response filter. Both filters are created using the Web GUI. In this example the subattribute values are retrieved from an LDAP database (`ldapauth.aut` file).

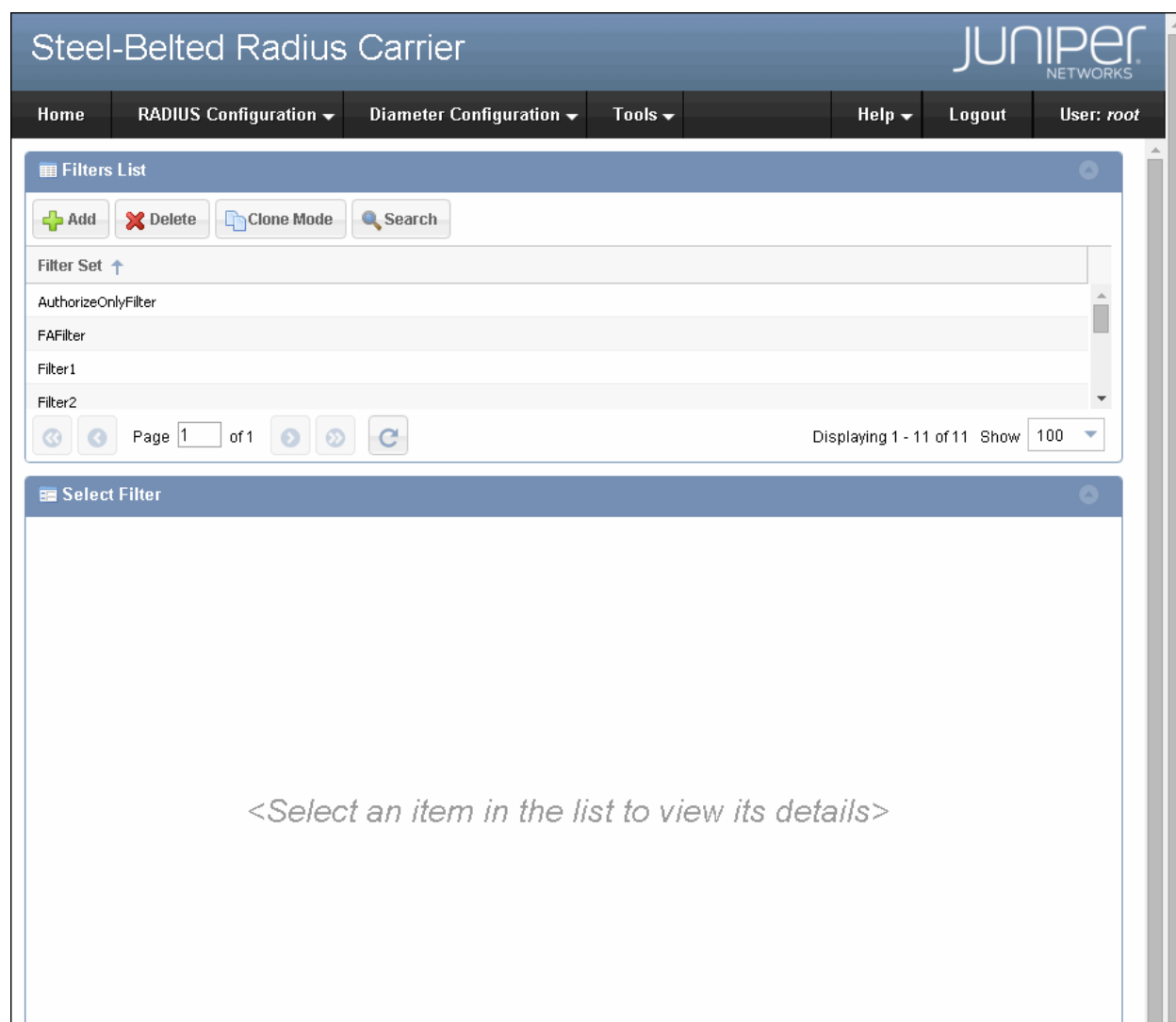
Configuring the Filters

To configure request and response filters using the Web GUI:

1. Select **RADIUS Configuration > Filters**.

The **Filters List** page (Figure 232 on page 596) appears.

Figure 232: Filters List Page—Session Hotlining Filter Configuration



2. Click **Add**.

The **Create Filter** pane (Figure 233 on page 597) appears.

Figure 233: Adding New Session Hotlining Filter

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

Filters List

Create Filter

Name: WIMAXHotlineFilter

Description: New session hotlining filter

Default Rule: ☐ Allow ☒ Exclude

Rules

Rule Type	Attribute	Value	Replacement Attribute	Replacement Value	Script Name

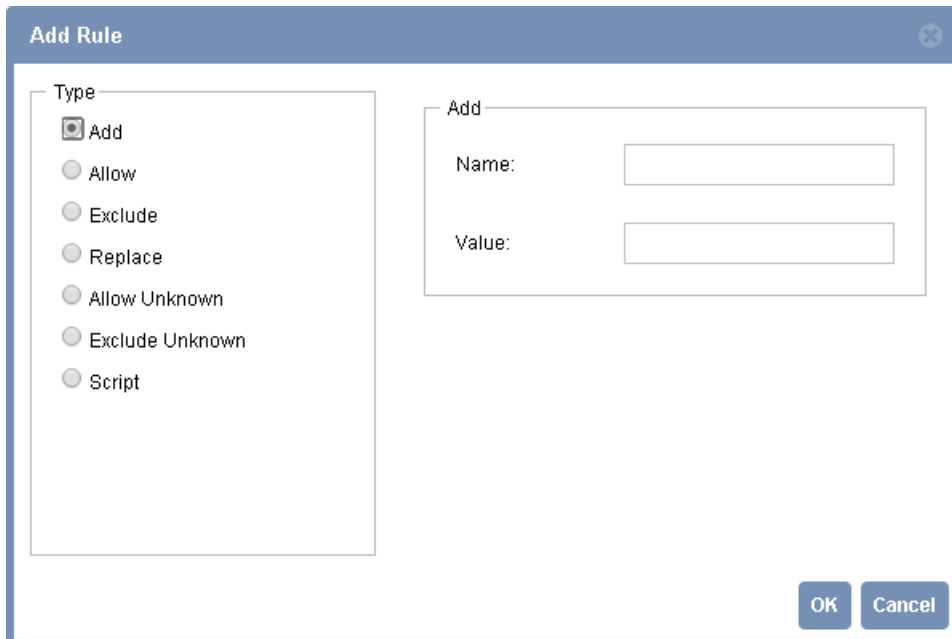
Add Edit Delete

Save Clear Cancel

3. In the **Name** field, enter the filter name as **WIMAXHotlineFilter**.
4. Select the **Exclude** option button.
5. Click **Add** in the **Rules** area.

The **Add Rule** dialog box ([Figure 234 on page 598](#)) appears.

Figure 234: Adding Attributes and Values to Session Hotlining Filter



The image shows a dialog box titled "Add Rule" with a close button in the top right corner. It is divided into two main sections. The left section, labeled "Type", contains a list of radio button options: "Add" (which is selected), "Allow", "Exclude", "Replace", "Allow Unknown", "Exclude Unknown", and "Script". The right section, labeled "Add", contains two text input fields: "Name:" and "Value:". At the bottom right of the dialog are "OK" and "Cancel" buttons.

6. Select the **Add** option button.
7. Add the following attribute names and values to the filter:
 - **WiMAX-Capability.Values.Hotlining-Capabilities.Profile-based** attribute with the value set to 01.
 - **WiMAX-Capability.Values.Hotlining-Capabilities.Rule-based-ByNAS-Filter** attribute with the value set to 01.
8. Click **OK**.

The **Rules** area in the **Create Filter** pane ([Figure 235 on page 599](#)) displays the updated lists of selected rules.

Figure 235: Hotlining Capabilities Filter

Steel-Belted Radius Carrier

JUNIPER NETWORKS

HomeRADIUS ConfigurationDiameter ConfigurationToolsHelpLogoutUser: root

Filters List

Create Filter

Name:WIMAXHotlineFilter

Description:New session hotlining filter

Default Rule:☐ Allow ☒ Exclude

Rules

Rule Type	Attribute	Value
Add	WIMAX-Capability.Values.Hotlining-Capabilities.Profile-based	01
Add	WIMAX-Capability.Values.Hotlining-Capabilities.Rule-based-ByNAS-Filter	01

+

+

AddEditDelete

Save

Clear

Cancel

9. Click **Save** to save the filter configuration.
- The **Filters List** page (Figure 232 on page 596) displays an updated list of filter entries.
10. Click **Add** in the **Filters List** page (Figure 232 on page 596) to add a TTLS-Accept filter.
- The **Create Filter** pane (Figure 236 on page 600) appears.

Figure 236: Adding TTLS-Accept Filter

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

Filters List

Create Filter

Name:

Description:

Default Rule: ☒ Allow ☐ Exclude

Rules

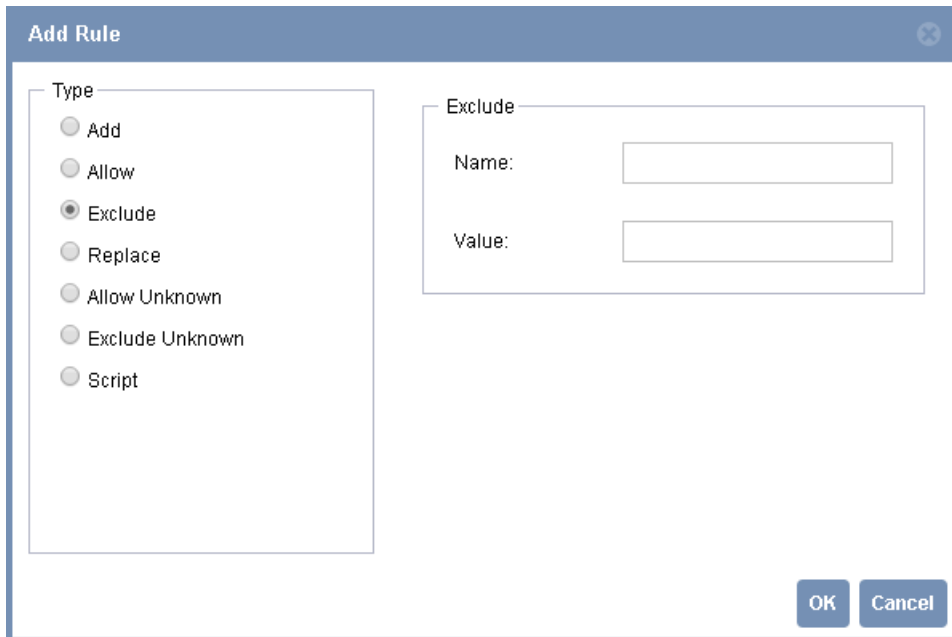
Rule Type	Attribute	Value	Replacement Attribute	Replacement Value	Script Name
-----------	-----------	-------	-----------------------	-------------------	-------------

Add Edit Delete

Save Clear Cancel

- 11. In the **Name** field, enter the filter name as **ttls_accept**.
 - 12. Select the **Allow** option button.
 - 13. Click **Add** in the **Rules** area.
- The **Add Rule** dialog box (Figure 237 on page 601) appears.

Figure 237: Add Rule for TTLS-Accept Filter



The image shows a dialog box titled "Add Rule" with a close button in the top right corner. On the left, under the heading "Type", there is a list of radio button options: "Add", "Allow", "Exclude" (which is selected), "Replace", "Allow Unknown", "Exclude Unknown", and "Script". On the right, under the heading "Exclude", there are two text input fields labeled "Name:" and "Value:". At the bottom right of the dialog are "OK" and "Cancel" buttons.

14. Select the **Exclude** option button.

15. Add the following attribute names to the filter:

- a. Class
- b. EAP-Message
- c. MS-MPPE-Recv-Key
- d. MS-MPPE-Send-Key
- e. MS-CHAPV2-Success

16. Click **OK**.

The **Rules** area in the **Create Filter** pane ([Figure 238 on page 602](#)) displays the updated lists of selected rules.

Figure 238: TTLS-Accept Filter

Steel-Belted Radius Carrier

JUNIPER NETWORKS

HomeRADIUS Configuration ▼Diameter Configuration ▼Tools ▼Help ▼LogoutUser: root

Filters List

Create Filter

Name:ttls_accept

Description:TTLS-Accept filter

Default Rule:☒ Allow ☐ Exclude

Rules

Rule Type	Attribute	Value
Exclude	Class	
Exclude	EAP-Message	
Exclude	MS-MPPE-Recv-Key	
Exclude	MS-MPPE-Send-Key	
Exclude	MS-CHAPV2-Success	

AddEditDelete

Save

Clear

Cancel

- 17.Click **Save** to save the filter configuration.
- The **Filters List** page (Figure 232 on page 596) displays an updated list of filter entries.
- 18.Select **RADIUS Configuration > Authentication Policies > EAP Methods**.
- The **EAP Methods List** page (Figure 239 on page 603) appears.

Figure 239: EAP Methods List Page—Session Hotlining Filter Configuration

Steel-Belted Radius Carrier JUNIPER NETWORKS

Home RADIUS Configuration ▾ Diameter Configuration ▾ Tools ▾ Help ▾ Logout User: root

EAP Methods List

Apply Reset Refresh

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Selected EAP Method: EAP-TTLS

☐ Enable EAP-TTLS Method

Client Certificate Validation

Request Filters

Response Filters

Session Resumption

Inner Authentication

Advanced Server Settings

Enable CRL Checking: ☐

Require Client Certificate: ☐

Retrieval Timeout:

Expiration Grace Period:

Allow Missing CDP Attribute: ☒

CRL Cache Timeout Period: ☐

Default LDAP Server Name:

Include Certificate Info: ☐

Save Reset Cancel

19. Select **EAP-TTLS**.

20. Click the **Request Filters** tab (Figure 240 on page 604).

Figure 240: Request Filters Tab—Session Hotlining Filter Configuration

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and a user profile for 'root'. The main content area is titled 'EAP Methods List' and contains a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Below the table, the 'Selected EAP Method: EAP-TTLS' section is active. It includes a checkbox for 'Enable EAP-TTLS Method' and a tabbed interface with the following tabs: Client Certificate Validation, Request Filters, Response Filters, Session Resumption, Inner Authentication, and Advanced Server Settings. The 'Request Filters' tab is selected, showing the following configuration options:

- Transfer Outer Attribs to New:** A checked checkbox next to a dropdown menu showing 'WiMAXHotlineFilter'.
- Transfer Outer Attribs to continue:** An unchecked checkbox next to an empty dropdown menu.
- Edit New:** An unchecked checkbox next to an empty dropdown menu.
- Edit Continue:** An unchecked checkbox next to an empty dropdown menu.

At the bottom of the interface, there are buttons for Save, Reset, and Cancel.

21. Select the **Transfer Outer Attribs to New** check box and select **WiMAXHotlineFilter** from the **Transfer Outer Attribs to New** list.
22. Click the **Response Filters** tab (Figure 241 on page 605).

Figure 241: Response Filters Tab—Session Hotlining Filter Configuration

The screenshot shows the Steel-Belted Radius Carrier web interface. At the top, there's a navigation bar with 'Home', 'RADIUS Configuration', 'Diameter Configuration', 'Tools', 'Help', 'Logout', and 'User: root'. Below this is the 'EAP Methods List' section with buttons for 'Apply', 'Reset', and 'Refresh'. A table lists EAP methods: EAP-TLS, EAP-TTLS (selected), EAP-PEAP, and EAP-TLS Helper, all with a status of 'Disabled'. Below the table, the 'Selected EAP Method: EAP-TTLS' section is shown. It includes a checkbox for 'Enable EAP-TTLS Method' and several tabs: 'Client Certificate Validation', 'Request Filters', 'Response Filters' (active), 'Session Resumption', 'Inner Authentication', and 'Advanced Server Settings'. Under the 'Response Filters' tab, there are two sections: 'Transfer Inner Attribs To Accept' (checked) and 'Transfer Inner Attribs To Reject' (unchecked). A dropdown menu is open for the 'Transfer Inner Attribs To Accept' section, showing a list of filters: FAFilter, HASKeyFilter, MNHAFilter, HAUserFilter, SIPFilter, AuthorizeOnlyFilter, SimpleFilter, **ttls_accept** (selected), ttls_reject, ram, tnm_demo_vlan, tnm_demo_vlan_1, ram_1, and ram_2. At the bottom of the configuration area are 'Save', 'Reset', and 'Cancel' buttons.

23. Select the **Transfer Inner Attribs To Accept** check box and select **ttls_accept** from the **Transfer Inner Attribs To Accept** list.

24. Click **Save** to save the configuration.

25. In the **wimax.ini** file, set the **ASNGW-Accept-Filter** parameter in the [ASN-GW-Requests] section to **ttls_accept**.

Configuring the LDAP Authentication File

For this example, the **ldap.aut** file shipped with Steel-Belted Radius Carrier is modified to retrieve the values of the subattributes.

```
[Bootstrap]
LibraryName=ldapauth.so
Enable=1
InitializationString=WIMAX_HOTLINE

[Settings]
```

```

MaxConcurrent=1
Timeout=20
ConnectTimeout=25
QueryTimeout=10
WaitReconnect=2
MaxWaitReconnect=360
; BindName=uid=<User-Name>, ou=sales, o=bigco.com
; BindName = cn=Manager,o=sbrsim, c=US
LogLevel = 2
UpperCaseName = 0
PasswordCase=original
PasswordFormat = 0
; Search = DoLdapSearch
SSL = 0
MaxScriptSteps = 1000
ScriptTraceLevel =2
;FilterSpecialCharacterHandling = 0

[NDS]
;Enable = 0
;AllowExpiredAccountsForUsers = 0
;ProfileForExpiredUsers = profile1
;AllowGraceLoginsForUsers = 1
;ProfileForGraceLoginUsers = profile2

[Server]
s1=

[Server/s1]
Host=172.28.84.27
Port = 21002
BindName=cn=Directory Manager
BindPassword=sbrcarrier

[Failure]
;Accept=0
;Profile=xyz
;FullName=Remote User

[Request]
%UserName = UserName
Juniper-WiMAX-Client-Type =clientType
WiMAX-Capability.Values.Hotlining-Capabilities.Rule-based-By-NAS-Filter= hotlineCaps_nas
WiMAX-Capability.Values.Hotlining-Capabilities.Profile-based= hotlineCaps_profile

```

```
[Response]
%Password = telephonenumber
%Profile = profile
@WiMAX-Hotline-Indicator= hotlineIndicator
@WiMAX-Hotline-Profile-ID= hotlineProfile
@WiMAX-Hotline-Session-Timer= hotlineTimer
```

```
[Search/SubscriberRecord]
;Base = o=sbrsim, c=US
Base =dc=carrier,dc=spgma,dc=juniper,dc=net
Scope = 2
Filter = uid=<LDAPRecordKey>
Attributes = SubscriberAttr
Timeout = 20
%DN = dn
```

```
[Attributes/SubscriberAttr]
profile
telephonenumber
wimax-hotline-profile-id
wimax-hotline-session-timer
wimax-hotline-indicator
```

Configuring the EAP Methods for WiMAX

The following EAP authentication protocols can be used for WiMAX sessions:

- EAP-TLS (Transport Layer Security) protocol
- EAP-TTLS (Tunneled Transport Layer Security) protocol
- EAP-AKA (Authentication and Key Agreement) protocol

NOTE: The EAP-AKA protocol requires a license for the optional SIM authentication module.

For WiMAX the only specific parameter that needs to be configured for these EAP authentication protocols is ensuring that the Return MPPE Keys check box is disabled. Other than disabling this option, you do not need to configure any other special parameters to use these authentication methods for WiMAX sessions.

Follow the procedures described in [“Setting Up EAP Methods” on page 253](#) for the EAP-TLS and EAP-TTLS protocols. For the EAP-AKA protocol, follow the procedures described in [“Configuring the WiMAX Mobility Module” on page 580](#) in this guide and the section on *Optional Authentication Module Configuration Files* in the *SBR Carrier Reference Guide*.

EAP-TTLS Secondary Authentication Support

For WiMAX applications using EAP-TTLS, you can perform a secondary authentication that verifies the MAC address field in the client certificate against the Calling-Station-Id in the outer Access-Request. If they do not match, the request is rejected. This checking is enabled by setting the **Check-CN-In-TTLS-Client-Certificate** parameter in the **wimax.ini** file. For more information, see the *WiMAX Mobility Module Configuration File* section in the *SBR Carrier Reference Guide*.

8

PART

Optional Session State Register (High Availability) Module for a Clustered Environment

[Session State Register Overview](#) | **610**

[Session State Register Administration](#) | **628**

Session State Register Overview

IN THIS CHAPTER

- [SSR Cluster Overview | 610](#)
- [Data Replication Between Two Different or Remote SSR Clusters | 611](#)
- [SSR Cluster Concepts and Terminology | 615](#)
- [Supported SBR Carrier SSR Cluster Configurations | 621](#)
- [Session State Register Database Tables | 625](#)

This chapter presents general information about the Steel-Belted Radius Carrier high-availability option. It contains information about the servers for the SSR module. These topics are in this chapter:

SSR Cluster Overview

A standalone SBR Carrier server runs the entire AAA process using a local session database process on a single machine.

The SSR module helps you implement a highly reliable and highly available session database. The SSR module enables the AAA functions that are running on two or more RADIUS servers in a clustered environment to use a common session database, making the RADIUS servers stateless.

The RADIUS server and session database collaborate to create a single virtual AAA server that provides:

- High availability
- State preserved during failover between front-end SBR Carrier nodes
- Application session awareness
- Centralized IP address management
- Centralized concurrency management (if the optional Concurrency and Wholesale Module is installed on all SBR Carrier servers)

To work efficiently and to provide redundancy, a production SSR cluster must be built on a fast, isolated, and redundant network infrastructure. It must include multiple hosts and nodes so a single node failure cannot prevent the cluster from operating.

Because the SSR database contains detailed attributes for each session it serves, it is a logical source for applications that query the AAA environment for correlative and verification purposes. The default SSR database schema addresses standard requirements for most wireless and wireline service providers. It can also be customized to address unique requirements or to support other applications. For example, a video portal or streaming server would know the IP address of a session but needs to discover the user identity attached to the session for billing or personalization. The video gateway can query the SSR database in real time to correlate information about any active session.

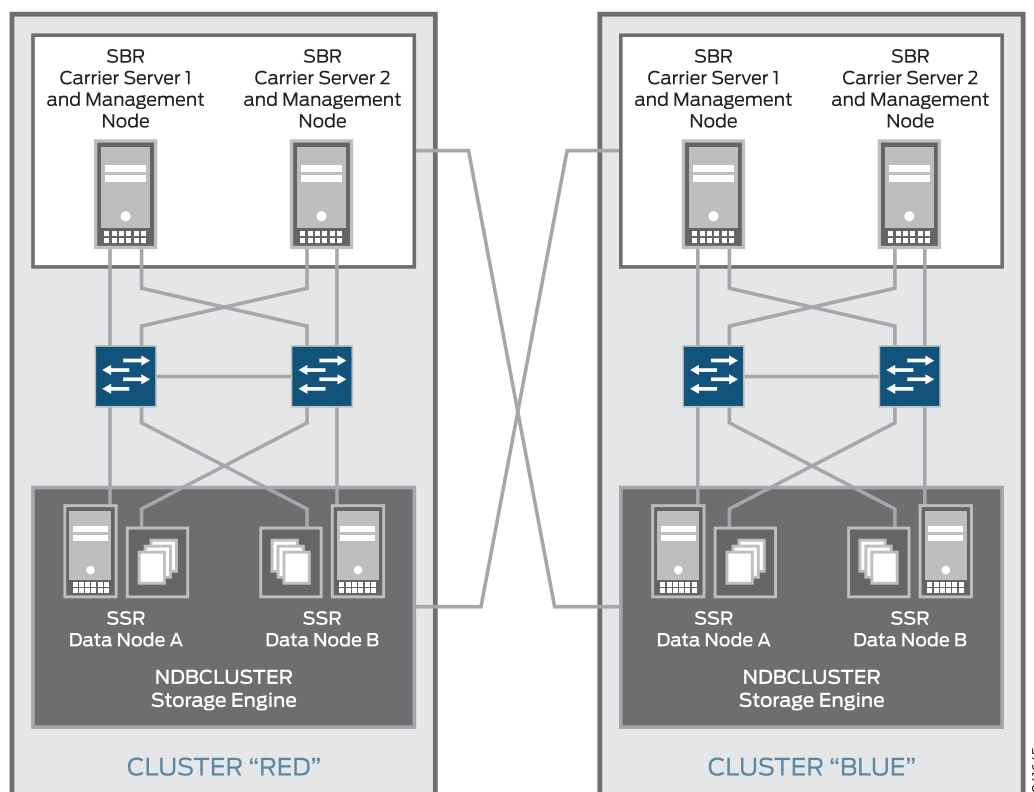
RADIUS Class Attributes are part of the RADIUS standard, but not all network access servers (NAS) fully support them. In order for the SSR to reliably track sessions throughout each session's lifetime, the network NASs must support the Radius Class Attribute.

NOTE: During a multinode failure of the cluster, the stability of SBR Carrier may be affected. You can use the auto-restart module (radiusd script) to mitigate such events.

Data Replication Between Two Different or Remote SSR Clusters

The SBR Carrier software supports the Geo-redundancy feature that allows data to be replicated among nodes of two remote SSR clusters located in different geographical locations. The Geo-redundancy feature provides a consolidated session store, enabling all sessions from a geographically diverse system to be accessed from a single database. [Figure 242 on page 612](#) represents the Geo-redundancy environment. The SBR Carrier uses the Geo-redundancy plug-in to optimize data replication. The Geo-redundancy plug-in is responsible for sending and receiving replication packets between clusters.

Figure 242: Geo-redundancy–Data Replication Between Two Remote SSR Clusters



NOTE: The Geo-redundancy feature is supported between two SSR clusters only if both SSR clusters have the same CST schema.

NOTE: When SBR Carrier is configured with the Geo-redundancy feature, SBR Carrier supports up to 3000 transactions per second at 0.10 latency without affecting accounting.

Configuring the Client Component and Server Component

The client component refers to a local cluster, which replicates session information to a remote cluster that is located geographically apart from the local cluster. For example, if the session information of Cluster RED is to be replicated to Cluster BLUE as shown in [Figure 242 on page 612](#), then Cluster RED is considered as the Geo-redundancy client for Cluster BLUE. The client component is enabled by default in RADIUS front ends of the clusters. You can configure the client component by defining the parameters in the **[ClientSettings]** section of the **georedSess.ses** file. For complete details about the parameters of the **[ClientSettings]** section, see the *[ClientSettings] Section* in the *SBR Carrier Reference Guide*.

The server component refers to a remote cluster to which the session information is replicated from the local cluster. For example, if the session information of Cluster RED is to be replicated to Cluster BLUE as shown in [Figure 242 on page 612](#), then Cluster BLUE is considered as the Geo-redundancy server for Cluster RED. The server component is enabled by default in RADIUS front ends of the clusters. You can configure the server component by defining the parameters in the **[ServerSettings]** section of the **georedSess.ses** file. For complete details about the parameters of the **[ServerSettings]** section, see the *[ServerSettings] Section* in the *SBR Carrier Reference Guide*.

To enable a Geo-redundancy server to match incoming accounting requests to replicated session records using the Class attribute (transaction ID), you must configure the **spi.ini** file on each SBR Carrier server with the IP addresses of all the SBR Carrier servers that might receive accounting requests. For more information about how to configure the **spi.ini** file, see the *spi.ini File* section in the *SBR Carrier Reference Guide*.

NOTE: The Geo-redundancy feature is disabled if both the client and server components are disabled.

List of CST Fields That Can Be Replicated Across SSR Clusters

The following CST fields can be replicated across SSR clusters:

- System core fields:
 - Sbr_UniqueSessionId
 - Sbr_CreationTime
 - Sbr_ExpirationTime
 - Sbr_Ipv4Address
 - Sbr_Ipv6Address
 - Sbr_IpPoolOrdinal
 - Sbr_NasName
 - Sbr_SessionState
 - Sbr_UserConcurrencyId
 - Sbr_3gpp2ReqType
 - Sbr_WimaxClientType
 - Sbr_3gpp2HomeAgentAddr

- Sbr_AcctAutoStop
- Sbr_ClassAttribute
- System optional fields:
 - Sbr_UserName
 - Sbr_AcctSessionId
 - Sbr_TransactionId
 - Sbr_NasPortType
 - Sbr_NasPort
 - Sbr_CallingStationId
 - Sbr_CalledStationId
 - Sbr_MobileCorrelationId
 - Sbr_Ipv6InterfaceId
 - Sbr_Ipv6Prefix
 - Sbr_NasIpv4Address
 - Sbr_NasIpv6Address
 - Sbr_NasClientName
 - Sbr_NasDeviceModel
 - Sbr_ProxyState
 - Sbr_ProxyRealm
- Admin radattr fields:
 - FunkOuterUserName
 - AcctMultiSessionId

Possible Uses and Limitations of the Geo-redundancy Feature

This section lists the possible uses and limitations of the Geo-redundancy feature.

The Geo-redundancy feature could be used to perform the following tasks:

- Consolidating session information across different SSR clusters in a single entity
- Replicating selective information about a user session from a CST
- Applying policies on the basis of the roaming facility
- Regulating traffic on the basis of volume

- Limiting the number of sessions per user who roams between multiple clusters
- Backing up intelligent data without using a separate physical backup cluster
- Replicating session information from multiple standalone SBRs to a single SSR cluster
- Replicating data between Linux-based SSR and Solaris-based SSR, or vice versa
- Supporting peer-to-peer cluster replication and hierarchical primary cluster replication

The Geo-redundancy feature has the following limitations:

- The Geo-redundancy feature does not support multiple-to-multiple cluster replication.
- IPv6 hosts cannot be specified as the Geo-redundancy server or client.
- The following information cannot be replicated:
 - User concurrency table
 - IP pool and address information
 - SIM session information
 - Custom SSR fields
 - Worldwide Interoperability for Microwave Access (WiMAX)

SSR Cluster Concepts and Terminology

A Session State Register cluster has both a physical and logical organization. The physical elements are *servers*. The logical elements are *nodes*. The two terms should not be used as synonyms; they are not interchangeable.

Session State Register Servers

The Session State Register has requirements for the entire cluster and all servers that participate in the cluster, over and above the requirements for standalone SBR Carrier servers. A SSR cluster should not have any single point of failure, so each server in a cluster must have its own memory and disks. We do not recommend or support virtual servers, network shares, network file systems, and SANs.

All servers in the cluster require at least two physical Ethernet ports that provide the same throughput. Multi-pathing the NICs to a single IP address is required. Session State Register Cluster can work over a 100Base-T network but we recommend 1000Base-T (gigabit Ethernet).

All data servers must have equal processor power, memory space, and available bandwidth because they are tightly coupled and share data. If the overall throughput of the data servers varies from machine to

machine, performance degrades. SBR Carrier servers and management servers' configuration may vary from machine to machine, so long as the basic standalone requirements are met.

Session State Register Nodes

There are three types of SSR nodes, each with a specific role within the cluster:

- A *SBR Carrier node* is a machine that hosts the RADIUS process. It runs the RADIUS process, any optional modules, and all related processes that read and write data into the SSR database. This type of node accesses and manipulates the cluster's shared data that is hosted by the data nodes.
- A *management node* is a machine that hosts the *SSR management process*. It controls itself and all data nodes in the cluster. It provides configuration data, starts and stops nodes, can back up the database and perform other database operations. It also manages a database process that supports the SSR storage engine. Cluster configuration data is located in an identical **config.ini** file on each of the cluster's management nodes.
- A *data node* is machine that hosts the *SSR data process*. It runs the **ndbd** data process. The **ndbd** process cooperatively manages, replicates, and stores data in the SSR storage engine with other data nodes. Each data node has its own memory and permanent storage. Each one maintains both a portion of the working copy of the SSR database and a portion of one or more replicas of the database.
- A SBRC/SSR management node is a machine that hosts both a RADIUS process and an SSR management process.

Each machine in the front end and the SSR cluster is assigned a *node type* which indicates what processes it hosts: S (SBRC), SM (SBRC and SSR management), M (management), D (data). Thus, the following terms may be used to describe machines that are members of either the front end or the SSR cluster: *S node*, *SM node*, *M node*, and *D node*.

All the data nodes in a cluster run a special process called the *shared memory engine* that manages the working copy of the SSR database. The management nodes coordinate the service among the participating data nodes. The shared storage engine and the SSR database replaces the on-board database used by standalone Steel-Belted Radius Carrier servers. The shared memory engine ensures that the database is updated by a synchronous replication mechanism that keeps cluster nodes synchronized: a transaction is not committed until all cluster nodes are updated.

NOTE: In some cases, the terms *node* and *machine* have been used interchangeably. The term *node* refers to software processes that can be collocated on the same machine.

SSR Data Entities

Each data node participates in a *node group* of two data nodes. A Starter Kit cluster has a single node group with two members; a Starter Kit with an Expansion Kit has two node groups, each with two data nodes. Each node group stores different partitions and replicas.

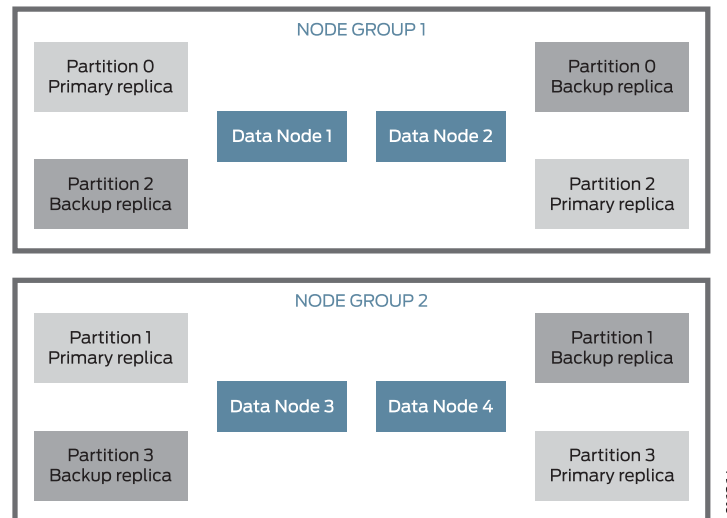
- A *partition* is a portion of all the data stored by the cluster. There are as many cluster partitions as node groups in the cluster. Each node group keeps at least one copy of any partitions assigned to it (that is, at least one replica) available to the cluster.
- A *replica* is a copy of a partition. Each data node in a node group stores a replica of a partition. A replica belongs entirely to a single data node; a node can (and usually does) store several replicas because maintaining two replicas is the fixed setting for SSR.

Figure 243 on page 617 shows the data components of a data cluster with four data nodes arranged in two node groups of two nodes each. Nodes 1 and 2 belong to Node Group 1. Nodes 3 and 4 belong to Node Group 2.

- Because there are four data nodes, there are four partitions.
- The number of replicas is two, to create a two copies of each primary partition.

So long as either both nodes in one node group, or one node in each node group is operating, the cluster remains viable.

Figure 243: SSR with Four Data Nodes in Two Groups



The data stored by the cluster in Figure 243 on page 617 is divided into four partitions: 0, 1, 2, and 3. Multiple copies of each partition are stored within the same node group. Partitions are stored on alternate node groups:

- Partition 0 is stored on Node Group 1. A primary replica is stored on Data Node 1 and a backup replica is stored on Data Node 2.
- Partition 1 is stored on the other node group, Node Group 2. The primary replica is on Data Node 3 and its backup replica is on Data Node 4.
- Partition 2 is stored on Node Group 1. The placement of its two replicas is reversed from that of Partition 0; the primary replica is stored on Data Node 2 and the backup on Data Node 1.
- Partition 3 is stored on Node Group 2, and the placement of its two replicas are reversed from those of partition 1: the primary replica is on Data Node 4 and the backup on Data Node 3.

TIP: *Primary* and *replica* are used in another context in the Steel-Belted Radius Carrier environment and documentation, which can cause some confusion. These terms mean something specific in the context of Session State Register, but they are also used when talking about centralized configuration management, or CCM.

CCM is a feature that coordinates Steel-Belted Radius Carrier server settings between a primary RADIUS server and one or more replica RADIUS servers. It copies critical configuration files from the primary to the replicas, so it keeps multiple SBR Carrier servers operating the same way.

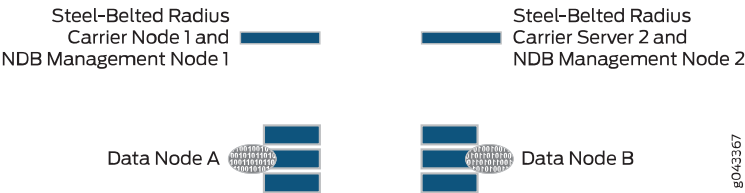
CCM is a separate tool and process that is not tied or linked to SSR, but it is often used in SSR environments to keep the SBR Carrier nodes operating identically.

Cluster Configurations

For the highest level of redundancy, we recommend that each node in a cluster run on its own server. In many locations and for many installations, that might not be practical, so you can run a SBR Carrier and a management node together on the same server—in fact, that is the default configuration for the SSR Starter Kit cluster. However, neither a management node nor an SBR Carrier node can run on the same machine as a data node. Separation is required so that management arbitration services continue if one of the data node servers fails.

Using these separation guidelines, the recommended minimum size of a Session State Register cluster is four physical computers: two servers that each run a SBR Carrier and a management node, and two servers to host the data nodes. This configuration supports all licenses and nodes included in the Session State Register Cluster Starter Kit and is shown in [Figure 244 on page 619](#):

Figure 244: Basic Session State Register Starter Kit Cluster



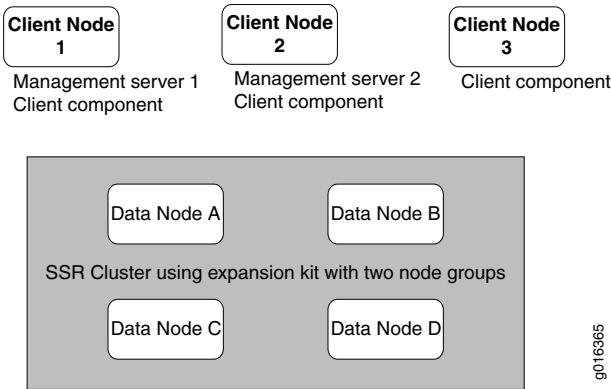
Session State Register Scaling

You scale a Session State Register cluster when you add a separately-licensed SSR Expansion Kit to a Starter Kit, a third management node, or additional SBR Carrier front end systems.

Adding a Data Node Expansion Kit

An Expansion Kit adds two data nodes to increase the number of data nodes in a cluster to four. The additional nodes form a second node group (as shown in [Figure 243 on page 617](#)) that provide more working memory for the SSR shared database. With the Expansion Kit in place, each node group manages a partition of the primary database and replicas. The data in each partition is synchronously replicated between the group's data nodes, so if one data node fails, the remaining node can still access all the data. This configuration also provides very quick failover times if a node fails.

Figure 245: SSR Cluster with an Expansion Kit Setup to Create Two-Node Groups



Adding a Third Management Node

A Management Node Expansion Kit provides software and a license for a third management node. If it is set up on a separate host instead of alongside a SBR Carrier node on a shared server, this also increases the resiliency of the cluster by providing an additional arbiter in case of a node failure.

Adding More SBR Carrier Front End Servers

The service capacity of the SBR Carrier environment grows when you add additional stateless SBR servers to the front end. Adding additional SBR Carrier servers increases the resiliency of the cluster and the speed

of processing a particular transaction because wait time is reduced. Up to 20 Steel-Belted Radius Carrier nodes can be supported by a data cluster.

The SBR Carrier servers do not require identical configurations; they can be configured with different optional modules or communications interfaces. Each one requires a separate SBR Carrier license, but they all share the Session State Register license.

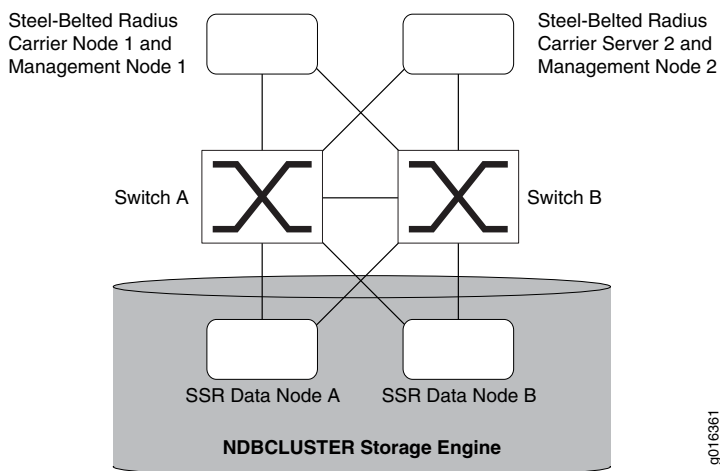
We recommend installing a load balancer in front of the SBR Carrier servers to evenly distribute the RADIUS load between front end SBR Carrier nodes. Regular server-based load balancing works if the front ends only processes RADIUS transactions, Use a RADIUS-aware load balancer if the front ends perform multi-round authentication.

Cluster Network Requirements

A redundant cluster requires a redundant network. At the computer level, we require dual interface cards in each computer and multi-pathing.

We recommend that the network be a dedicated subnet with dual switches. This fully duplicates the network and each computer in the cluster has at least two routes to all other computers, as shown in [Figure 246 on page 620](#).

Figure 246: Starter Kit SSR Cluster with Redundant Network



The SSR database schema uses primary key lookups as often as possible during transaction processing, so the database cluster performance scales almost linearly based on the number of data nodes in the cluster.

Do not configure the subnet to be shared beyond the cluster computers because communications between nodes are not encrypted or shielded in any way. The only means of protecting transmissions within a cluster is to run your cluster on a protected network; do not interpose firewalls between any of the nodes.

Running the cluster on a private or protected network also increases efficiency because the cluster has exclusive use of all bandwidth between cluster hosts. This protects the cluster nodes from interference caused by transmissions between other computers on the network.

Gigabit Ethernet is the strongly recommended network type; 100Base-T is the minimum supported speed. Network latency can severely degrade performance, so we also recommend that all servers be close enough together that latency is always less—much less—than 10 ms.

Table 70: Latency Between Servers and Its Effect on Performance

Latency Times	Performance Degradation
0 ms latency (LAN)	Baseline performance as designed.
10 ms latency	Up to 40% performance loss
20 ms latency	Up to 60% performance loss
More than 20ms latency	Not supported.

Supported SBR Carrier SSR Cluster Configurations

If all add-on products are added to the Starter Kit cluster, the maximum size of a data cluster is four data nodes, three management nodes, and up to 20 SBR Carrier nodes, as shown in [Table 71 on page 621](#).



CAUTION: Setting up an unsupported configuration can put data and equipment at risk and is not supported by Juniper Networks.

Also, note the latency limitation in [Table 70 on page 621](#). We do not support cluster configurations with latency between nodes that exceeds 20 ms, as errors can occur if servers are set up to spread a cluster across widely separated locations.

Table 71: Supported SBR Carrier SSR Cluster Configurations

Cluster Configuration	SBR Carrier Nodes (S)	Management Nodes (M)	Data Nodes (D)
Starter Kit	Two nodes.	Two nodes.	Two nodes, each on a discrete server.
	You may install the four Starter Kit nodes on four discrete servers (a s/s/m/m configuration), or combine one node of each type on two servers (a sm/sm configuration).		Two servers required.

Table 71: Supported SBR Carrier SSR Cluster Configurations (*continued*)

Cluster Configuration	SBR Carrier Nodes (S)	Management Nodes (M)	Data Nodes (D)
Starter Kit and one Data Expansion Kit	You may install the four Starter Kit nodes on four discrete servers (a s/s/m/m configuration), or combine one node of each type on two servers (a sm/sm configuration).		Four nodes, each on a discrete server. Four servers required
Either of the previously listed configurations and a Management Node Expansion Kit	Either Starter Kit configuration (s/s/m/m or sm/sm) may be enhanced with a third management node. We recommend that the third management node be installed on a discrete server.		Either two or four nodes, each on a discrete server.
Any of the previously listed configurations and additional SBR Carrier Nodes (front ends)	One node on a discrete server. Up to 18 additional servers supported, for a total of 20.	Any of the previously listed configurations.	Either two or four nodes, each on a discrete server.

BEST PRACTICE: To minimize the chance and impact of a service interruption, increase the number of data nodes. A Starter Kit can be vulnerable because the loss of one data node and one management node takes the cluster down to the critical 50 percent level. A Starter Kit with a data Expansion Kit (creating two-node groups) is much less likely to lose two or more of the data nodes simultaneously.

Failover Overview

To continue functioning without a service interruption after a component failure, a cluster requires at least 50 percent of its data and management nodes to be functional. If more than 50 percent of the data nodes fail, expect a service interruption, but continued operation of the available nodes.

Because SBR Carrier nodes function as front ends to the data cluster, they are not involved in any failover operations performed by the data cluster. However, as an administrator, you need to ensure that the front end environment is configured, usually through a load balancer, so that it can survive the loss of SBR Carrier nodes.

A data cluster prepares for failover automatically when the cluster starts. During startup, two events occur:

- One of the data nodes (usually the node with the lowest nodeid) becomes the *primary* of the node group. The primary node stores the authoritative copy of the database.

- One data or management node is elected *arbitrator*. The arbitrator is responsible for conducting elections among the survivors to determine roles in case of node failures.

In a cluster, each management and data node is allocated a vote that is used during this startup election and during failover operations. One management node is selected as the initial arbitrator failover problems and of elections that result from them.

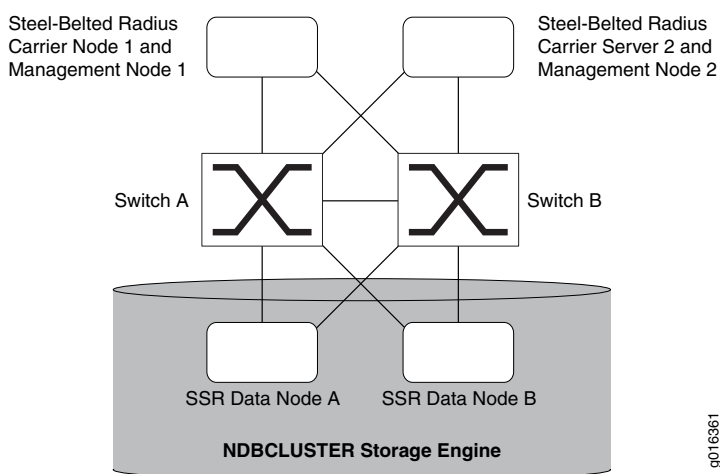
Within the cluster, data and management nodes monitor each other to detect communications loss and heartbeat failure. When either type of failure is detected, as long as nodes with more than 50 percent of the votes are alive, there is instantaneous failover and no service interruption. If exactly 50 percent of nodes and votes are lost, and if a data node is one of the lost nodes, the cluster decides which half of the database is to remain in operation. The half with the arbitrator (which usually includes the primary node) stays up and the other half shuts down to prevent each node or node group from updating information independently.

When a failed data node (or nodes) returns to service, the working nodes resynchronize the current data with the restored nodes so all data nodes are up to date. How quickly this takes place depends on the current load on the cluster, the length of time the nodes were offline, the number of sessions stored, the IO bandwidth on the D node, and other factors.

Failover Examples

The following examples are based on the basic Starter Kit deployment setup with the recommended redundant network as shown in [Figure 247 on page 623](#). The cluster is set up in a single data center on a fully switched, redundant, layer 2 network. Each of the nodes is connected to two switches using the Solaris IP-multipathing feature for interface failover. The switches have a back-to-back connection.

Figure 247: Starter Kit SSR Cluster with Redundant Network



Possible Failure Scenarios

With these basic configurations, a high level of redundancy is supported. So long as one data node is available to one SBR Carrier node, the cluster is viable and functional.

- If either SBR Carrier Server 1 or 2, which also run the cluster's management nodes, (s1 and m1 or s2 and m2) goes down, the effect on the facility and cluster is:
 - No AAA service impact.
 - NAS devices (depending on the failover mechanism in the device) switch to their secondary targets—the remaining SBR Carrier Server. Recovery of the NAS device when the SBR Carrier Server returns to service depends on NAS device implementation.
- If either data node A or B goes down, the effect is:
 - No AAA service impact; both SBR Carrier nodes continue operation using the surviving data node.
 - The management nodes and surviving data node detect that one data node has gone down, but no action is required because failover is automatic.
 - When the data node returns to service, it synchronizes its NDB data with the surviving node and resumes operation.
- If both management nodes (m1 and m2) go down, the effect is:
 - No AAA service impact because the all s and d nodes are still available. The data nodes continue to update themselves.
- If both data nodes go down, the effect on:
 - The management nodes is minimal. They detect that the data nodes are offline, but can only monitor them.
 - The SBR Carrier nodes varies:
 - Authentication of users and accounting for users that do not require shared resources such as the IP address pool or concurrency continues uninterrupted.
 - Users that require shared resources are rejected.

The carrier nodes continue to operate this way until the data cluster comes back online; the cluster resumes normal AAA operation using the data cluster automatically.
- If one half of the cluster (SBR Carrier Server 1, management node 1, and data node A or SBR Carrier Server 2, management node 2, and data node B) go down, the effect is:
 - No AAA service impact because the SBR Carrier node, a management node, and a data node are all still in service. NAS devices using the failed SBR Carrier Server fail over SBR Carrier Server.
 - When the failed data node returns to service, it synchronizes and updates its NDB data with the surviving node and resumes operation.

- When the failed SBR Carrier Server returns to service, the NAS devices assigned to use it as a primary resource return to service depending on the NAS device implementation.

Session State Register Database Tables

The database cluster uses a relational SQL design to store data. The database contains multiple tables that store Steel-Belted Radius Carrier state information such as IP address pool information and current sessions.

IP Address Pools

Session State Register stores information about all IP address pools centrally in a dedicated table in the SSR database. Network Access Servers can send RADIUS requests to any stateless SBR Carrier server, and the SBR Carrier server allocates IP addresses from the shared central pool in response, after authentication.

Operators can query from any SSR management node to verify what addresses are in use and how many addresses remain available for an IP address pool or set of pools.

Subscriber Session Data Controls

Session State Register stores session data centrally in the Current Sessions Table in the SSR database.

The table's format is controlled by a configuration file on the cluster's management nodes; one file is copied to all nodes in a node group, so all nodes operate with the same information. The standard Session State Register database schema addresses the needs of most carriers, but you can modify the CST to address unique needs and situations. The **CurrentSessions.sql** file determines the schema used by the CST table maintained by the data nodes. When the database is created, the table is set up using the schema outlined in the **CurrentSessions.sql** file.

Session control features include:

- User concurrency control—you can enforce concurrency across your entire network by controlling the number of active connections on a per-user, per-cluster basis using the username. You can set a limit on the maximum number of concurrent connections that a user can have and you may use any configured attribute to track instead of just the username.

Subsequently, when the user requests a new connection, the Steel-Belted Radius Carrier servers in the cluster compare the current number of connections to the maximum limit. If a new connection exceeds the limit, the additional connection can be disallowed, or allowed and logged.

- Optional attribute-based concurrency control—If you have a license for the SSR Optional Concurrency and Wholesale Module, you can enforce concurrency across the network by tracking any attribute of a user account, not just the username.

- Change of Authorization (CoA)/Disconnect Message (DM) processing—you can change the state of authenticated sessions dynamically. In accordance with the Dynamic Authentication RFC 3576, you can:
 - Deploy prepaid scenarios in WLAN integration for GSM/UMTS networks.
 - Place users in legal intercept.
 - Enable online service and billing profile changes.
 - Disconnect or quarantine users for abusing the network.

RELATED DOCUMENTATION

[Introduction to Managing and Controlling Sessions in SBR Carrier](#) | 695

[Using the Command Line Utility to Manage and Control Sessions](#) | 730

Application Support

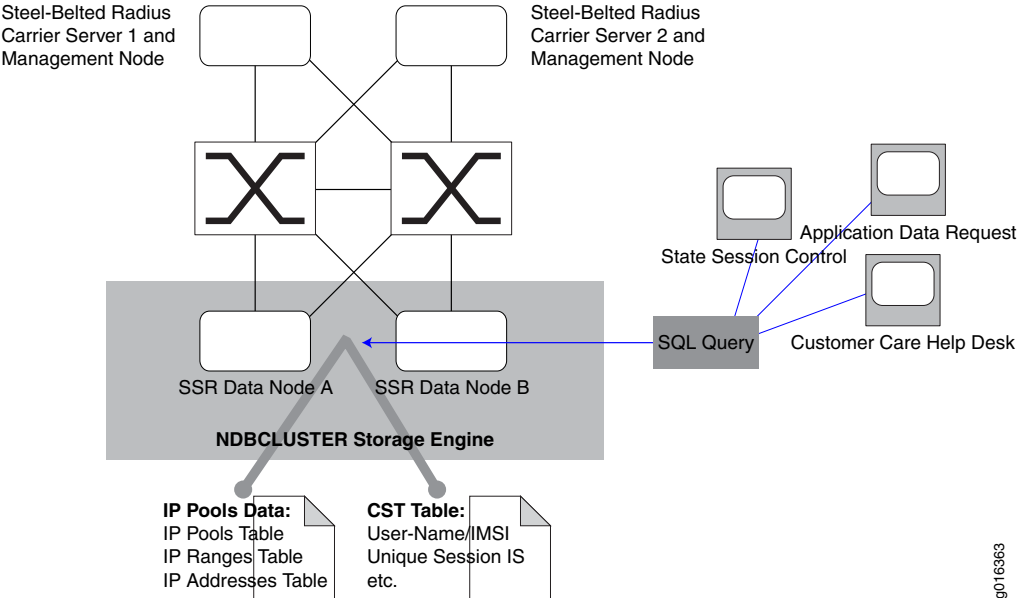
Because the CST is centrally located and can be easily modified, it is often used as the entry point for third party applications that need current session data. This enables Steel-Belted Radius Carrier Session State Register to be the logical source for other applications to query for correlative and verification purposes.

A calling application can make its request through an API as SBR Carrier does, or with a SQL query as shown in [Figure 248 on page 627](#). The SQL query usually takes more processing resources and cycles to complete than an API call, so we recommend against allowing multiple or repeated queries because they can slow down the cluster's primary tasks.

Rows in each table are evenly distributed among a node group's data nodes. A client application can locate a particular record by accessing the unique primary key of any table, which only involves a single node in the cluster. Scans of tables for records meeting broader criteria (such as a search for sessions more than two hours old) involves all the data nodes in the primary node group, and would take more processing cycles to complete. We recommend that queries from outside applications be as precise as possible to avoid slowing all operations down.

[Figure 248 on page 627](#) shows the general deployment scenario with usage examples. Communications initiated by applications or systems in the operational support system (OSS) can be formatted as standard SQL queries, shell scripts, or stored procedures, depending on usage requirements.

Figure 248: Database Access by Other Applications



For example, a video gateway might use its own internal identifier (such as an MSISDN, IMSI, or username) for a subscriber, but might not have the subscriber's current IP address. The video gateway could query the Current Sessions Table in real time to retrieve information about an active session.

Session State Register Administration

IN THIS CHAPTER

- [SSR Administration Overview | 628](#)
- [Overview of Starting and Stopping a Session State Register Cluster | 629](#)
- [Administration Scripts Overview | 637](#)
- [SSR Database Management Scripts | 639](#)
- [Steel-Belted Radius Carrier Node Administration Scripts | 646](#)
- [SSR Session Management | 670](#)
- [Administration Script Control Files | 678](#)

This chapter presents information about maintaining and working the Session State Register (SSR) module. These topics are in this chapter:

SSR Administration Overview

Administering the Steel-Belted Radius Carrier's Session State Register module requires several unique administration tasks that use tools (shell commands and scripts) not available on standalone SBR Carrier servers. Both the tasks and tools tend to break down into these functional areas:

- Cluster and node startup and shutdown
- Steel-Belted Radius Carrier node administration
- Management node administration
- Data node administration

Startup and shut down commands must be executed by root on each node.

Administration scripts usually affect the SSR database or the SBR Carrier node interaction with the database. They are executed by hadm on any of a cluster's management nodes.

Overview of Starting and Stopping a Session State Register Cluster

Having to stop all nodes in a cluster is uncommon because most system maintenance can be done on one system at a time. Taking the whole cluster offline defeats the intention of the cluster—to avoid down time. So ensure that taking all systems down at the same time is required before proceeding. Rather than taking down all nodes, determine whether stopping just the SBR processes or the database management processes might be sufficient.

Stopping a server that hosts both a SBR Carrier and a management node creates a double fault, but does not damage the cluster because a fully redundant cluster always has more than one of each type of node. Stopping multiple nodes that provide redundancy to each other causes multiple faults that may damage the cluster and take the entire cluster off-line.

In the SSR environment, each type of node is started in a specific order so that required resources are available when other nodes require them, and stopped. This means that several commands may be executed on servers that host both SBR Carrier and management nodes.

Startup and shutdown commands must be executed by root on each node.

Starting the Cluster

If all nodes in the cluster are shut down, restarting requires bringing each type of node online in the order as shown in [Table 72 on page 629](#). If the systems are completely shut down, rebooting the computer should restart the appropriate processes automatically because automatic start is the default configuration for all types of Session State Register nodes.

If the systems have not been totally shut down and just the SSR processes have been halted, log in as root and execute the command listed in [Table 72 on page 629](#) to start each type of node's processes. If more than one type of node is running on a single system, maintain the listed order and start all management nodes first, then all data nodes, and finish with the SBR Carrier nodes.

Table 72: Starting Nodes in a Session State Register Cluster

Start Order and Type of Node	Execute This Command
1. SSR Management Node	<code>/opt/JNPRsbr/radius/sbrd start ssr</code>
2. SSR Data Node	<code>/opt/JNPRsbr/radius/sbrd start ssr</code>
3. Steel-Belted Radius Carrier	<code>/opt/JNPRsbr/radius/sbrd start radius</code>

To monitor the cluster coming online:

1. Log in to a management node as `hadm`. (Or if you are root, execute: `su - hadm`.)
2. Run the `show` command.

Execute:

./Monitor.sh "ndb_mgm -e show"

Results similar to this example are displayed:

```
hadm@sbrha-4:~> ./Monitor.sh "ndb_mgm -e show"
```

```
=====[1] Wed Mar 18 17:13:56 (TZ=+00:00) 2009=====
Connected to Management Server at: 172.28.84.36:5235
Cluster Configuration
-----
[ndbd(NDB)]      2 node(s)
id=10   @172.28.84.163  (mysql-5.7.25 ndb-7.6.9, Nodegroup: 0, Master)
id=11   @172.28.84.113  (mysql-5.7.25 ndb-7.6.9, Nodegroup: 0)
```

```
[ndb_mgmd(MGM)] 2 node(s)
id=1    @172.28.84.36  (mysql-5.7.25 ndb-7.6.9)
id=2    @172.28.84.166 (mysql-5.7.25 ndb-7.6.9)
```

```
[mysqld(API)]   4 node(s)
id=21   @172.28.84.36  (mysql-5.7.25 ndb-7.6.9)
id=22   @172.28.84.166 (mysql-5.7.25 ndb-7.6.9)
id=30   @172.28.84.36  (mysql-5.7.25 ndb-7.6.9)
id=31   @172.28.84.166 (mysql-5.7.25 ndb-7.6.9)
```

Stopping the Cluster

If you do need to shut down all nodes in the cluster, make sure that you shut each type of node down in the order shown in [Table 73 on page 631](#). Log in as root on each computer in turn and execute the command to stop the node's processes. If more than one type of node is running on a single computer, maintain the correct order and stop all SBR Carrier nodes, then all management nodes, and finally then all data nodes.



CAUTION: Stopping multiple systems and processes removes all redundancy and can create multiple faults and may damage the cluster. If you do need to stop the cluster, be sure to restart the cluster properly. See [“Starting the Cluster” on page 629](#).

Table 73: Stopping Nodes in a Session State Register Cluster

Stop Order and Type of Node	Execute This Command
1. Steel-Belted Radius Carrier	<code>/opt/JNPRsbr/radius/sbrd stop</code>
2. SSR Management Node	<code>/opt/JNPRsbr/radius/sbrd stop ssr</code>
3. SSR Data Node	<code>/opt/JNPRsbr/radius/sbrd stop ssr</code>

NOTE: To shut down remote SSR Management nodes, execute the `/opt/JNPRsbr/radius/sbrd stop ssr` command on each node, separately.

Stopping a Single Node

You can stop any single node in the cluster to perform maintenance without affecting the integrity of the cluster (because the server’s redundant partner takes on the primary role). These commands just start the node’s process—they have no effect on the system itself, which may still need to be shut down.

If the node is a SBR Carrier node, modifying configuration files often requires a restart.

The stop commands for each type of node are listed in [Table 74 on page 631](#):

Table 74: Stopping a Single Node in a Session State Register Cluster

Type of Node	Execute This Command
Steel-Belted Radius Carrier	<code>/opt/JNPRsbr/radius/sbrd stop radius</code>
SSR Management Node	<code>/opt/JNPRsbr/radius/sbrd stop ssr</code>
SSR Data Node	<code>/opt/JNPRsbr/radius/sbrd stop ssr</code>

Starting a Single Node

To restart a node use the appropriate command from [Table 75 on page 632](#):

Table 75: Starting a Single Node in a Session State Register Cluster

Start Order and Type of Node	Execute This Command
Steel-Belted Radius Carrier	<code>/opt/JNPRsbr/radius/sbrd start radius</code>
SSR Management Node	<code>/opt/JNPRsbr/radius/sbrd start ssr</code>
SSR Data Node	<code>/opt/JNPRsbr/radius/sbrd start ssr</code>

To monitor the startup cycle, use the **show** command. See [“Starting the Cluster” on page 629](#).

sbrd

The **sbrd** script starts and stops different processes on host machines for all three types of Session State Register nodes. The **sbrd** script may be in either of two directories on servers, depending on whether they have been configured to automatically start all procedures or not.

All **sbrd** commands are executed by root. In an SSR environment, the hadm user can execute the script on SSR processes, but expect errors with RADIUS processes that are owned by root.

Running sbrd on Session State Register Nodes

This section applies to running **sbrd** on a nodes in a Session State Register cluster.

Syntax

```
sbrd status [radius|ssr|GWrelay]
sbrd start [radius|ssr|GWrelay] [force]
sbrd start ssr --nowait-nodes=node-ids
sbrd stop [radius|ssr|GWrelay] [force]
sbrd stop [cluster] [force]
sbrd restart [radius|ssr|GWrelay] [force]
sbrd clean [radius|ssr] [force]
sbrd hup [radius|ssr|authGateway [process-name]]
sbrd status [radius|ssr|GWrelay] -v [-p <LCI password>]
```

Options

- The **start**, **stop**, and **restart** arguments start, stop, and restart the process. If a subsystem is not specified, the command works only on RADIUS and GWrelay processes because SSR processes normally are not stopped; to stop them, **ssr** must be invoked. For example: **sbrd stop ssr**.
- Executing **stop cluster** on a SBR Carrier server stops both SSR and RADIUS processes. Executing **stop cluster** on a management node also stops the data nodes controlled by the management node.
- The **clean** argument removes temporary files. When it is executed on a data node, **clean** also prepares the node to take part in a new environment; for example, if an expansion kit is added to increase the number of data nodes from two to four.
- The **status** option displays information such as SBR package version, SBR process status, and loaded plug-in information. For more information about the RADIUS status information, see the *Displaying RADIUS Status Information* section in the *SBR Carrier Installation Guide*.
- The **hup** option operates as the **kill -HUP** command does on SBR Carrier nodes, but does not require the process ID. Executing **sbrd hup authGateway** issues the SIGHUP (1) signal to all the authGateway processes running on SBR Carrier. To issue the SIGHUP (1) signal only to the specific authGateway process, you must execute the hup option with the authGateway process name, for example: **sbrd hup authGateway GMT**.
- The **radius**, **ssr**, or **GWrelay** optional argument specifies which process to operate on when executed on a server that hosts more than one node.
 - **radius** specifies the local Steel-Belted Radius Carrier processes
 - **ssr** specifies data node and management node processes according to the type of node on which it is executed
 - **GWrelay** specifies the GWrelay application.
- Executing **start ssr --nowait-nodes=node-ids** starts the cluster without waiting for the full cluster to be initialized. The *node-ids* variable specifies the comma-separated list of node IDs that are unreachable, for example: **sbrd start ssr --nowait-nodes=51,52**. You must use this argument only if one half of the cluster has network connectivity, but has lost the ability to communicate with the other half. When the network connectivity between the two halves of the cluster is restored, you can start the remaining nodes with the normal startup scripts.
- The **force** argument makes **sbrd** attempt to disregard or overcome any errors that occur when processing the command. Normal behavior without the argument is to halt on errors. For example, **sbrd start** does not attempt to start software that is already running, but **sbrd start force** ignores a running process. This may produce unintended results, so use force with great care.
- The **-v** option displays additional information about the RADIUS process along with basic information such as the SBR package version, SBR process status, and SBR process ID. If you have changed the default Lightweight Directory Access Protocol (LDAP) Configuration Interface (LCI) password, you should use the **-p** option to specify the password. For more information about the RADIUS status information, see *Displaying RADIUS Status Information* section in the *SBR Carrier Installation Guide*.

Examples

This example shows the effect of **sbrd stop ssr** executed on a cluster management node:

```
root@wrx07:~> /etc/init.d/sbrd stop ssr
Stopping ssr auxiliary processes
Stopping ssr management processes
```

```
Connected to Management Server at: 172.28.84.36:5235
Node 2 has shutdown.
Disconnecting to allow Management Server to shutdown
```

This example shows the effect of **sbrd start ssr** on a management node. Be aware that this does not start the data nodes.

```
root@wrx07:~> /etc/init.d/sbrd start ssr
Starting ssr management processes
bash-3.00#
```

NOTE:

- When **sbrd** is executed without a node type argument, it runs against all node processes on the server. For example, **sbrd start** starts both RADIUS and SSR processes for all nodes on a server.
- In an SSR environment, because some servers may host both SBR Carrier and management nodes, **sbrd** may be executed more than once with different arguments.
- Use the **clean** argument only when initializing new data nodes; it removes temporary files and sets file locks to support creation of a new database.

When to Stop, Start, or Restart SBR Carrier Nodes

When modifications are made to SBR Carrier node configuration files, some processes on the node must be restarted to force the newly modified file to be read and used. [Table 76 on page 635](#) lists typical

configuration control files and settings. The **Yes** entry indicates the *least* drastic action that causes the new settings to take effect:

Table 76: When to Stop and Restart SBR Carrier and SSR Processes

Item changes:	Save the window or file	Issue a SIGHUP (1) signal	Stop/restart the server
Access window or object	Yes	(Also works)	(Also works)
access.ini file	No	No	Yes
*.acc files	No	No	Yes
account.ini file	No	No	Yes
admin.ini file	No	No	Yes
*.aut files	No	(Sometimes)	Yes
blacklist.ini file	No	No	Yes
Authentication policy	Yes	(Also works)	(Also works)
*.dct files	No	No	Yes
*.dic	No	No	Yes
*.dhc files	No	No	Yes
dhcp.ini file	No	No	Yes
*.dir files	No	(Sometimes)	Yes
*.eap files	No	(Sometimes)	Yes
eap.ini file	No	No	Yes
enterprises.oid file	No	No	Yes
events.ini file	No	No	Yes
filter.ini file	No	Yes	(Also works)

Table 76: When to Stop and Restart SBR Carrier and SSR Processes *(continued)*

Item changes:	Save the window or file	Issue a SIGHUP (1) signal	Stop/restart the server
*.gen files	No	No	Yes
Import *.rif or users file	Yes	(Also works)	(Also works)
*.ini for directed accounting	No	No	Yes
IP Pools dialog or object	Yes	(Also works)	(Also works)
lockout.ini file	No	No	Yes
Log levels (in radius.ini file)	No	Yes	(Also works)
Profiles dialog or object	Yes	(Also works)	(Also works)
Proxy dialog or object	Yes	(Also works)	(Also works)
*.pro files	No	Yes	(Also works)
proxy.ini file	No	(Sometimes)	Yes
radius.dct file	No	No	Yes
radius.ini file	No	(Sometimes)	Yes
RADIUS Clients dialog or object	Yes	(Also works)	(Also works)
Servers dialog or object	Yes	(Also works)	(Also works)
services file	No	No	Yes
snmpdx.acl file	No	No	Yes
tacplus.ini file	No	No	Yes
Trace levels	No	Yes	(Also works)
Tunnels page or object	Yes	(Also works)	(Also works)
Users dialog or object	Yes	(Also works)	(Also works)

Table 76: When to Stop and Restart SBR Carrier and SSR Processes *(continued)*

Item changes:	Save the window or file	Issue a SIGHUP (1) signal	Stop/restart the server
vendor.ini file	No	No	Yes

Administration Scripts Overview

You use the hadm account to execute a suite of shell scripts to manage the Steel-Belted Radius Carrier Session State Register database. The administration scripts are listed in [Table 77 on page 637](#).

NOTE: Do not attempt to modify or maintain the database except by using these shell scripts, and do not edit the individual configuration files after installation, especially **DBName.txt**, **DBPwd.txt**, and **DBUser.txt**.

Table 77: Steel-Belted Radius Carrier High Availability Scripts

Script Type	Script Name
Database Maintenance	See “CreateDB.sh” on page 641 See “DestroyDB.sh” on page 643
Cache Maintenance	See “ClearCache.sh” on page 649 See “ShowCaches.sh” on page 650
IP Pool Maintenance	See “AddPool.sh” on page 652 See “RenamePool.sh” on page 654 See “DelPool.sh” on page 655 See “ShowPools.sh” on page 656
IP Range Maintenance	See “AddRange.sh” on page 658 See “DelRange.sh” on page 660 See “ShowRanges.sh” on page 661
IP Address Maintenance	See “KillZombieAddrs.sh” on page 662 See “ShowAddrs.sh” on page 664
Session Maintenance	See “ShowSessions.sh” on page 670 See “DelSession.sh” on page 673

Table 77: Steel-Belted Radius Carrier High Availability Scripts (continued)

Script Type	Script Name
User Concurrency	See “ShowUserConc.sh” on page 675 See “DelUserConc.sh” on page 678
Monitoring	See “Monitor.sh” on page 639

- Administration scripts are designed to preserve cross-table referential integrity in the face of administrator errors, such as an attempt to define overlapping ranges or an attempt to delete addresses in the middle of an address range.
- We recommend that only one administrator perform maintenance tasks at a time. The scripts do not lock records, so running multiple scripts that individually modify the database can corrupt the database. (More than one administrator can safely read the databases simultaneously.)
- Do not terminate any running scripts. Interrupting administration scripts that modify the database can leave the database in an invalid state. If you inadvertently run an administration script, allow the script to finish executing and then undo it.
- IP addresses produced as output by an administration script such as **ShowRanges.sh**, or **ShowAddrs.sh** are displayed in a modified “decimal dotted-quad” format. Spaces are added to each quad to improve readability in tables. For example, IP address 192.168.21.3 is presented as 192.168. 21. 3 in script output. Spaces cannot be used as input to scripts that accept IP addresses as input arguments, such as **AddRange.sh**, **DelRange.sh**, **DelSession.sh**, **ShowAddrs.sh**, and **ShowSessions.sh**.
- Administration scripts that display information do not lock the database, because locking can slow down use of the database by RADIUS servers. As a consequence, dynamically changing data, such as cache-related information, is statistically accurate but is not a perfect snapshot of the database.
- Administration scripts (especially **ShowCaches.sh**) identify servers by node ID. This node is specified in the configuration files **config.ini** and **dbclusterndb.gen**. It is also logged in the server log file (**yyyymmdd.log**) as **NDB Node ID = value**.
- Before you execute scripts that modify the SSR database, you must put all SBR Carrier servers that use the database into management mode.
- You can execute multiple *Show* scripts serially by entering each command (and its arguments, if appropriate) on the same command line, separating each command with semicolons. For example, this command executes the **ShowPools.sh**, **ShowRanges.sh**, and **ShowAddrs.sh** scripts:

```
hadm$- ShowPools.sh; ShowRanges.sh; ShowAddrs.sh -a
```

- The scripts use confirmation dialogs to help avoid unintended changes. Answering **no** to a confirmation dialog terminates a script immediately; answering **yes** allows the script to continue. If you are familiar with running scripts, you can bypass all confirmation messages by using the **yes** command, for example:

```
yes yes | CreateDB.sh
```

SSR Database Management Scripts

All SSR administration scripts are executed on management nodes. The scripts directly related to cluster and database management are included in this section.

Using the Monitor Script

The **Monitor.sh** script is useful for monitoring cluster operations, either to take a snapshot or to use for real-time monitoring.

Monitor.sh

The **Monitor.sh** script provides real-time monitoring of the SSR cluster. The **Monitor.sh** script periodically executes one or more scripts, such as **ShowPools.sh**, which looks for changes in the Steel-Belted Radius Carrier high availability database. If a change is found, the **Monitor.sh** script updates the statistics it displays when a change is detected.

NOTE: To change the default time zone setting from UTC time, you must edit this script and the **DBtime zone.txt** field. For more information about editing the scripts, see [Table 102 on page 679](#).

Syntax

```
Monitor.sh [-s secs-between-commands (def=1) ] [-l lim (def=0) ] commands "scriptname [arguments]
[scriptname [arguments] [...]]"
Monitor.sh -c
Monitor.sh -h
```

Options

Table 78: Monitor.sh Options

Option	Description
-seconds	Specifies the number of seconds the Monitor.sh script should wait between execution of its commands. The number of seconds must follow the -s argument without a space. Default value is 1 second.
-l	Specifies only a limited number of printouts; 0 defaults to no limit.
scriptname	Specifies the name of the script you want Monitor.sh to execute.
arguments	Specifies the arguments for the script that you want Monitor.sh to execute.
-c	Clears the screen before each display page (by default, a blank line separates the displays).
-h	Displays help for the Monitor.sh script.

Example

The following example shows the output for the **Monitor.sh** script:

```
hadm$- Monitor.sh "Ndb_mgm -e show"

===== [1] Mon Apr 23 15:38:24 (TZ=+00:00) 2009 =====
Connected to Management Server at: 172.25.98.250:1186
Cluster Configuration
-----
[ndbd(NDB)]      6 node(s)
id=10   @172.25.98.245   (Version: 5.0.30, Nodegroup: 0, Master)
id=11   @172.25.98.246   (Version: 5.0.30, Nodegroup: 0)
id=12   @172.25.98.247   (Version: 5.0.30, Nodegroup: 1)
id=13   @172.25.98.248   (Version: 5.0.30, Nodegroup: 1)
id=14   @172.25.98.244   (Version: 5.0.30, Nodegroup: 2)
id=15   @172.25.98.251   (Version: 5.0.30, Nodegroup: 2)

[ndb_mgmd(MGM)]  2 node(s)
id=1     @172.25.98.249   (Version: 5.0.30)
id=2     @172.25.98.250   (Version: 5.0.30)
```

```
[mysqld(API)] 8 node(s)
id=20 @172.25.98.249 (Version: 5.0.30)
id=21 @172.25.98.250 (Version: 5.0.30)
id=30 @172.25.98.243 (Version: 5.0.30)
id=32 (not connected, accepting connect from 172.25.98.250)
id=34 (not connected, accepting connect from 172.25.98.252)
id=35 @172.25.98.231 (Version: 5.0.30)
id=36 @172.25.98.232 (Version: 5.0.30)
id=63 (not connected, accepting connect from any host)
```

Creating and Destroying the SSR Database

One script creates the SSR database; a second destroys the existing database.

CreateDB.sh

The **CreateDB.sh** script creates a new database from the schema in the **CurrentSessions.sql** file. For details, see the section on *Customizing the SSR Database Current Sessions Table* in the *SBR Carrier Installation Guide*. It rebuilds all related tables and stored procedures, and gives the new database the name specified in the **DBName.txt** file.

CreateDB.sh must be executed on all management nodes in the cluster.

Syntax

```
CreateDB.sh
CreateDB.sh -h
```

Options

Table 79: CreateDB.sh Options

Option	Description
-h	Displays help for the CreateDB.sh script.

Example

The following example displays the contents of the **DBName.txt** file (to verify the name (“**SteelBeltedRadius**”) to be used to create the database) and creates a database named **SteelBeltedRadius**.

```
hadm$ cat DBName.txt
SteelBeltedRadius
hadm$ CreateDB.sh
Creating database “SteelBeltedRadius” (using ENGINE ndbcluster).
```

Creating misc tables.
 Creating IP Pool, Range and Address tables.
 Creating Current Sessions table.
 (Proxy AutoStop feature not configured.)
 (Session Timeout on Missed Account Stop feature not configured.)
 (Use Single Class Attribute feature not configured.)
 Creating User Concurrency table.
 Creating stored routines.

Usage Notes

The **CreateDB.sh** script determines whether a database exists when it starts; if a database is found, the **CreateDB.sh** script halts.

Auxiliary SQL Files Used by CreateDB.sh

Table 80 on page 642 lists the auxiliary SQL files in **/opt/JNPRhadm** that contain information used by the **CreateDB.sh** script.

Table 80: Auxiliary SQL Files Used by CreateDB.sh

SQL Filename	Description	Table Schemas
Misc.sql	Contains schemas for miscellaneous internal tables	<ul style="list-style-type: none"> • Sbr_Abort—Used for terminating transactions/scripts • Sbr_LastPoolOrd—Used for numbering IP pools
IPAddrs.sql	Contains schemas for tables involved in IP address management	<ul style="list-style-type: none"> • Sbr_IpPools—Contains the names of the IP pools and their ordinal numbers • Sbr_IpRanges—Contains the ranges of the IP addresses and the pools that they are in • Sbr_IpAddrs—Contains all configured IP addresses and the details
CurrentSessions.sql	Contains schema for the Current Sessions Table	<ul style="list-style-type: none"> • Sbr_CurrentSessions—Contains information about current (ongoing) sessions
UserConcurrency.sql	Contains schema for the User Concurrency Table	<ul style="list-style-type: none"> • Sbr_UserConcurrency—Contains identifiers and counts
StoredRoutines.sql	Contains various stored routines used by the administration scripts	

DestroyDB.sh

The **DestroyDB.sh** script deletes the existing database.

DestroyDB.sh needs to be run once, on one management node.

Syntax

```
DestroyDB.sh
DestroyDB.sh -h
```

Options

Table 81: DestroyDB.sh Options

Option	Description
-h	Displays help for the DestroyDB.sh script.

Example

```
hadm$- DestroyDB.sh
SBRs must be offline;OK <yes|no>? yes
This will destroy the "SteelBeltedRadius" database; OK? <yes|no>? yes
Really? <yes|no>? yes
Database "SteelBeltedRadius" destroyed
```

Creating a Demonstration Database

One script creates a demonstration database that is useful for testing, but is not intended to create a working production database.

DemoSetup.sh

The **DemoSetup.sh** script creates a new database and related tables, gives the new database the name specified in the **DBName.txt** file, and populates the database with a sample IP address pool configuration. You can use this program to test a cluster environment without using live production data.

Syntax

```
DemoSetup.sh [ numpools maxranges minaddrs maxaddrs ]
DemoSetup.sh -h
```

Options

Table 82: DemoSetup.sh Options

Option	Description
numpools	A number in the range 1–25 that specifies the number of IP address pools for the script to create. Default value is 5.
maxranges	A number in the range 1–20 that specifies the number of address ranges within each IP address pool for the script to create. Default value is 4.
minaddrs	A number in the range 1–100,000 that specifies the minimum number of IP addresses per range for the script to create. Default value is 1000.
maxaddrs	A number in the range 1–100,000 that specifies the maximum number of IP addresses per range for the script to create. Default value is the value of minaddrs x 2.
-h	Displays help for the DemoSetup script.

Example

This example displays the contents of the **DBName.txt** file (to verify that **SteelBeltedRadius** is used to create the database) and creates a database called **SteelBeltedRadius**:

```
hadm$- DemoSetup.sh
SBRs must be offline, do you want to proceed? <yes|no> yes
This will destroy the "SteelBeltedRadius" database (if it exists), OK? <yes|no>
yes
NUMPOOLS=5 MAXRANGES=4 MINADDRS=1000 MAXADDRS=2000
Destroying old database
Database "SteelBeltedRadius" destroyed.
Creating new database
Creating database "SteelBeltedRadius" (using ENGINE ndbcluster).
Creating misc tables.
Creating IP Pool, Range and Address tables.
Creating Current Sessions table.
(Proxy AutoStop feature not configured.)
(Session Timeout on Missed Account Stop feature not configured.)
```

```

    (Use Single Class Attribute feature not configured.)
    LDAP/SQL Bridge feature configured (though not yet enabled).
    Creating User Concurrency table.
    Creating stored routines.
    Adding pool: A-PLATINUM
    Adding range: A-PLATINUM 238.203.131.14 670
    Adding range: A-PLATINUM 48.226.119.162 761
    Adding pool: B-GOLD
    Adding range: B-GOLD 90.169.221.242 549
    Adding range: B-GOLD 20.97.26.189 89
    Adding range: B-GOLD 114.23.180.47 407
    Adding range: B-GOLD 236.99.7.33 384
    Adding pool: C-SILVER
    Adding range: C-SILVER 201.213.13.198 978
    Adding range: C-SILVER 15.28.0.17 22
    Adding range: C-SILVER 28.189.195.246 2
    Adding range: C-SILVER 132.185.74.85 11
    Adding pool: D-BRONZE
    Adding range: D-BRONZE 122.219.182.131 1247
    Adding range: D-BRONZE 247.168.228.227 20
    Adding pool: E-ZINC
    Adding range: E-ZINC 135.54.24.139 1794

```

IpPools:

Name	Ord	Count
(z o m b i e)	0	0
A-PLATINUM	1	1,431
B-GOLD	2	1,429
C-SILVER	3	1,013
D-BRONZE	4	1,267
E-ZINC	5	1,794

Total Pools : 5 + 1 zombie pseudo-pool.

Total Ranges: 13 + 0 zombie pseudo-ranges.

Total Addrs : 6,934 + 0 zombie pseudo-addrs.

IpRanges:

Pool	StartAddr	EndAddr	Count
(z o m b i e)	(v a r i o u s)	(v a r i o u s)	0
A-PLATINUM	48.226.119.162	48.226.122.154	761
A-PLATINUM	238.203.131. 14	238.203.133.171	670
B-GOLD	20. 97. 26.189	20. 97. 27. 21	89

B-GOLD	90.169.221.242	90.169.224. 22	549	
B-GOLD	114. 23.180. 47	114. 23.181.197	407	
B-GOLD	236. 99. 7. 33	236. 99. 8.160	384	
C-SILVER	15. 28. 0. 17	15. 28. 0. 38	22	
C-SILVER	28.189.195.246	28.189.195.247	2	
C-SILVER	132.185. 74. 85	132.185. 74. 95	11	
C-SILVER	201.213. 13.198	201.213. 17.151	978	
D-BRONZE	122.219.182.131	122.219.187. 97	1,247	
D-BRONZE	247.168.228.227	247.168.228.246	20	
E-ZINC	135. 54. 24.139	135. 54. 31.140	1,794	
+-----+-----+-----+-----+				

Steel-Belted Radius Carrier Node Administration Scripts

Several types of scripts effect different areas of SBR Carrier operation.

Using IP Address and IP Address Pool Scripts

To set up or modify the IP Address Pool, you use specific administration scripts that modify the IP Address Pool table in the SSR database. You must place all Steel-Belted Radius Carrier nodes into management mode before you work on the IP address tables. When a SBR Carrier node is in management mode, caching is disabled, and that slows down performance, so we recommend that you schedule IP pool maintenance for off-peak hours.

You can modify configuration parameters (other than the **ManagementMode** parameter in **dbclusterndb.gen**) without putting the servers into management mode. When you change other settings on the Steel-Belted Radius Carrier node, the server enters and exits management mode silently when appropriate.

You must place all of the cluster's SBR Carrier nodes in management mode to execute these scripts because they write IP address information into the database:

- See ["AddPool.sh" on page 652](#)
- See ["AddRange.sh" on page 658](#)
- See ["ClearCache.sh" on page 649](#)
- See ["DelPool.sh" on page 655](#)
- See ["DelRange.sh" on page 660](#)
- See ["KillZombieAddrs.sh" on page 662](#)

- See [“RenamePool.sh” on page 654](#)
- See [“RestoreIP.sh” on page 668](#)

You can execute these IP-related scripts without entering management mode because they do not write information to the database:

- See [“Monitor.sh” on page 639](#)
- See [“ShowAddrs.sh” on page 664](#)
- See [“ShowCaches.sh” on page 650](#)
- See [“ShowPools.sh” on page 656](#)
- See [“ShowRanges.sh” on page 661](#)

Using Management Mode

You must place all SBR Carrier servers that are the front end for a SSR cluster into management mode before you modify any part of the SSR IP address tables.

Placing a SBR Carrier Server into Management Mode

To place a SBR Carrier server into management mode, as root:

1. Edit `/opt/JNPRsbr/radius/dbclusterndb.gen`.

Scroll down the file past the opening comments and the management mode comments to the **ManagementMode** entry:

```
... (comments)
;-----
; ManagementMode=<0|1> (0=false, 1=true); default=0          {updatable}
... (comments)
```

```
;-----
ManagementMode = 0
```

2. Set the value of the **ManagementMode** parameter to **1**.

Save the file.

3. Restart the process.

Either:

- Execute a HUP command.

- Execute: `/opt/JNPRsbr/radius/sbrd hup`.

4. Scan or tail the server log to confirm that it contains this entry:

DB Cluster plugin has entered Management Mode

5. Repeat this procedure on each SBR Carrier node in the cluster before executing the IP address database command scripts.

Removing a SBR Carrier Server from Management Mode

Remove all SBR Carrier servers associated with an SSR cluster from management mode after you reconfigure your SSR IP address tables. To remove a SBR Carrier server from management mode:

To place a SBR Carrier server into management mode, as root:

1. Edit `/opt/JNPRsbr/radius/dbclusterndb.gen`.

Scroll down the file past the opening comments and the management mode comments to the **ManagementMode** entry:

```
... (comments)
;-----
; ManagementMode=<0|1> (0=false, 1=true); default=0 {updatable}
... (comments)
```

```
;-----
ManagementMode = 1
```

2. Set the value of the **ManagementMode** parameter to **0**.

Save the file.

3. Restart the process.

Either:

- Execute a HUP command.
- Execute: `/opt/JNPRsbr/radius/sbrd hup`.

4. Scan or tail the server log to confirm that it contains this entry:

DB Cluster plugin has left Management Mode

5. Repeat this procedure on each SBR Carrier node in the cluster.

ClearCache.sh

The **ClearCache.sh** script lets you mark the cached addresses associated with a specified SSR node (or all cached addresses) as uncached.

Syntax

All SBR Carrier nodes must be in management mode before you execute this script.

```
ClearCache.sh -a
ClearCache.sh nodeid
ClearCache.sh -h
```

Options

Table 83: ClearCache.sh Options

Option	Description
-a	Specifies that all cached addresses should be marked uncached.
nodeid	Specifies that all cached addresses associated with the specified SSR node should be marked as uncached. The value you enter for nodeid is the SSR node identifier specified in the config.ini file on the database management server.
-h	Displays help for the ClearCache.sh script.

Example

The following example (1) uses the **ShowAddrs.sh** script to list the address cache list for Node 24, which contains four IP addresses; (2) uses the **ClearCache.sh** script to clear the address cache for Node 24; and (3) uses the **ShowAddrs.sh** script to show that no IP addresses remain in the address cache list for Node 24.

```
hadm$- ShowAddrs.sh
```

```
IpAddrs:
```

```
+-----+-----+-----+-----+-----+
| IpAddress | Pool | Cache | InUse | LastUse |
+-----+-----+-----+-----+-----+
| 0. 0. 0. 10 | B-GOLD | 0 | 0 | 2005-11-02 14:08:36 |
| 0. 0. 0. 11 | B-GOLD | 24 | 0 | 0000-00-00 00:00:00 |
| 0. 0. 0. 12 | B-GOLD | 0 | 0 | 2005-11-02 14:08:47 |
| 0. 0. 0. 13 | B-GOLD | 24 | 0 | 0000-00-00 00:00:00 |
| 0. 0. 0. 14 | B-GOLD | 24 | 0 | 0000-00-00 00:00:00 |
| 0. 0. 0. 15 | B-GOLD | 24 | 0 | 0000-00-00 00:00:00 |
```

```
+-----+-----+-----+-----+-----+
hadm$- ClearCache.sh 24
hadm$- ShowAddrs.sh
IpAddrs:
+-----+-----+-----+-----+-----+
| IpAddress      | Pool      | Cache | InUse | LastUse          |
+-----+-----+-----+-----+-----+
| 0. 0. 0. 10    | B-GOLD    | 0      | 0      | 2005-11-02 14:08:36 |
| 0. 0. 0. 11    | B-GOLD    | 0      | 0      | 0000-00-00 00:00:00 |
| 0. 0. 0. 12    | B-GOLD    | 0      | 0      | 2005-11-02 14:08:47 |
| 0. 0. 0. 13    | B-GOLD    | 0      | 0      | 0000-00-00 00:00:00 |
| 0. 0. 0. 14    | B-GOLD    | 0      | 0      | 0000-00-00 00:00:00 |
| 0. 0. 0. 15    | B-GOLD    | 0      | 0      | 0000-00-00 00:00:00 |
+-----+-----+-----+-----+-----+
```

Usage Notes

The **ClearCache.sh** script is intended for use in situations where a SBR Carrier server has failed and you want to recover the addresses associated with the node. If you execute the **ClearCache.sh** script against an operational node, SBR Carrier cannot track which addresses from that node have been issued, potentially resulting in addresses being assigned to more than one user.

ShowCaches.sh

The **ShowCaches.sh** script enables you to display detailed information about IP address pool caches.

Syntax

```
ShowCaches.sh
ShowCaches.sh -h
```

Options

Table 84: ShowCaches.sh Options

Option	Description
-h	Displays help for the ShowCaches.sh script.

Example

The following example shows the combined output of the **ShowCaches.sh** script and the **ShowPools.sh** script using the **-r** option.

```
hadm$- ShowPools.sh; ShowRanges.sh; ShowPools.sh -v; ShowPools.sh -vr;
ShowCaches.sh
```


IpPools:

Name	Ord	Count
(z o m b i e)	0	0
A-PLATINUM	1	1,775
B-GOLD	2	1,429
C-SILVER	3	1,540
D-BRONZE	4	1,375
E-ZINC	5	1,966

Total Pools : 5 + 1 zombie pseudo-pool.

Total Ranges: 10 + 0 zombie pseudo-ranges.

Total Addrs : 8,085 + 0 zombie pseudo-addrs.

IpRanges:

Pool	StartAddr	EndAddr	Count
(z o m b i e)	(v a r i o u s)	(v a r i o u s)	0
A-PLATINUM	51. 11.158.213	51.11.160.187	487
A-PLATINUM	114. 90.189. 68	114.90.190.234	423
A-PLATINUM	118.218. 81. 77	118.218.84.173	865
B-GOLD	34.201.180.235	34.201.186.127	1,429
C-SILVER	60.103.156.174	60.103.162. 43	1,406
C-SILVER	72.138.136.244	72.138.137. 82	95
C-SILVER	73. 45. 8. 88	73.45.8.126	39
D-BRONZE	212.182.252.126	212.183.1.141	1,296
D-BRONZE	255. 65.167.181	255.65.168.3	79
E-ZINC	218. 88. 62.250	218.88.70.167	1,966

IpPools:

Name	Ord	Count	%Free	%Cache	%InUse
(z o m b i e)	0	0	----	----	----
A-PLATINUM	1	1,775	66.2	33.8	0.0
B-GOLD	2	1,429	58.0	42.0	0.0
C-SILVER	3	1,540	61.0	39.0	0.0
D-BRONZE	4	1,375	56.4	43.6	0.0
E-ZINC	5	1,966	69.5	30.5	0.0

IpPools:

Name	Ord	Count	%Free	%Cache	%InUse
------	-----	-------	-------	--------	--------

```

+-----+-----+-----+-----+-----+-----+
| (z o m b i e) | 0 | 0 | 0 | 0 | 0 |
| A-PLATINUM | 1 | 1,775 | 1,175 | 600 | 0 |
| B-GOLD | 2 | 1,429 | 829 | 600 | 0 |
| C-SILVER | 3 | 1,540 | 940 | 600 | 0 |
| D-BRONZE | 4 | 1,375 | 775 | 600 | 0 |
| E-ZINC | 5 | 1,966 | 1,366 | 600 | 0 |
+-----+-----+-----+-----+-----+-----+
IpCaches:
+-----+-----+-----+-----+
| Pool | Node | Count | %Cache |
+-----+-----+-----+-----+
| A-PLATINUM | 30 | 250 | 14.08 |
| A-PLATINUM | 31 | 150 | 8.45 |
| A-PLATINUM | 32 | 200 | 11.27 |
| B-GOLD | 30 | 250 | 17.49 |
| B-GOLD | 31 | 150 | 10.50 |
| B-GOLD | 32 | 200 | 14.00 |
| C-SILVER | 30 | 250 | 16.23 |
| C-SILVER | 32 | 150 | 9.74 |
| C-SILVER | 32 | 200 | 12.99 |
| D-BRONZE | 30 | 250 | 18.18 |
| D-BRONZE | 31 | 150 | 10.91 |
| D-BRONZE | 32 | 200 | 14.55 |
| E-ZINC | 30 | 250 | 12.72 |
| E-ZINC | 31 | 150 | 7.63 |
| E-ZINC | 32 | 200 | 10.17 |
+-----+-----+-----+-----+
hadm$-

```

This output shows five pools and ten ranges. There are three SBR Carrier servers attached to the cluster (node IDs 30, 31, and 32). It also shows the state right after all three SBR Carrier servers have been booted, with each node caching its “high-water mark” of addresses, and no addresses yet “in use”.

Node 30 has a high-water mark of 250 for each pool, Node 31 has 150, and Node 32 has 200. The “%Cache” column in the **ShowPools.sh** script is cumulative over SBR Carrier servers, while “%Cache” in the **ShowCaches.sh** script is per-SBR Carrier server (finer granularity); the latter adds up to the former, for each pool.

AddPool.sh

The **AddPool.sh** script adds a named IP address pool to the IpPools database table.

Syntax

All SBR Carrier nodes must be in management mode.

```
AddPool.sh poolname
AddPool.sh -h
```

Options

Table 85: AddPool.sh Options

Option	Description
<i>poolname</i>	Address pool names can be entered in lower, mixed, or upper case, but names are converted to all capitals before the pool is created. User-created address pool names cannot include spaces or tab characters.
-h	Displays help for the AddPool.sh script.

Example

To add an IP address pool called **GOLD** to the database and then show it, enter the following commands:

```
hadm$-- AddPool.sh GOLD
SBRs must be offline or in ManagementMode; OK? <yes|no> yes

hadm$-- ShowPools.sh
IpPools:
+-----+-----+-----+
|Name      |Ord   |Count |
+-----+-----+-----+
|(zombie)  |0     |0     |
| GOLD     |1     |0     |
+-----+-----+-----+
Total Pools: 1 + 1 zombie pseudo-pool.
Total Ranges: 0 + 0 zombie pseudo-ranges.
Total Addrs: 0 + 0 zombie pseudo-addrs.
```

Usage Notes

- When an IP address pool is created, the script assigns the pool a permanent identification number (“ordinal”) in the range 1–65535. Ordinals are assigned in “wrap-around” fashion, where ordinals of increasing values are used before ordinals of lower value are re-used. When ordinals wrap around, values that are still in use are skipped.
- Ordinal number 0 is reserved for the zombie address pool.

RenamePool.sh

The **RenamePool.sh** script enables you to change the name assigned to an IP address pool in the **IpPools** database table. The **RenamePool.sh** script does not change the status of IP address ranges associated with a pool or the status of IP addresses in the pool. You cannot rename a zombie pool.

Syntax

All SBR Carrier nodes must be in management mode.

```
RenamePool.sh oldname newname
RenamePool.sh -h
```

Options

Table 86: RenamePool.sh Options

Option	Description
<i>oldname</i>	Current name of the IP address pool that you want to rename.
<i>newname</i>	New name you want to assign to the IP address pool. Address pool names can be entered in lower, mixed, or upper case, but names are converted to all capitals before the pool is created. User-created address pool names cannot include spaces or tab characters.
-h	Displays help for the RenamePool.sh script.

Example

The following example lists the address pools configured for Steel-Belted Radius Carrier high availability, changes the name of the GLOD pool to GOLD, and confirms the correction was accepted.

```
hadm$-- ShowPools.sh
IpPools:
+-----+-----+-----+
|Name|Ord|Count|
+-----+-----+-----+
| (zombie) | 0 | 0 |
| GLOD | 2 | 0 |
+-----+-----+-----+
Total Pools: 1 + 1 zombie pseudo-pool.
Total Ranges: 0 + 0 zombie pseudo-ranges.
Total Addrs: 0 + 0 zombie pseudo-addrs.
```

```
hadm$-- RenamePool.sh GLOD GOLD
hadm$-- ShowPools.sh
IpPools:
+-----+-----+
|Name      |Ord      |Count|
+-----+-----+
| (zombie) |0        |0    |
| GOLD     |1        |0    |
+-----+-----+
Total Pools: 1 + 1 zombie pseudo-pool.
Total Ranges: 0 + 0 zombie pseudo-ranges.
Total Addrs: 0 + 0 zombie pseudo-addrs.
```

DelPool.sh

The **DelPool.sh** script enables you to delete an IP address pool in the **IpPools** database table. When you use the **DelPool.sh** script to delete an address pool, Steel-Belted Radius Carrier high availability moves the IP addresses in that pool to the zombie pool to preserve their usage statistics.

Syntax

All SBR Carrier nodes must be in management mode.

```
DelPool.sh poolname
DelPool.sh -a
DelPool.sh -z
DelPool.sh -h
```

Options

Table 87: DelPools.sh Options

Option	Description
<i>poolname</i>	Name of the IP address pool that you want to delete.
-a	Deletes all IP address pools in the IpPools database and corresponding ranges from the IpRanges table.
-z	Indicates whether zombie addresses are to be created.
-h	Displays help for the DelPool.sh script.

Example

The following example lists the address pools configured for Steel-Belted Radius Carrier high availability, deletes the C-SILVER address pool, and confirms the correction was accepted.

```

hadm$- DelPool.sh C-SILVER
hadm$- ShowPools.sh
IpPools:
+-----+-----+-----+-----+-----+-----+
|           Name |   Ord |         Count |   %Free |   %Cache |   %InUse |
+-----+-----+-----+-----+-----+-----+
| (z o m b i e) |     0 |           0 |   ---- |   ---- |   ---- |
|           B-GOLD |     1 |          15 |   46.6 |   46.6 |    6.6 |
+-----+-----+-----+-----+-----+-----+

```

Usage Notes

You cannot delete the zombie pool.

ShowPools.sh

The **ShowPools.sh** script displays the name and ordinal number of a specified IP address pool or all IP address pools, along with statistics for how many addresses are configured in the pool and how many addresses are currently in an available, cached, or assigned state.

Syntax

```

ShowPools.sh or ShowPools.sh -r
ShowPools.sh -v
ShowPools.sh -p Poolname or ShowPools.sh -r -p Poolname
ShowPools.sh -v -p Poolname
ShowPools.sh -v-r -p Poolname
ShowPools.sh -h

```

Options

Table 88: ShowPools.sh Options

Option	Description
-v	Displays output in percentages for the ShowPools.sh command in verbose mode. The percentages always add to 100% per pool.
-r	Displays output in raw numbers instead of percentages. When invoked with the -v option, you can use any format: -vr , -rv , -v -r , or -r -v . If the -r option is used without the -v option, the command is ignored.

Table 88: ShowPools.sh Options (continued)

Option	Description
-p <i>Poolname</i>	<p>Displays output of the specified IP address pool in raw numbers, when invoked with the -r, -vr, -rv, -v -r, or -r -v option. The output also contains the statistics for the total number of IP pools, pool ranges, and IP addresses in the database.</p> <p>When invoked with the -v option, this script displays the output of the specified IP address pool in percentages.</p> <p>NOTE: The exact pool name should be used for <i>Poolname</i>. Wildcard characters (such as *) are not allowed in the script.</p>
-h	Displays help for the ShowPools.sh script.

Example

The following example lists the address pools configured for Steel-Belted Radius Carrier high availability.

```

hadm$- ShowPools.sh -v
IpPools:
+-----+-----+-----+-----+-----+-----+
|Name          |Ord   |Count  |%Free  |%Cache  |%InUse  |
+-----+-----+-----+-----+-----+-----+
|(z o m b i e) |0     |0       |----   |----    |----    |
|POOL1         |1     |10      |0.0    |100.0   |0.0     |
|POOL2         |2     |20      |0.0    |100.0   |0.0     |
|POOL3         |3     |255     |0.0    |100.0   |0.0     |
+-----+-----+-----+-----+-----+-----+

hadm$- ShowPools.sh -r
IpPools:
+-----+-----+-----+
|Name          |Ord   |Count  |
+-----+-----+-----+
|(z o m b i e) |0     |0       |
|POOL1         |1     |10      |
|POOL2         |2     |20      |
|POOL3         |3     |255     |
+-----+-----+-----+

Total Pools :    3 + 1 zombie pseudo-pool.
Total Ranges:    3 + 0 zombie pseudo-ranges.
Total Addrs : 285 + 0 zombie pseudo-addrs.

hadm$- ShowPools.sh -v -p POOL1

```

```

IpPools:
+-----+-----+-----+-----+-----+-----+
|Name          |Ord   |Count  |%Free  |%Cache  |%InUse  |
+-----+-----+-----+-----+-----+-----+
|POOL1         |1     |10     |0.0    |100.0   |0.0     |
+-----+-----+-----+-----+-----+-----+

hadm$- ShowPools.sh -r -p POOL1
IpPools:
+-----+-----+-----+
|Name          |Ord   |Count  |
+-----+-----+-----+
|POOL1         |1     |10     |
+-----+-----+-----+

Total Pools :    3 + 1 zombie pseudo-pool.
Total Ranges:    3 + 0 zombie pseudo-ranges.
Total Addrs : 285 + 0 zombie pseudo-addrs.

```

Usage Notes

- The output of the **ShowPools.sh** script is sorted alphabetically by pool name.
- The **Count** field displays the total number of IP addresses in the pool. If an address pool consists of more than one address range, the **Count** field displays the sum of the number of addresses in each range. When the **Count** field is zero, the percentage fields are represented with dashes because the quotient $\text{InUse}/\text{Count}$ does not exist.
- The **Free**, **Cache**, and **InUse** fields, which are populated when the **-v** (verbose) option is used, display the percentage of addresses in an address pool that are in each state.
 - The **Free** field displays the percentage of addresses that are available for caching or assignment.
 - The **Cache** field displays the percentage of addresses that have been cached by a Steel-Belted Radius Carrier node.
 - The **InUse** field displays the percentage of addresses in the pool that are currently assigned to users.

AddRange.sh

The **AddRange.sh** script adds an IP address range to the **IpRanges** database table. The new range cannot intersect with an existing range. If no zombie addresses exist in the new range, the **AddRange.sh** script will run quickly; otherwise it will run slowly.

Syntax

All SBR Carrier nodes must be in management mode.


```
AddRange.sh poolname startaddr { count | endaddr }
AddRange.sh -u
AddRange.sh -h
```

Options

Table 89: AddRange.sh Options

Option	Description
<i>poolname</i>	Specifies the name of the IP address pool to which you want to add an IP address range. Address pool names can be entered in lower, mixed, or upper case, but names are converted to all capitals before the script continues.
<i>startaddr</i>	Specifies the first IP address in the IP address range. Enter IP addresses in dotted decimal format.
<i>count</i>	Specifies the number of addresses in the address range. You must enter a <i>count</i> argument if you do not enter an <i>endaddr</i> argument.
<i>endaddr</i>	Specifies the last IP address in the IP address range. Enter IP addresses in dotted decimal format. You must enter an <i>endaddr</i> argument if you do not enter a <i>count</i> argument.
-u	“Unzombifies” the addresses in the Current Sessions Table and updates the corresponding <i>Sbr_IpPoolOrdinal</i> ’s in the Current Sessions Table.
-h	Displays help for the AddRange.sh script.

Example

The following example adds an address range consisting of 15 addresses to the B-GOLD pool and confirms the address range was added.

```
hadm$- AddRange.sh B-GOLD 192.168.0.0 15
hadm$- ShowPools.sh
IpPools:
+-----+-----+-----+-----+-----+-----+
|           Name |      Ord |      Count | %Free | %Cache | %InUse |
```

+-----+-----+-----+-----+-----+-----+-----+						
(z o m b i e)	0	0	----	----	----	
B-GOLD	1	15	100.0	0.0	0.0	
+-----+-----+-----+-----+-----+-----+-----+						

Usage Notes

When the **AddRange.sh** script is executed, it determines the status of the addresses in the range. If the addresses do not exist in the **IpAddrs** database table, the **AddRange.sh** script adds them to the table. If the addresses exist in the **IpAddrs** database table and are flagged as zombies, the script updates the pool ordinal of each address to reflect its address pool assignment.

DelRange.sh

The **DelRange.sh** script enables you to delete an IP address range from the **IpRanges** database table and delete the addresses in the IP address range from the **IpAddrs** database table.

Syntax

All SBR Carrier nodes must be in management mode.

```
DelRange.sh startaddr
DelRange.sh -a
DelRange.sh -z
DelRange.sh -h
```

Options

Table 90: DelRange.sh Options

Option	Description
startaddr	Specifies the starting IP address of the range you want to delete.
-a	Deletes all ranges.
-z	Indicates whether zombie IP addresses were created.
-h	Displays help for the DelRange.sh script.

Example

The following example removes the address range starting with 192.168.0.0 from the B-GOLD pool and confirms the address range was deleted.

```

hadm$- ShowRanges.sh
IpRanges:
+-----+-----+-----+-----+
|          Pool |      StartAddr |      EndAddr |      Count |
+-----+-----+-----+-----+
|          B-GOLD | 0. 0. 0. 10 | 0. 0. 0. 15 |          6 |
|          B-GOLD | 192.168. 0. 0 | 192.168. 0. 14 |          15 |
+-----+-----+-----+-----+

hadm$- DelRange.sh 192.168.0.0
hadm$- ShowRanges.sh
IpRanges:
+-----+-----+-----+-----+
|          Pool |      StartAddr |      EndAddr |      Count |
+-----+-----+-----+-----+
|          B-GOLD | 0. 0. 0. 10 | 0. 0. 0. 15 |          6 |
+-----+-----+-----+-----+

```

Usage Notes

- You are not prompted to confirm that you want to delete an address range when you execute the **DelRange.sh** command.
- When you execute the **DelRange.sh** script, Steel-Belted Radius Carrier high availability transfers the addresses in the deleted range to the zombie pool to preserve their usage statistics.
- Put your SBR Carrier servers into management mode before you execute **DelRange.sh**.

ShowRanges.sh

The **ShowRanges.sh** script enables you to display the starting and ending address of each IP address range, the number of addresses in the range, and the name of the pool to which a range belongs.

Syntax

```

ShowRanges.sh
ShowRanges.sh -r
ShowRanges.sh -c
ShowRanges.sh -h

```

Options

Table 91: ShowRanges.sh Options

Option	Description
-r	Orders the range primarily by range start/end addresses, and secondarily by pool name.
-c	Orders the range primarily by count, and secondarily by range or pool name according to how the -r argument was specified.
-h	Displays help for the ShowRanges.sh script.

Example

The following example displays the output of the **ShowRanges.sh** script, which indicates that the B-GOLD IP address pool consists of one range of 15 IP addresses.

```
hadm$- ShowRanges.sh
IpRanges:
+-----+-----+-----+-----+
| Pool           | StartAddr      | EndAddr         | Count          |
+-----+-----+-----+-----+
| B-GOLD         | 192.168. 0. 0  | 192.168. 0. 14  | 15             |
+-----+-----+-----+-----+
```

Usage Notes

- The output of the **ShowRanges.sh** script is sorted alphabetically by pool name, with ranges within an address pool being sorted numerically.
- IP addresses in the output of the **ShowRanges.sh** script are padded with spaces to improve readability. Before you can use an IP address copied from the **ShowRanges.sh** output as an argument for another command (such as **AddRange.sh**, **DelRange.sh**, **DelSession.sh**, **ShowAddrs.sh**, or **ShowSessions.sh**), you must remove the embedded spaces.

KillZombieAddrs.sh

A zombie address or pseudo-address is an entry in the **IpAddrs** database table that is not associated to any pool defined in the **IpPools** database table. A zombie pool is defined in the **ipAddrs** database table and given an ordinal number of 0 with the name “(z o m b i e)”. No other pool can have space-characters or lowercase characters in its name.

When you change pool settings, some addresses may be shuffled amongst different address pools. As a result, some addresses may become disassociated from any defined pool. These zombie addresses are held in the zombie pool (a temporary holding tank for addresses not currently assigned to a pool). The

zombie pool preserves the **LastUse** timestamp on these addresses to support SSR aging strategy, which enables the addresses to be moved from one pool to another.

The **KillZombieAddrs.sh** script deletes all zombie addresses from the **IpAddrs** and **IpRanges** database tables and the zombie address pool. After completing the reconfiguration, if you do not want to preserve the usage statistics for the affected addresses, run the **KillZombieAddrs.sh** script to delete the IP address pools or ranges (Steel-Belted Radius Carrier high availability puts the affected IP addresses in the zombie address pool). The **KillZombieAddrs.sh** script also updates the **IpPools** database table to remove zombie address information.

You cannot remove selected addresses (while leaving others) with the **KillZombieAddrs.sh** script.

Syntax

All SBR Carrier nodes must be in management mode.

```
KillZombieAddrs.sh
KillZombieAddrs.sh -h
```

Options

Table 92: KillZombieAddrs.sh Options

Option	Description
-h	Displays help for the KillZombieAddrs.sh script.

Example

The following example displays the output of the **ShowPools.sh** and **ShowAddrs.sh** scripts before and after the **KillZombieAddrs.sh** script is executed.

```

hadm$- ShowPools.sh -v
IpPools:
+-----+-----+-----+-----+-----+-----+
| Name          | Ord  | Count | %Free | %Cache | %InUse |
+-----+-----+-----+-----+-----+-----+
| (z o m b i e) | 0    | 5     | 80.0  | 20.0   | 0.0    |
| B-GOLD        | 1    | 6     | 33.3  | 66.6   | 0.0    |
+-----+-----+-----+-----+-----+-----+

hadm$- ShowAddrs.sh
IpAddrs:
+-----+-----+-----+-----+-----+-----+
| IpAddress      | Pool          | Cache | InUse | LastUse              |
+-----+-----+-----+-----+-----+-----+
| 0. 0. 0. 10    | B-GOLD        | 0     | 0     | 2005-11-02 14:08:36 |

```

```

| 0. 0. 0. 11      | B-GOLD          | 24   | 0   | 0000-00-00 00:00:00 |
| 0. 0. 0. 12      | B-GOLD          | 0    | 0   | 2005-11-02 14:08:47 |
| 0. 0. 0. 13      | B-GOLD          | 24   | 0   | 0000-00-00 00:00:00 |
| 0. 0. 0. 14      | B-GOLD          | 24   | 0   | 0000-00-00 00:00:00 |
| 0. 0. 0. 15      | B-GOLD          | 24   | 0   | 0000-00-00 00:00:00 |
| 192.168. 0. 0    | (z o m b i e)   | 0    | 0   | 2005-11-02 14:06:42 |
| 192.168. 0. 1    | (z o m b i e)   | 0    | 0   | 2005-11-02 14:07:05 |
| 192.168. 0. 2    | (z o m b i e)   | 0    | 0   | 2005-11-02 14:07:33 |
| 192.168. 0. 3    | (z o m b i e)   | 0    | 0   | 2005-11-02 14:07:28 |
| 192.168. 0. 4    | (z o m b i e)   | 23   | 0   | 2005-11-02 13:52:49 |
+-----+-----+-----+-----+-----+

```

```
hadm$- KillZombieAddr.sh -v
```

```
hadm$- ShowPools.sh
```

```
IpPools:
```

```

+-----+-----+-----+-----+-----+-----+
| Name          | Ord  | Count | %Free | %Cache | %InUse |
+-----+-----+-----+-----+-----+-----+
| (z o m b i e) | 0    | 0     | ----  | ----  | ----  |
| B-GOLD        | 1    | 6     | 33.3  | 66.6  | 0.0   |
+-----+-----+-----+-----+-----+-----+

```

```
hadm$- ShowAddr.sh
```

```
IpAddr:
```

```

+-----+-----+-----+-----+-----+-----+
| IpAddress      | Pool          | Cache | InUse | LastUse          |
+-----+-----+-----+-----+-----+-----+
| 0. 0. 0. 10    | B-GOLD        | 0     | 0     | 2005-11-02 14:08:36 |
| 0. 0. 0. 11    | B-GOLD        | 24    | 0     | 0000-00-00 00:00:00 |
| 0. 0. 0. 12    | B-GOLD        | 0     | 0     | 2005-11-02 14:08:47 |
| 0. 0. 0. 13    | B-GOLD        | 24    | 0     | 0000-00-00 00:00:00 |
| 0. 0. 0. 14    | B-GOLD        | 24    | 0     | 0000-00-00 00:00:00 |
| 0. 0. 0. 15    | B-GOLD        | 24    | 0     | 0000-00-00 00:00:00 |
+-----+-----+-----+-----+-----+-----+

```

ShowAddr.sh

The **ShowAddr.sh** script enables you to display the contents of the **IpAddr** database table.

Syntax

```

ShowAddr.sh [-a ][startaddr [ count |endaddr ]]
ShowAddr.sh -o
ShowAddr.sh -h

```

Options

Table 93: ShowAddr.sh Options

Option	Description
-a	If the -a argument is entered, information about IP addresses associated with all nodes is displayed.
startaddr	<p>If used without an endaddr argument, only information about the specified address is displayed.</p> <p>If used with an endaddr argument, only information about addresses in the specified address range is displayed.</p> <p>If used with the count argument but without an endaddr argument, only addresses within the startaddr...startaddr+count-1 are displayed.</p>
count	Specifies the number of addresses in the address range.
endaddr	<p>Specifies that only information about addresses in the address range between startaddr and endaddr is displayed.</p> <p>You cannot use the endaddr argument without a startaddr argument.</p>
-o	<p>Displays the output as an unordered list. You can use this argument when the IpAddr table is very, very large and you want to maximize efficiency.</p> <p>If used with a startaddr argument, displays only that address only.</p> <p>If used with startaddr and endaddr arguments, displays only those addresses between the specified address range.</p> <p>If used with the count argument and an addr argument, only addresses within the startaddr...startaddr+count-1 are displayed.</p>
-h	Displays help for the ShowRanges.sh script.

Example

The following example displays the output of the **ShowAddr.sh** script, which lists the status of each IP address configured in the Steel-Belted Radius Carrier high availability database. The following example indicates that node 24 has four IP addresses in its cache, that node 23 has three IP addresses in its cache, and that one IP address is currently in use by node 24.

```
hadm$- ShowAddr.sh
IpAddr:
```

IpAddress	Pool	Cache	InUse	LastUse
192.168. 0. 0	B-GOLD	24	0	2005-11-02 13:53:02
192.168. 0. 1	B-GOLD	24	0	2005-11-02 13:52:47
192.168. 0. 2	B-GOLD	24	0	2005-11-02 13:52:47
192.168. 0. 3	B-GOLD	24	0	2005-11-02 13:52:47
192.168. 0. 4	B-GOLD	23	0	2005-11-02 13:52:49
192.168. 0. 5	B-GOLD	23	0	2005-11-02 13:52:49
192.168. 0. 6	B-GOLD	0	0	2005-11-02 13:52:49
192.168. 0. 7	B-GOLD	0	0	2005-11-02 13:52:49
192.168. 0. 8	B-GOLD	0	0	2005-11-02 13:52:49
192.168. 0. 9	B-GOLD	0	0	2005-11-02 13:52:49
192.168. 0. 10	B-GOLD	0	24	2005-11-02 13:53:08
192.168. 0. 11	B-GOLD	0	0	2005-11-02 13:52:49
192.168. 0. 12	B-GOLD	0	0	2005-11-02 13:52:49
192.168. 0. 13	B-GOLD	23	0	2005-11-02 13:52:49
192.168. 0. 14	B-GOLD	0	0	2005-11-02 13:52:49

Output Notes

Table 94: ShowAddrs.sh Output Fields

Field	Description
IPAddress	Identifies the IP address in dotted-decimal format (with space padding for readability).
Pool	Identifies the pool using the IP address.
Cache	Identifies the SSR node ID of the SBR Carrier server that has cached the IP address. A value of 0 indicates the address is not cached.
InUse	Identifies the SSR node ID of the SBR Carrier server that assigned the IP address to a user.

Table 94: ShowAddr.sh Output Fields (continued)

Field	Description
LastUse	<p>If the address is in a cache (if the Cache field is not 0), identifies the date and time on which the IP address was last cached by a server.</p> <p>If the address is currently assigned to a user (if the InUse field is not 0), identifies the date and time on which the IP address was assigned to the user or on which the time stamp was updated.</p> <p>If the address is currently available (if the Cache and InUse fields are 0), identifies the date and time on which the IP address was freed at the end of a user session.</p>

Usage Notes

If SBR Carrier does not receive an **Accounting-Stop** message for an address in the **InUse** column, it does not return the address to the pool of available addresses immediately. SBR Carrier sweeps the Current Sessions Table periodically for stale sessions, where the **StaleSessionTimeoutSecs** setting in **radius.ini** specifies the period. To remove an address from the InUse column manually, use the **DelSession.sh** command.

BackupIP.sh

The script enables you to create a backup of IP address pools in your SSR database. You can make multiple backup copies using this script. During the backup process, the values of **Cache** and **InUse** columns in the database are set to 0 in order to aid the restoration of the IP address pools after the database creation.

Syntax

```
BackupIP.sh filename
```

```
BackupIP.sh -h
```

Options

Table 95: BackupIP.sh Options

Option	Description
<i>filename</i>	Specifies the filename in which you want to backup IP address pools.
-h	Displays help for the BackupIP.sh script.

Example

The following example (1) lists the address pools configured for Steel-Belted Radius Carrier high availability and (2) creates a backup of the address pools in the filename called **Backup1**.

```

hadm@- ./ShowPools.sh
IpPools:
+-----+-----+-----+
|           Name |   Ord |   Count |
+-----+-----+-----+
|   (z o m b i e) |     0 |        0 |
|           POOL1 |     4 |       1,000 |
|           POOL2 |     2 |       1,000 |
|           POOL3 |     3 |        500 |
+-----+-----+-----+
Total Pools :      3 + 1 zombie pseudo-pool.
Total Ranges:      3 + 0 zombie pseudo-ranges.
Total Addrs : 2,500 + 0 zombie pseudo-addrs.

hadm@- ./BackupIP.sh Backup1
Backup Process Started
Backup Process Completed

```

Usage Notes

The backed up file can be restored using the **RestoreIP.sh** script.

RestoreIP.sh

The script enables you to restore the IP address pools from the backup file you created in your SSR database.

Syntax

```

RestoreIP.sh filename

RestoreIP.sh -h

```

Options

Table 96: RestoreIP.sh Options

Option	Description
<i>filename</i>	Specifies the filename that contains the backup.
-h	Displays help for the RestoreIP.sh script.

Example

The following example displays the output of the **ShowPools.sh** script before and after the **RestoreIP.sh** script is executed.

```

hadm@- ./ShowPools.sh
IpPools:
+-----+-----+-----+
|           Name |   Ord |   Count |
+-----+-----+-----+
|   (z o m b i e) |     0 |       0 |
|           POOL1 |     4 |    1,000 |
+-----+-----+-----+
Total Pools :      1 + 1 zombie pseudo-pool.
Total Ranges:      1 + 0 zombie pseudo-ranges.
Total Addrs : 1,000 + 0 zombie pseudo-addrs.

hadm@- ./RestoreIP.sh Backup1
WARNING - Please confirm that DestroyDB.sh and CreateDB.sh scripts has been
executed; OK <yes|no>  yes
SBRs must be offline or in ManagementMode; OK? <yes|no>  yes
Restoring Process Started
Restoring Process Completed.

hadm@- ./ShowPools.sh
IpPools:
+-----+-----+-----+
|           Name |   Ord |   Count |
+-----+-----+-----+
|   (z o m b i e) |     0 |       0 |
|           POOL1 |     5 |    1,000 |
|           POOL2 |     6 |    1,000 |
|           POOL3 |     7 |       500 |
+-----+-----+-----+
Total Pools :      3 + 1 zombie pseudo-pool.
Total Ranges:      3 + 0 zombie pseudo-ranges.
Total Addrs : 2,500 + 0 zombie pseudo-addrs.

```

Usage Notes

Executing the **RestoreIP.sh** script deletes the existing database, creates a new database, and restores the IP address pools from the backup file you created in your SSR database.

The values in the **ord** column may get changed after restoring the IP pools.

NOTE: Place the SBR Carrier servers into management mode before you execute **RestoreIP.sh**.

SSR Session Management

Session management scripts track and allow modification of sessions. These scripts are executed on management nodes.

Session Management Scripts

ShowSessions.sh

The **ShowSessions.sh** script enables you to display the contents of the Current Sessions Table, which tracks IP addresses used by customers. Based on your customization, the following Current Sessions Table fields may be displayed:

```
CORE
UniqueSessionId
    CreationTime
ExpirationTime
    Ipv4Address
    IpAddrPool
    NasName
    Status
UserConcurrencyId
MobileIpType
3gpp2ReqType
WimaxClientType
WimaxAcctFlows
Ipv6Address
FEATURE
AcctAutoStop
SessionTimeout
ClassAttribute
OPTIONAL
UserName
AcctSessionId
TransactionId
NasPortType
NasPort
```

```
CallingStationId
CalledStationId
MobileCorrelationId
Ipv6InterfaceId
Ipv6Prefix
NasIpv4Address
NasIpv6Address
RADATTR
{Specified by admin -- Customizable Current Sessions Table feature}
PRIVATE
{Specified by admin -- Customizable Current Sessions Table feature}
```

Syntax

```
ShowSessions.sh
ShowSessions.sh -a
ShowSessions.sh -c
ShowSessions.sh -o
ShowSessions.sh -u username [ startaddr [ endaddr]]
ShowSessions.sh startaddr [ endaddr]]
ShowSessions.sh -h
ShowSessions.sh [-c] -n NASName
```

Options

Table 97: ShowSessions.sh Options

Option	Description
-a	Displays information for all sessions.
-c	Displays only the count of the current session.
-o	Displays the output as an unordered list. If used with a <i>startaddr</i> argument, displays the session with that address only. If used with <i>startaddr</i> and <i>endaddr</i> arguments, displays only those sessions that have addresses between the specified address range.
-u <i>username</i>	Displays only those sessions for the specified user.

Table 97: ShowSessions.sh Options (continued)

Option	Description
startaddr	<p>If used without an endaddr argument, only information about sessions the specified address is displayed.</p> <p>If used with an endaddr argument, only information about sessions addresses in the specified address range is displayed.</p>
endaddr	<p>Specifies that only information about sessions with addresses in the address range between startaddr and endaddr is displayed.</p> <p>You cannot use the endaddr argument without a startaddr argument.</p>
-n NASName	<p>Displays information about the sessions associated with the specified NAD.</p> <p>If used with the -c option, displays the number of sessions associated with the specified NAD.</p>
-h	Displays help for the ShowSessions.sh script.

Example

The following example displays the output of the **ShowSessions.sh** script and shows the number of current sessions and all of the information for a session:

```

hadm$- ShowSessions.sh -c
Total Sessions: 1.

hadm$- ShowSessions.sh -a
CurrentSessions:
+ -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- + (1)
CORE
    UniqueSessionId: 'e86cb6d3717ab5331c68e6bf2b8ec6ef'x
      CreationTime: 2017-08-17 12:44:58 (TZ=+00:00)
      ExpirationTime: 2017-08-18 12:44:58 (TZ=+00:00)
      Ipv4Address: 10.212.10.17
      IpAddrPool: (n u l l)
      NasName: "LOCALHOST"
      Status: Active (2)
UserConcurrencyId: (n u l l)
  MobileIpType: 0
    3gpp2ReqType: 0
  WimaxClientType: 0

```


Options

Table 98: DelSession.sh Options

Option	Description
-a	Specifies that you want to delete all sessions from the Current Sessions Table.
-f	Specifies that you want to force the deletion of the specified session or sessions. Using the -f option is appropriate when SBR Carrier is not running, and you want to delete sessions directly from the database. NOTE: Do not use the -f option when SBR Carrier is running. Doing so prevents SBR Carrier from performing other cleanup tasks, such as sending Auto Acct. Stop messages.
addr	Specifies the IP address of the session you want to delete from the Current Sessions Table.
uniquesessionid	Specifies the ID of the session you want to delete from the Current Sessions Table. Necessary for sessions that do not have an associated address, such as phantom sessions that are not managed by SBR Carrier.
-p Poolname	Specifies that all sessions associated with the specified IP address pool should be deleted from the Current Sessions Table.
-n NASName	Specifies that all sessions associated with the specified NAD should be deleted from the Current Sessions Table.
-l limit	Specifies the number of session entries (associated with provided pool name or NAD) to be deleted in sequential order from the Current Sessions Table.
-h	Displays help for the DelSession.sh script.

Example

The following shows all IP addresses and then deletes all sessions.

```
hadm$- ShowAddrs.sh
```

```
IpAddrs:
```

```
+-----+-----+-----+-----+-----+
|      IpAddress      |      Pool      | Cache | InUse |      LastUse      |
+-----+-----+-----+-----+-----+
| 192.168.  0.   8 |      B-GOLD     |    0  |    0  | 2005-11-02 13:52:49 |
```



```
| 192.168. 0. 9 |      B-GOLD |      0 |      0 | 2005-11-02 13:52:49 |
| 192.168. 0. 10 |     B-GOLD |      0 |      0 | 2005-11-02 14:00:11 |
| 192.168. 0. 11 |     B-GOLD |      0 |      0 | 2005-11-02 13:52:49 |
+-----+-----+-----+-----+
hadm$- DelSession.sh -af
This will delete all sessions; OK? <yes|no> yes
This will set all addrs not-InUse; OK? <yes|no> yes
SBRs must be offline; OK? <yes|no> yes
hadm$- ShowSessions.sh -a
CurrentSessions:
+-----+ (end)
```

Usage Notes

You are not prompted to confirm that you want to delete a session when you execute the **DelSession.sh** script.

User Concurrency Scripts

ShowUserConc.sh

The **ShowUserConc.sh** script enables you to display the contents of the Sbr_UserConcurrency table.

Syntax

```
ShowUserConc.sh
ShowUserConc.sh -a
ShowUserConc.sh username
ShowUserConc.sh -h
```

Options

Table 99: ShowUserConc.sh Options

Option	Description
-a	Displays all user concurrency information.
<i>username</i>	Displays all entries whose UserConcurrency ID has the specified <i>username</i> (matching is case-insensitive). UserConcurrency ID's use the format: <i>authtype-username</i> , where <i>authtype</i> identifies an authentication type or method (uses the format <i>number-</i> or <i>proxynome-</i>
-h	Displays help for the ShowUserConc.sh script.

Example

The following example displays all of the user concurrency information in the `Sbr_UserConcurrency` table.

```

hadm$- ShowUserConc.sh -a
UserConcurrency:
+-----+-----+
| Id      | Count  |
+-----+-----+
| 0-JANE  | 1      |
| 0-JILL  | 10     |
| 0-JOE   | 3      |
+-----+-----+

```

The **UserConcurrencyID** consists of a prefix identifying the authentication method, a hyphen, and the username. The username is the same as the **Sbr_UserName** field in the Current Sessions Table.

The authentication method prefix specifies the domain/context in which the username is to be interpreted (a managed database of account names). There are two authentication methods:

- Enumerated—Identified by an integer in the range 0–999.
- Proxy—Identified by a target-name (for simple proxy) or realm-name (for extended proxy).

For example, **0-JANE**, 0 indicates a “native user” enumerated authentication method. See [Table 100 on page 676](#) for a complete list.

Table 100: Enumerated Authentication Methods

Enumerated Prefix	Method Description
0-User	Native user
13-User	UNIX user
14-User	UNIX group
200-User	All generic plug-in authentication methods
201-User	SIM user
400-User	LDAP plug-in authentication method

Table 102: Configuration Files for Administration Scripts

File Name	Function	Default Value
ErrMsg.txt	Error message file.	Empty (no errors recorded)
DBName.txt	Specifies the default name for the database created with the CreateDB.sh script (described on “CreateDB.sh” on page 641).	SteelBeltedRadius
DBSock.txt	<p>Specifies the socket on which the mysqld process is listening.</p> <p>If the DBSock.txt file is present and not empty, then the administration script assumes mysqld is running on the local machine, and listening on the specified socket, and the administration scripts communicate with mysqld using this socket.</p> <p>If the DBSock.txt file is absent or empty, then the administration scripts use DBHost.txt/DBPort.txt to communicate with mysqld.</p>	/opt/JNPRhadm/.mysql.sock
DBHost.txt	Specifies the name of the host on which mysqld is running.	localhost
DBPort.txt	Specifies the default port for communicating with mysqld.	3001
DBUser.txt	Specifies the name of the default SSR identity (account).	hadmsbr
DBPwd.txt	Specifies the password of the default SSR identity (account).	someStrongHadmSbrPassword
Scratch.txt	A scratch file used by the scripts.	Empty (no value present)

Table 102: Configuration Files for Administration Scripts *(continued)*

File Name	Function	Default Value
DBtime zone.txt	<p>Specifies the time zone used by the administration scripts. Available choices are: UTC (the default value of 00:00), time zone of the machine where the administration scripts are running, or the time zone of the machine that hosts the management node.</p> <p>If you need to change the time zone, you must edit the DBtime zone.txt field. For example, to set the time zone to Pacific Standard Time (PST), first set the DBtime zone.txt field to -07:00.</p> <p>This setting affects the timestamp display in the ShowAddr.sh and ShowSessions.sh scripts. However, this setting does not affect the timestamps held in the database.</p>	+00:00

9

PART

Optional Concurrency Module

Managing User Concurrency with Session State Register | **682**

Managing Concurrency with Attributes in Session State Register | **688**

Managing User Concurrency with Session State Register

IN THIS CHAPTER

- Overview | 682
- How User Concurrency Works | 683

This chapter describes how to manage user concurrency across multiple servers connected to the same database cluster. You can limit the number of active user connections across all SBR Carrier nodes in the Session State Register cluster.

This chapter documents the default behavior of the SSR cluster that tracks concurrency using usernames; see [“Managing Concurrency with Attributes in Session State Register” on page 688](#) for information about the expanded capabilities of the Optional User Concurrency and Wholesale module.

These topics are in this chapter:

Overview

You can set a limit on the maximum number of concurrent connections that a user can have by counting the number of connections associated with a username. When user concurrency management is enabled, SBR Carrier servers in the cluster compare the number of active connections for a user to the user's maximum limit each time a new connection is requested. If a new connection exceeds the limit, Session State Register may reject or allow the additional connection. In either case, the event is logged in the server log.

NOTE: When counting connections, Session State Register does not distinguish between multi-link connections and new user authentication attempts.

How User Concurrency Works

To support user concurrency on all SBR Carrier servers in a cluster, a simple database table (**Sbr_UserConcurrency**) in the NDB database stores two fields: the user ID and the current count of user sessions. The **Sbr_UserConcurrency** table is created automatically when you run the **CreateDB.sh** script to create a database.

NOTE: User concurrency limits (quotas) are enforced during authentication. If the SBR Carrier server is being used as an accounting-only node, user concurrency limits are not enforced.

When an Access-Request is received, the user is authenticated and an ID consisting of an authentication method prefix, a hyphen, and a username is created, such as **0-JANE**.

This example script displays all of the user concurrency information in the **Sbr_UserConcurrency** table.

```
hadm$- ShowUserConc.sh -a
UserConcurrency:
+-----+-----+
| Id      | Count |
+-----+-----+
| 0-JANE  | 1     |
| 0-JILL  | 10    |
| 0-JOE   | 3     |
+-----+-----+
```

The ID, **0-JANE**, is used to look up the count of open sessions in the table. If this number exceeds the number of sessions allowed, as determined either by native user configuration, LDAP/SQL lookup, or Funk VSA in the Access-Request, the request is rejected. Otherwise, the count is incremented and an Access-Accept is sent.

When an Accounting-Stop is received, the count is decremented. The Class attribute sent in the Access-Accept must be returned in the Accounting-Request so that the user can be matched with the ID in the table for user session counts.

The **DelSession.sh** administrative script decrements the **Count** field as required. For more details about the **DelSession.sh** script, see [“DelSession.sh” on page 673](#).

UserConcurrencyID Construction

The **UserConcurrencyID** consists of a prefix identifying the authentication method, a hyphen, and a UserName.

Username

The username is the same as the **Sbr_UserName** field in the Current Sessions Table.

The size of the **Sbr_UserName** field in the Current Sessions Table and the **ID** field in the **Sbr_UserConcurrency** table must be the same. By default, the SQL datatype and size definition is: VARCHAR(32) utf8. It has a customizable character set and size, with a maximum size of 84 characters. The **Sbr_UserName** field must be large enough to hold all user concurrency IDs.

NOTE: The username is not guaranteed to be unique among all users; it is only unique when combined with an authentication method.

For details about the **Sbr_UserName** field and the Current Sessions Table, see the section on *Customizing the SSR Database Current Sessions Table* in the *SBR Carrier Installation Guide*.

Authentication Method

The prefix of the **UserConcurrency ID** is determined by the authentication method. The authentication method specifies the domain/context in which the username is to be interpreted (a managed database of account names).

There are two authentication methods:

- Enumerated
- Proxy

Enumerated Authentication Methods

Enumerated authentication methods are identified by an integer in the range 0–999. In the previous example, **0-JANE**, 0 indicates a “native user” enumerated authentication method. [Table 103 on page 684](#) contains the list of supported methods and prefixes.

Table 103: Enumerated Authentication Methods

Enumerated Prefix	Method Description
0-User	Native user
13-User	UNIX user
14-User	UNIX group
200-User	All generic plug-in authentication methods
201-User	SIM user

Table 103: Enumerated Authentication Methods (*continued*)

Enumerated Prefix	Method Description
400-User	LDAP plug-in authentication method
500-User	TLS plug-in authentication method
600-User	TTLS plug-in authentication method
700-User	PEAP plug-in authentication method
800-User	JDBC plug-in authentication method
900-User	Oracle plug-in authentication method

Proxy Authentication Methods

Proxy authentication methods are identified by a target-name (for simple proxy) or realm-name (for extended proxy). When defining proxy authentication methods, you must ensure that the target-name or realm-name does not exceed the length of the **Sbr_UserName** field in the Current Sessions Table and the **ID** field in the **Sbr_UserConcurrency** table.

To save space in the NDB database and use minimally sized **Sbr_UserConcurrencyId** and **Id** fields, you can name your proxy authentication methods using letters such as A or B.

For proxy authentication methods, the **UserConcurrencyID** comprises a string-concatenation of the target-name or realm-name enclosed in angle brackets, a hyphen, and a username; for example: **<allegro>-JoeUser**.

NOTE: Because they are protected characters used to enclose target and realm names, you cannot use angle bracket or the “greater than/less than” characters < and > as part of the target-name or realm-name.

Authentication Method Consistency

You must make sure that all of the authentication methods used by the Session State Register servers in a farm use the same name definition for the same authentication method.

For example, if server #1 defines a JoeUser, native-account for the real-world person Joseph A. User; and server #2 defines a JoeUser, native-account for the real-world person Josephine B. Userette, then both will be assigned a **UserConcurrencyID** of **0-JoeUser** because 0 is the integer value associated with the native-user authentication method. Their user concurrency counts will be corrupt. Similarly, corruption

will also occur if server #1 and server #2 both define proxy authentication methods with the same name which don't refer to the same real-world proxy target/realm.

Retrospective Dynamicity

Dynamicity refers to your ability to modify a user's concurrency limit after the user's account has been created. As a result, the user could have some pre-existing sessions after you change the concurrency limit. Retrospectivity refers to what happens to those pre-existing sessions when you change the user's concurrency limit.

In a case where a user has no concurrency limit imposed, to optimize performance, set the **UserConcurrencyId** field in the Current Sessions Table to **NULL**. No corresponding record for that user will be created in the **Sbr_UserConcurrency** table. This minimizes the number of accesses to the database because it is not necessary to update the **Sbr_UserConcurrency** table when no concurrency limit has been imposed on the user. This represents the no-retrospective-dynamicity model which is the model supported by Session State Register.

Consequences of the No-Retrospective-Dynamicity Model

The following examples explain the consequences of the no-retrospective-dynamicity model.

- **Example #1:** JoeUser (a native user) initially has no concurrency limits imposed so his concurrency count is not being tracked in the Current Sessions Table or the **Sbr_UserConcurrency** table. This enables Joe to start up any number of sessions. In this example, Joe has five existing sessions. Then, the administrator decides to impose a concurrency limit of three on Joe. Joe's previous existing five sessions do not count against his new concurrency limit of three. Instead, this allows Joe to have up to eight simultaneous sessions until some of them start to expire. As a result, the administrator's intent of limiting sessions to three is not being honored (temporarily).
- **Example #2:** JoeUser initially has a concurrency limit of seven imposed so his concurrency count is being tracked in the Current Sessions Table and the **Sbr_UserConcurrency** table. In this example, Joe has five existing sessions. Then, the administrator decides to re-set Joe's concurrency limit to three. Joe still has five previous sessions because Session State Register does not try to kill two of Joe's sessions. However, Session State Register will not allow him to create any new sessions until he drops below the new limit of three. As a result, the administrator's intent of limiting sessions to three is not being honored (temporarily).

Benefits of the No-Retrospective-Dynamicity Model

The benefits of the no-retrospective-dynamicity model are:

- There is a distinct performance advantage when no concurrency limit is imposed since it requires less accesses to the database, while it causes no adverse effects on performance when concurrency limits are imposed.
- If an administrator really wants to kill Joe's sessions, he or she can use the **DelSession.sh** admin script.

- Sessions have a limited lifetime, so they will die off eventually.
- The no-retrospective-dynamicity model is the most common implementation of limits (quotas), so it is already familiar to administrators.

Managing Concurrency with Attributes in Session State Register

IN THIS CHAPTER

- [Overview | 688](#)
- [How Attribute-Based Concurrency Works | 689](#)
- [Configuring Attribute-Based Concurrency | 690](#)

This chapter describes how to manage concurrency in a Session State Register cluster that uses the Optional Concurrency and Wholesale Module. This module expands the default user concurrency method by allowing concurrent sessions to be tracked by other attributes than the username. (Basic username concurrency is described in [“Managing User Concurrency with Session State Register” on page 682.](#))

These topics are in this chapter:

Overview

Over and above the default username concurrency controls of the Session State Register, the Optional Concurrency and Wholesale Module provides expanded capabilities to track user concurrency using any user attribute.

NOTE: Concurrency on any attribute is only supported when SBR Carrier is running as part of a Session State Register cluster. This feature is not supported on a standalone SBR Carrier server.

NOTE: RADIUS Class Attributes are an optional part of the RADIUS standard, but not all network access devices fully support attributes. In order to work at the RADIUS server level, the network NADs must support the Radius Class Attribute. If they do not, the attribute data the server requires is not created.

As with username-based concurrency, you can set a limit on the maximum number of concurrent connections that a user can have. SBR Carrier servers in the cluster compare the number of active connections for a particular string attribute up to the maximum limit each time a new connection is requested.

If a new connection exceeds the limit, Session State Register may reject or allow the additional connection. In either case, the event is logged in the server log.

NOTE: When counting connections, Session State Register does not distinguish between multi-link connections and new authentication attempts.

How Attribute-Based Concurrency Works

To support attribute-based concurrency on all SBR Carrier servers in a cluster, the **Sbr_UserConcurrency** table in the cluster's SSR database contains fields for specified attributes in addition to the default user ID and the current count of sessions for each specified attribute. The **Sbr_UserConcurrency** table is created automatically when you run the **CreateDB.sh** script on each management node to create the database.

NOTE: Concurrency limits (quotas) are enforced during authentication. If the SBR Carrier server is being used as an accounting-only node, user concurrency limits are not enforced.

When an Access-Request is received, the user is authenticated and an ID consisting of an authentication method prefix, a hyphen, and a username or attribute is created in the table. Each specified attribute is tracked in other columns.

The ID and attribute(s) are used to look up the count of open sessions in the table. If this number exceeds the number of sessions allowed, as determined either by native user configuration, LDAP/SQL lookup, or Funk VSA in the Access-Request, the request is rejected. Otherwise, the count is incremented and an Access-Accept is sent.

When an Accounting-Stop is received, the count is decremented. The Class attribute sent in the Access-Accept must be returned in the Accounting-Request so that the user can be matched with the ID in the table for user session counts.

The **DelSession.sh** administrative script decrements the **Count** fields as required. For more details about the **DelSession.sh** script, see [“DelSession.sh” on page 673](#).

Configuring Attribute-Based Concurrency

If the Optional Concurrency and Wholesale Module is not enabled during initial installation, it needs to be activated by entering a license number in the Web GUI too.

There is not a default setting for attribute-based concurrency because attributes attached to a user can vary from site to site. (For tunneled authentication methods, any attribute available in the inner request may be used.)

To set up the Optional Concurrency and Wholesale Module, you edit the **/opt/JNPRhadm/UserConcurrency.sql** file to increase the size of the ID field in the user concurrency table and edit the **/opt/JNPRsbr/radius/radius.ini** file to specify the attribute(s) to track.

Setting the Size of the ID Field in the User Concurrency Table

To increase the size of the ID Field in the user concurrency table:

1. Log in as hadm.
2. Change directories to **/opt/JNPRhadm/**.
3. Edit **UserConcurrency.sql** to increase the size of the ID field.
 - a. Near the top of the file, locate the line that reads:

```
Id      VARCHAR(84) CHARSET utf8 COLLATE utf8_general_ci
```

- b. Change the (84) to (235).

The modified file should look like this example:

```
#=====
```



```
CREATE TABLE Sbr_UserConcurrency
(
  Id      VARCHAR(235) CHARSET utf8 COLLATE utf8_general_ci
         NOT NULL,
  Count   INT UNSIGNED
         NOT NULL,
  PRIMARY KEY USING HASH (Id)
)
ENGINE = ndbcluster # NOTE: CreatedB.sh fiddles with this line!
;
```

```
#=====
```

Specifying the User Attribute

To specify the attribute or attributes to track:

1. Log in as root.
2. Change directories to **/opt/JNPRsbr/radius/**.
3. Edit **radius.ini** to specify the attribute or attribute to track.
 - a. Near the top of the file, locate the **[Configuration]** section and within it, the **Login-Limit-Key =** line. It looks like this example:

```
////////////////////////////////////
RADIUS.INI file - Version 8.6.R-8 (April 2020)
////////////////////////////////////
; This file defines operational characteristics of Juniper Network's
; Steel-Belted Radius server.
```

```
[Configuration]
```

```
; Login-Limits: limiting user concurrency
Apply-Login-Limits          = yes
```

```
; 'Login-Limit-Key' allows you to redefine the key used for
; login-limit concurrency checks. It is a list of attributes
; (space separated). Maximum payload length is 84 characters.
; Binary attributes will be rendered as a hex string.
; Login-Limit-Key          =
```

- b. Remove the comment semi-colon from the front of the **Login-Limit-Key** = line.
- c. Beyond the equals sign, type the user attribute or attributes to track.
 - If you enter multiple attribute names, separate them with spaces. The values of the attributes are concatenated to create the key.
 - If the field contains an empty value, it indicates that the default scheme of concatenating the username with the authentication method is used.

The modified file should resemble these examples:

```
; Login-Limit-Key          = WiMAX-QoS-Descriptor
```

or

```
; Login-Limit-Key          = WiMAX-QoS-Descriptor Login-IP-Host
```

Distributing the Files

If there are additional SBR Carrier servers in the cluster that have the Optional Concurrency and Wholesale Module installed, they must be configured identically. Copy the edited **/opt/JNPRsbr/radius/radius.ini** and **/opt/JNPRhadm/UserConcurrenty.sql** files to all other servers in the cluster.

10

PART

Managing and Controlling Sessions

Introduction to Managing and Controlling Sessions in SBR Carrier | **695**

Overview of the Optional Session Control Module | **700**

Using Web GUI to Manage and Control Sessions | **718**

Using the Command Line Utility to Manage and Control Sessions | **730**

Configuring the deviceModels.xml File | **743**

XML over HTTPS Interface | **760**

Example CoA/DM Configuration | **806**

Introduction to Managing and Controlling Sessions in SBR Carrier

IN THIS CHAPTER

- Overview of Managing and Controlling Sessions in SBR Carrier | 695

This chapter provides an introduction to the various ways you can manage sessions in SBR Carrier. The following topics are included:

If you are working on a SBR Carrier server that is part of a Session State Register cluster, see [“SSR Session Management” on page 670](#).

Overview of Managing and Controlling Sessions in SBR Carrier

Introduction

SBR Carrier tracks the status of user connections that it authenticates in a current sessions table (CST). Because the CST is based on RADIUS accounting data, the list of active sessions is accurate only if all of your network access servers (NAS) are configured to support RADIUS accounting. To accurately track session activity, you need to ensure:

- All clients in your configuration support RADIUS accounting
- All clients are configured to send accounting messages to SBR Carrier

Storing Sessions in the CST in a Standalone Server versus the SSR Cluster

Whether SBR Carrier is running in standalone mode or as part of a cluster, session information is stored in a CST. However, different files are used to set up the CST depending on whether SBR Carrier is running in standalone mode or in a SSR cluster.

Storing Sessions in the CST of a Standalone Server

For a standalone server, the CST is in local memory, and is configured with the **dbclusterlocal.gen** file when you run the configuration script. In addition, the setting of the **PersistSessions** parameter in the **radius.ini** file determines whether sessions are restored or not restored when SBR Carrier is restarted.

You cannot configure field names in the local CST (**dbclusterlocal.gen**). However, there are three predefined fields and seven generic fields you can configure using the **sessionTable.ini** file. In order to use any of the ten predefined fields, they must be mapped to attributes by editing the **sessionTable.ini** file.

Storing Sessions in the CST of the SSR Cluster

In SSR, the CST is stored in the data nodes of the back-end cluster, and is configured with the **dbclusterndb.gen** file when you run the configuration script. In SSR, you can configure the field names in the CST using the **sessionTable.ini** file.

For more information about configuring the CST for a standalone server or SSR, see the *SBR Carrier Installation Guide* for more information.

Persisting Sessions When SSR Cluster Is Down

If SBR Carrier fails to load the SSR cluster during startup, SBR Carrier reads the value of the **FallbackLocal** parameter in the **radius.ini** file. If this parameter is set to **true**, SBR Carrier uses the local CST and persists sessions in the **.hst** file. If the **FallbackLocal** parameter is set to **false**, SBR Carrier shuts down gracefully.

During SBR runtime, SBR Carrier identifies the cluster availability based on the number of NDB hard errors encountered for a NDB transaction. At the end of each transaction, SBR Carrier monitors the occurrence of hard errors for the transaction. If the number of hard errors exceeds the value set to the **NDBHardErrorThreshold** parameter in the **dbclusterndb.gen** file, SBR Carrier checks with the management node for the cluster's health. Based on the reports provided by the management node, SBR Carrier decides whether the cluster is available or down.

If SBR Carrier decides that the cluster is down and the **FallbackLocal** parameter in the **radius.ini** file is set to **true**, SBR Carrier stores sessions in the local CST. SBR Carrier constantly monitors the cluster health even after storing the session in the local CST. When the SSR cluster becomes available, SBR Carrier again stores the sessions from the local CST to the SSR cluster database. If the **FallbackLocal** parameter is set to **false**, SBR Carrier does not store sessions in the local CST even when the cluster is down.

NOTE:

- The session persistence process might be affected if the management node is not available.
- SBR Carrier does not support the synchronization of session data between the local CST and SSR cluster database. Hence, a phantom session created in the local CST cannot be made active in the SSR cluster, or vice versa. Similarly, the **Acct-Start** message in one store and the **Acct-Stop** or **Interim** message in another store do not end in successful transactions and may result in creation of duplicate sessions.
- Return list processing for IP address allocation may be affected during the session persistence.

Session Management and Control Capabilities

The session management and control capabilities in Steel-Belted Radius Carrier can be divided into two separate categories:

- The ability to view and delete active sessions
- The ability to dynamically disconnect sessions or change the state of a session

Viewing and Deleting Sessions

The ability to view and delete sessions is available with the basic Steel-Belted Radius Carrier license. With this basic license, you can search for all sessions, or specify certain criteria and display only sessions that match those criteria. You can then select a session or multiple sessions and delete them.

NOTE: Deleting a session deletes the session from the Steel-Belted Radius Carrier Current Sessions Table (CST). However, the session remains active. To inactivate a session, you must perform disconnect, which requires the optional Session Control module. For more information, see [“Overview of the Optional Session Control Module” on page 700](#).

NOTE: In order for a session to be found in the CST, the session must be running on a *controlled* device, such as a network access server (NAS), which must have sent an accounting start message to the SBR Carrier server.

Disconnecting or Changing the State of Active Sessions

Under some circumstances, you may want to disconnect or make changes to active sessions without requiring the NAS to initiate the change. For example, you may want to terminate an active user's session by issuing a Disconnect Message (DM) request to the NAS, or you may want to modify the authorization

level of an active user's session issuing a Change of Authorization (CoA) request to the NAS. For example, as a service provider, you may be required to comply with lawful intercept regulations by providing legal organizations with voice and data intercept capabilities. These might include access to private communications between organizations or individuals such as phone calls, e-mail, VoIP, or instant messaging. These lawful intercept capabilities can be performed by issuing a CoA request.

The optional Session Control module enables you to offer dynamic service changes to reinforce current service offerings. Using the Session Control module, you can customize the CoA/DM requests you want to support in your network. You can define *actions* that can be invoked on active sessions such as disconnecting an active session, increasing the bandwidth of an active session, or any other action you want to define. For more information, see [“Overview of the Optional Session Control Module” on page 700](#).

You can control sessions using Web GUI or a command line utility, or you can develop your own client management application to interface with the Steel-Belted Radius Carrier CoA/DM XML interface.

Available User Interfaces for Managing and Controlling Sessions

You can use the following user interfaces to manage and control sessions in Steel-Belted Radius Carrier:

Web GUI

You can use Web GUI to view and delete sessions, as well as execute actions on active sessions. After you define an *action* in the **deviceModels.xml** file, a button for invoking the action becomes visible in Web GUI. You simply perform a query to locate and select one or more active sessions, and then select the action button to execute the action on the selected sessions. For more information, see [“Using Web GUI to Manage and Control Sessions” on page 718](#).

Command Line Utility

You can use the command line utility to view sessions and to execute actions on active sessions. After you define an action in the **deviceModels.xml** file, you can use the action name as an argument in the command line utility to execute the action on the selected sessions.

NOTE: The command line utility (**SessionControl.sh**) is only used to manage and control sessions. It does not provide complete management of SBR Carrier servers.

For more information, see [“Using the Command Line Utility to Manage and Control Sessions” on page 730](#).

XML over HTTPS Interface

Steel-Belted Radius Carrier includes an application programming interface (API), which is a proprietary XML request/response interface that runs over an HTTPS connection (HTTPS over TLS). The Web GUI and command line utility management clients interact with this API when issuing CoA/DM (actions) requests and when receiving responses to those requests. The protocol used for this interface is mirrored on the Simple Object Access Protocol (SOAP), though it is not identical to it.

You can develop your own XML management client to integrate with this API in order to customize the CoA/DM actions you want to support in your network. If you choose to develop your own XML management client to issue CoA/DM actions, all client requests must be in the XML format specified in [“Client Request Schema Example” on page 762](#). In addition, your client management application must be capable of handling client responses from Steel-Belted Radius Carrier in the XML format specified in [“Client Response Schema Example” on page 770](#). Refer to [“Overview of the Optional Session Control Module” on page 700](#) for a discussion on how Steel-Belted Radius Carrier processes CoA/DM requests.

Administrative Scripts

If you have purchased the optional Subscriber State Register option, you can use the **ShowSessions.sh** and **DelSession.sh** scripts to view and delete sessions. For more information, see [“Session State Register Administration” on page 628](#).

Overview of the Optional Session Control Module

IN THIS CHAPTER

- [Change of Authorization/Disconnect Messages Overview | 700](#)
- [How Steel-Belted Radius Carrier Processes CoA/DM Messages | 702](#)
- [Sequence and Flow of CoA/DM Requests Through Steel-Belted Radius Carrier | 706](#)
- [Implementing CoA/DM Support | 711](#)
- [Processing Dynamic Authorization \(CoA/DM\) Messages as a Proxy Server | 713](#)
- [Processing Dynamic Authorization \(CoA/DM\) Messages as a Proxy Target | 715](#)
- [Settings to Support the Proxy CoA/DM Functionality | 715](#)

You can use the optional Session Control module to dynamically change the state of sessions previously authenticated in Steel-Belted Radius Carrier by issuing Change of Authorization and Disconnect Message (CoA/DM) requests. This chapter provides an overview of the how Steel-Belted Radius Carrier processes these requests. The following topics are included in this chapter:

NOTE: CoA/DM support is only available with the optional Session Control license.

NOTE: If the CoA/DM license is not present, the message "License check failed: ControlledDeviceMgr is disabled" appears at startup for every client.

Change of Authorization/Disconnect Messages Overview

The RADIUS protocol, as defined in RFC 2865, does not support unsolicited messages from a RADIUS server to a network access server (NAS). Under some circumstances, you may need to make changes in subscriber session characteristics without requiring the NAS to initiate the change. For example, you may want to terminate an active user's session (using a Disconnect Message), or if a user changes authorization

level, you may have to add, modify or delete authorization attributes for the user's session (using a CoA message). RFC 3576, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)," describes extended commands that provide support for unsolicited messages sent from the RADIUS server to the NAS.

The Steel-Belted Radius Carrier CoA/DM functionality is designed for mobile and wire-line operators who want to offer dynamic service changes to reinforce current service offerings. In order to use the CoA/DM functionality in Steel-Belted Radius Carrier, your NAS must support RFC 3576 Change of Authorization (CoA), Disconnect Message (DM), or the Cisco proprietary Packet of Disconnect (PoD). You can configure Steel-Belted Radius Carrier to send different requests of these packet types to deploy any of the following services:

- Prepaid scenarios—You can use CoA/DM functions to support control of data services where usage is metered by time or traffic volume, such as prepaid services or bandwidth-on-demand. When subscribers exhaust their prepaid service quota, the CoA/DM functionality enables you to either disconnect or redirect them to a webpage where they can purchase more time or data in mid-session, ensuring subscribers do not exceed their purchased limit.
- Lawful intercept—As a service provider, you may be required to comply with lawful intercept regulations by providing legal organizations with voice and data intercept capabilities. These might include monitoring both connection related information and actual session data of a specific subscriber including phone calls, e-mail, VoIP, or instant messaging, and providing this data to Law Enforcement Agencies (LEAs). These legal intercept capabilities can be performed by issuing a CoA request.
- Tiered services— CoA/DM functions can be used to provide new services, that provide new revenue creation opportunities. For example, if you are a fixed-line carrier, you may want to supplement your basic broadband service with a *turbo* option which provides subscribers with increased bandwidth-on-demand.
- Abuse control and threat mitigation—You can disconnect or quarantine subscribers who are using the network in an unauthorized manner, or who are logged in to multiple sessions concurrently. Abuse control can be managed manually or triggered by intrusion detection devices or network firewalls.
- Subscriber self-service—Subscribers can be redirected to portals where troubleshooting tips can be conveyed, decreasing the burden on your help desk team and providing faster resolution to common problems and concerns.

NOTE: For the remainder of this discussion, the functionality of CoA, DM, and PoD is referred to as CoA/DM.

How Disconnect Messages Work

When you want to terminate a user session, you or your Operations Support System (OSS) need to trigger Steel-Belted Radius Carrier to send a Disconnect-Request message to the NAS on UDP port 3799. The Disconnect-Request message identifies the NAS and the user session to be terminated. If the Disconnect-Request message correctly identifies a user session being maintained by the NAS, the NAS disconnects the user session and sends a confirmation message (Disconnect-ACK) back to Steel-Belted Radius Carrier. For a detailed example of a Disconnect-Request message sequence, see [Figure 250 on page 707](#).

How Change of Authorization Messages Work

When you want to change session authorizations, such as data filters associated with a user session, you or your Operations Support System (OSS) need to trigger Steel-Belted Radius Carrier to send a CoA-Request message to the NAS on UDP port 3799. The CoA-Request message may indicate the name of a data filter list to be applied for the session. If the NAS is able to change the authorizations for the user session, the NAS returns a confirmation message (CoA-ACK) to Steel-Belted Radius Carrier. If the request is unsuccessful, the NAS sends back a failure message (CoA-NAK).

How Steel-Belted Radius Carrier Processes CoA/DM Messages

To process CoA/DM requests, you need to:

- Identify the session you want to modify, delete, or disconnect
- Format the CoA/DM request with the correct attributes and send the request to the NAS

Current Sessions Table

Steel-Belted Radius Carrier dynamically maintains a list of all active sessions in the Current Sessions Table (CST). The CST contains one record (row) for each session created or updated by the server. Each record contains fields that uniquely identify the session. These fields generally include identification information such as the NAS device (NAS-Port-Type), port number (NAS-Port), or session identifier (Acct-Session-Id).

The information stored in the CST is customizable. To ensure that you can identify sessions accurately and format CoA/DM requests with the appropriate attributes for the services you are supporting in your network, we recommend that you customize the information stored in the CST for your particular network environment. For example, before you retrieve a value from the CST for use in an attribute in a CoA message, you must first map the CST field to a RADIUS attribute in the `dbc_mapping.xml` file. In the following example, the **FunkOuterUserName** field is mapped to the **Original-User-Name** attribute using the **attribute** element to enable retrieving the value in the **FunkOuterUserName** field as a result of a

session control request. The **queryAttribute** child element is used for indexing the CST while querying sessions using the **SessionControl.sh** script or Web GUI. If you want to query sessions using the attribute, then you need to specify the attribute in the **queryAttribute** child element.

```
<attributeMapping field="FunkOuterUserName" attribute="Original-User-Name">
  <queryAttribute name="Original-User-Name"/>
</attributeMapping>
```

This guide assumes that your site's CST is configured properly, and that it contains all relevant key information required to identify sessions and issue CoA/DM requests properly. For details about customizing the CST, see the *SBR Carrier Installation Guide*.

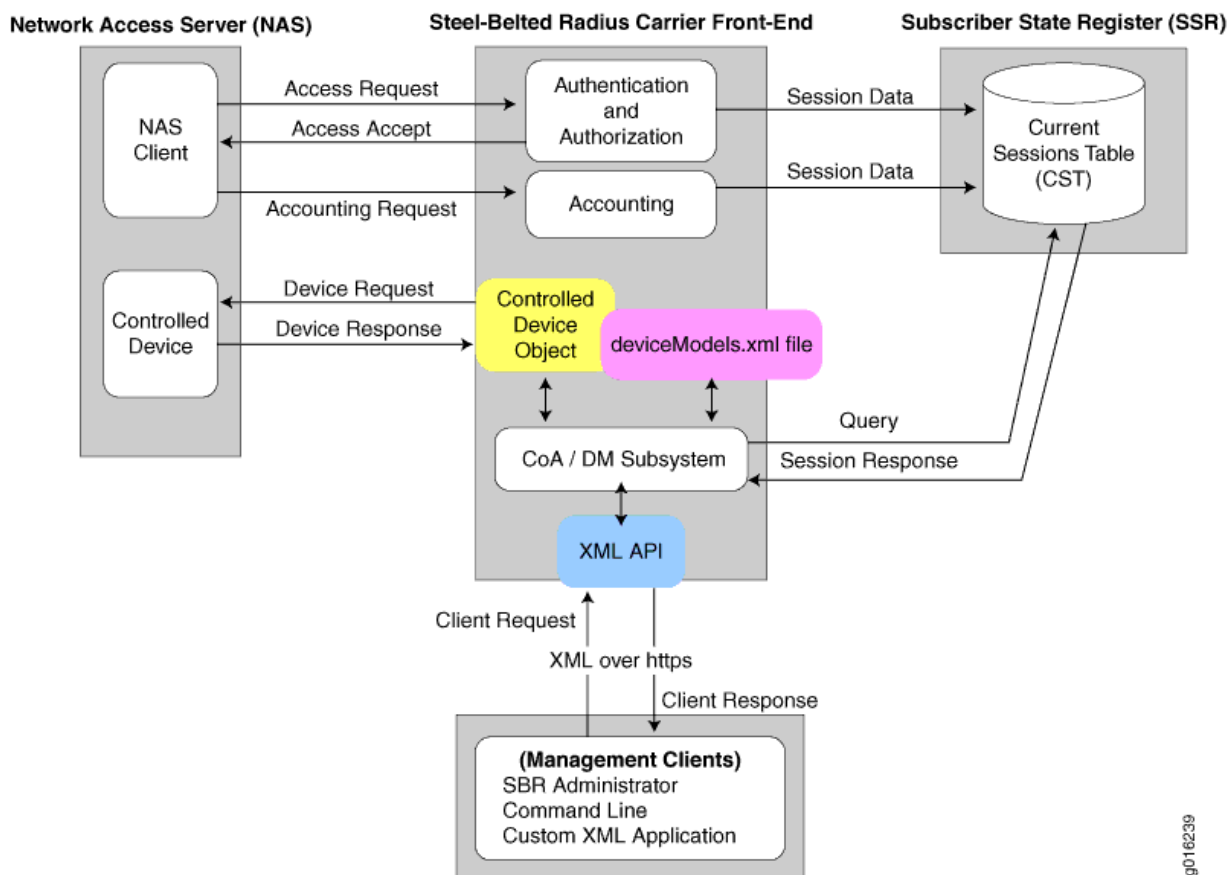
Formatting and Sending CoA/DM Requests with the Correct Attributes

Each NAS sending Access-Request messages to SBR Carrier must be defined as a client in SBR Carrier. When configuring client devices in SBR Carrier, you must define the *model* of the client device. Device model selections are defined by specifying the vendor-product parameter in the [Vendor-Product Identification] Section of the **vendor.ini** file. For each product listed in the **vendor.ini** file, SBR Carrier provides **.dct** (text) and **.dic** (xml) dictionary files. The dictionary files define the attributes SBR Carrier requires in messages it receives from a specific type of device, and the attributes to include in responses to that device.

To support CoA/DM, you need to configure a similar file that defines what attributes are included in CoA/DM requests and responses. The **deviceModels.xml** file defines the CoA/DM *actions* supported by the client devices in your network. CoA/DM actions are different *packing lists* of attribute value pairs (AVPs) that cause a particular effect, such as disconnecting a user (DM request) or reconfiguring a user bandwidth (CoA request). The **deviceModels.xml** file comes preconfigured with CoA/DM action templates for certain devices. However, you need to define CoA/DM actions in the **deviceModels.xml** file for each client device in your network that you want to support CoA/DM requests. The supported CoA/DM actions may vary depending on the NAS type.

When you configure the model for the client device (network access server), the CoA/DM subsystem in Steel-Belted Radius Carrier searches the **deviceModels.xml** file for a corresponding *controlledDeviceModel* entry matching the **Vendor-Product** (defined in the **vendor.ini** file). If a matching definition is found, a *controlled device object* is created and configured, with the CoA/DM actions defined in the **deviceModels.xml** file. If no matching *controlledDeviceModel* entry is found, then no *controlled device object* is created, and no CoA/DM actions are supported for that client device. [Figure 249 on page 704](#) shows the structural overview of the CoA/DM process within Steel-Belted Radius Carrier.

Figure 249: CoA/DM Structural Overview



CoA/DM requests can be issued to Steel-Belted Radius Carrier from several management clients including Web GUI and command line utility, or using a custom developed management application designed to connect with the Steel-Belted Radius Carrier XML interface. All CoA/DM client requests must be issued in the proper XML format as specified by this API. Refer to [“XML over HTTPS Interface”](#) on page 760 for complete details of the API.

For each targeted session you want to dynamically change, the relevant NAS (*controlled device object*) is determined. Based on the device model information defined for the client device, the required *packing list* of attributes for the current CoA/DM request type is calculated. The packing list may vary based on the CoA/DM action requested, and the make/model of the device. Packing lists are configured in XML files similar to SBR Carrier dictionary files.

NOTE: A controlled device object is associated with a controlled device (network access device), only when the device supports RFC 3576 Change of Authorization (CoA), Disconnect Message (DM), or the Cisco proprietary Packet of Disconnect (PoD).

Converting .dct Files to .dic Files

The **.dic** dictionary files are the XML format of the **.dct** dictionary files. The **.dic** dictionary files identify the attributes SBR Carrier expects when receiving Diameter requests from a specific type of device or while sending a Diameter or COA/DM requests to a specific type of device. You can convert the customized **.dct** files to **.dic** files using the **dct_to_dic_converter.sh** script.

While converting **.dct** files to **.dic** files, make sure that all attribute values are defined immediately after the specific attribute definition in the **.dct** file. For example:

ATTRIBUTE	Annex-System-Disc-Reason	Bay-VSA(44, integer)	
VALUE	Annex-System-Disc-Reason	Unknown	0
VALUE	Annex-System-Disc-Reason	Line-disconnected	1
VALUE	Annex-System-Disc-Reason	Dial-failed	2

For more information about the **.dic** dictionary files, see *SBR Carrier Reference Guide*. To convert **.dct** files to **.dic** files:

1. Execute the **dct_to_dic_converter.sh** script.

```
./dct_to_dic_converter.sh
```

2. Enter the name of the **.dct** file to be converted to the **.dic** format.

```
Enter the name of the .dct file which has to be converted to .dic format:
```

3. Enter a name for the converted **.dic** file. By default, the **.dic** file is saved in the current directory. If you want to save the **.dic** file in a different directory, you can add the directory path along with the filename—for example, **/tmp/radius.dic**.

```
Enter the filename to be used for naming the converted .dic file:
```

Controlled Devices and Actions

The *controlled device object* can support any number of actions such as disconnect. You must define each action you want to support in the **deviceModels.xml** file. We recommend you name each supported action generically so that different device models can all use the same name for the equivalent action. An action request includes some number of attributes, which can be key attributes used to find the session, or action attributes needed to instruct the device what to do.

For example, you can define the action *disconnect* with different attribute packing lists and different message types for various types of client devices. A Cisco Systems device might use a PoD message containing the attribute **User-Name** for the disconnect action. A Juniper Networks ERX device might use an RFC 3576 DM-request containing the attribute **Acct-Session-Id** for the disconnect action. However, by configuring the **deviceModels.xml** file with an entry for both the Cisco Systems device and the Juniper

Networks ERX device, and defining an action called *disconnect* for each of these devices, you can simply invoke the action *disconnect* with the argument **User-Name='bob'**, and the user session for bob is terminated regardless of whether it is found on the Juniper Networks device or Cisco Systems device.

In this example, the Juniper Networks ERX device requires the Acct-Session-Id for the DM-Request. The Acct-Session-Id attribute is looked up in the Current Session Table (CST). Therefore, we recommend you also configure the CST so that the required attributes are available from the session query to satisfy the packing list of each action supported by the device.

NOTE: The Acct-Session-Id attribute is commonly used in CoA/DM packing lists for router requirements. Packing lists represent the router's requirements for a valid DM or CoA request. Attributes that are not in the CST must come from the actual query to fulfill the request. To have data available for queries, you must configure this attribute in the packing list. If the AcctSessionId does not exist in the CST or is not found by the query itself, the request fails.

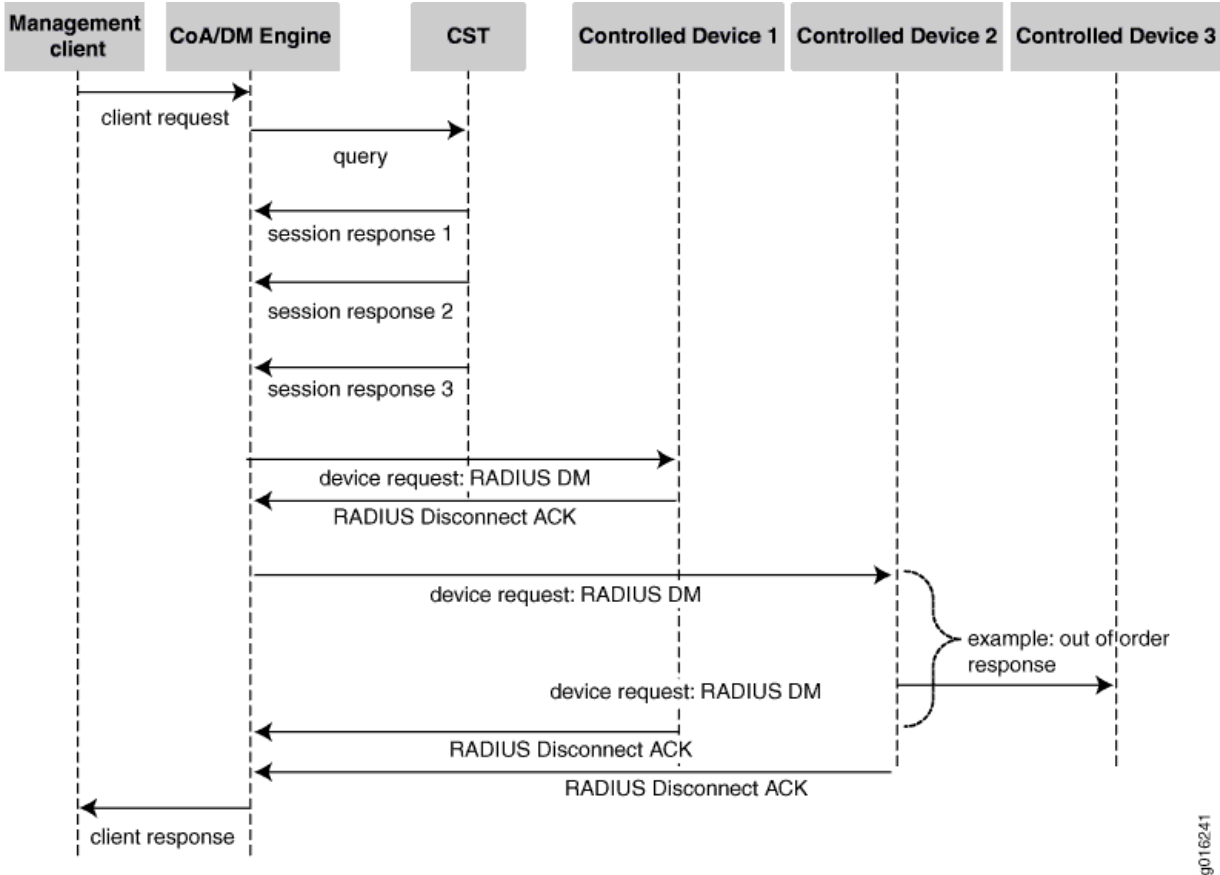


CAUTION: Only edit the attribute packing lists (actions) in the **deviceModels.xml** file with the assistance of Juniper Networks Sales Engineering or Professional Services.

Sequence and Flow of CoA/DM Requests Through Steel-Belted Radius Carrier

Figure 250 on page 707 shows the sequence diagram of requests and responses that occur between the initial client request and final client response for a disconnect message.

Figure 250: Example Sequence of a Disconnect Message (DM) Action



The XML client request is sent to the CoA/DM subsystem from the management client. The CST is queried for some number of sessions. A session is selected and a RADIUS DM-Request is sent to the various controlled devices (NAS). Finally, a response is sent back to the management client.

Figure 251 on page 708 shows how multiple device request/responses are created for one disconnect message.

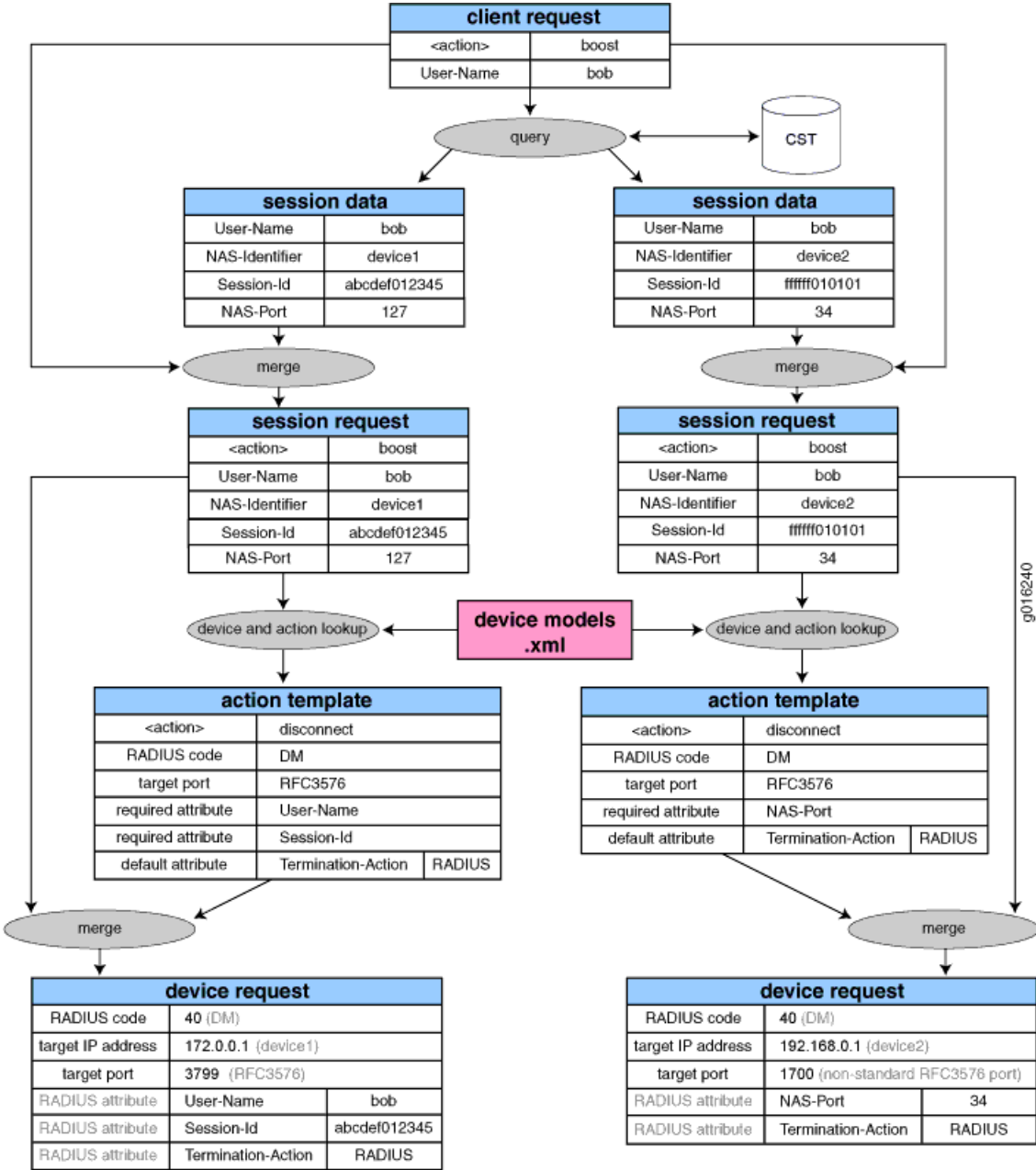
NOTE: In Figure 251 on page 708, and in the following discussion the terms *client request*, *session data*, *session request* and *device request*, are all XML elements in the Steel-Belted Radius Carrier XML interface and **deviceModels.xml** file.

The *action templates* shown in Figure 251 on page 708 equate to the `<action>` element in the **deviceModels.xml** file, and the `<deviceRequestSpec>` element in the API.

The example shown in Figure 251 on page 708 shows a *client request* that includes an action called *disconnect* for User-Name='bob'. The query of the CST finds two active sessions for User-Name='bob' being handled

by two different NAS: Session-Id=abcdef012345 is being handled by a NAS with NAS-Identifier=device1, and Session-Id=fffff010101 is being handled by a NAS with NAS-Identifier=device2.

Figure 251: Example Data Flow for a Disconnect-Request



Data from the *client request* is merged with the data from each active session found in the CST (*session data*), to form the two *session requests*. The data gathered from the device and action lookup in the **deviceModels.xml** file is combined with the existing *session request* data to form the *action template*. The action template specifies the attribute packing list for both device requests. The action template is merged with the *session request* to form the final request (*device request*) sent to each NAS.

If not all attributes in the packing list can be populated with the available information, the device request fails and Steel-Belted Radius Carrier returns a detailed response indicating the missing information to the client.

If the packing list is successfully satisfied, the specified RADIUS packet (*device request*) is sent to the device and Steel-Belted Radius Carrier waits for a successful response, failure response, or timeout, before returning an XML result report to the client.

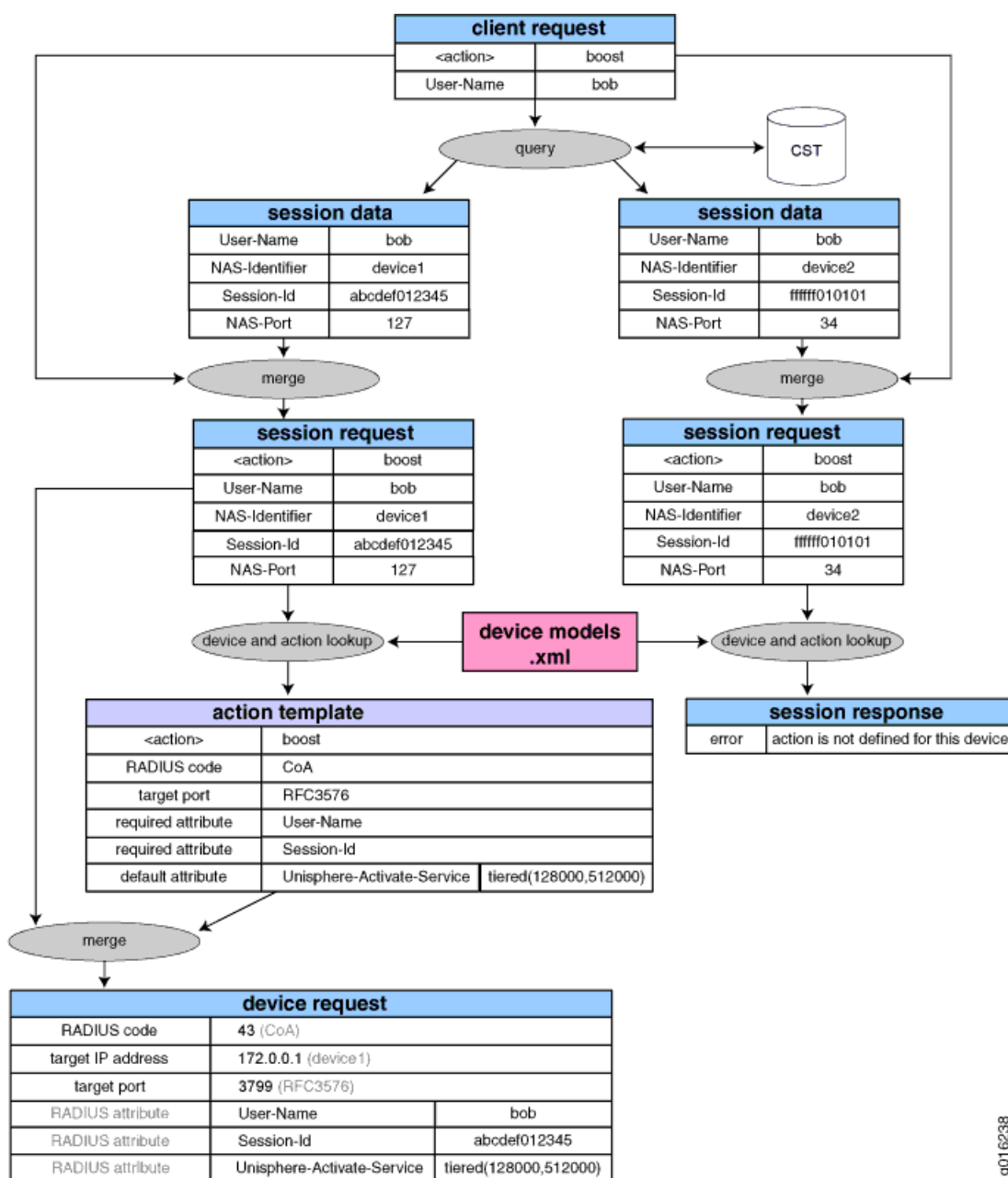
Defining the same action for all NAS in your network simplifies what you need to specify in client requests. In the example shown in [Figure 251 on page 708](#), because the action *disconnect* is defined in the **deviceModels.xml** file for both NAS (device1 and device2), you simply need to invoke the action *disconnect* in the client request, with the arguments User-Name='bob', and all user sessions for bob are terminated.

[Figure 252 on page 710](#) shows the data flow for an example CoA request.

NOTE: In [Figure 252 on page 710](#), and in the subsequent discussion the terms *client request*, *session data*, *session request* and *device request*, are all XML elements in the Steel-Belted Radius Carrier XML interface and **deviceModels.xml** file.

The *action templates* shown in [Figure 252 on page 710](#) equate to the <action> element in the **deviceModels.xml** file, and the <deviceRequestSpec> element in the API.

Figure 252: Example Data Flow for a CoA-Request



The example shown in [Figure 252 on page 710](#) includes a *client request* with an action called *boost* for User-Name='bob'. You can name actions anything you want. In this example, the name *boost* is meant to represent boosting the bandwidth of a session. The query of the CST finds two active sessions for User-Name='bob' being handled by two different NAS: Session-Id=abcdef012345 is being handled by a NAS with NAS-Identifier=device1, and Session-Id=fffff010101 is being handled by a NAS with NAS-Identifier=device2.

Data from the *client request* is merged with the data from each active session found in the CST (*session data*) to form the two *session requests*. The device and action lookup in the **deviceModels.xml** file determines

that the action called *boost* is defined for the NAS with NAS-Identifier=device1, but is not defined for the NAS with NAS-Identifier=device2. So, in this example, the action called *boost* can only be performed on Session-Id=abcdef012345 running on NAS-Identifier=device1. The data gathered from the lookup in the **deviceModels.xml** file for NAS with NAS-Identifier=device1 is combined with the existing *session request* data to form the *action template*. The action template specifies the attribute packing list for the device request. The action template is merged with the *session request* to form the final *device request*.

If not all attributes in the packing list can be populated with the available information, the device request fails and Steel-Belted Radius Carrier returns a detailed response indicating the missing information to the client.

If the packing list is successfully satisfied, the specified RADIUS packet (*device request*) is sent to the device and Steel-Belted Radius Carrier waits for a successful response, failure response, or timeout, before returning an XML result report to the client.

Defining the same action for all NAS in your network simplifies what you need to specify in client requests. In the example shown in [Figure 251 on page 708](#), because the action *disconnect* is defined in the **deviceModels.xml** file for both NAS (device1 and device2), you simply need to invoke the action *disconnect* in the client request, with the arguments User-Name='bob', and all user sessions for bob are terminated.

NOTE: When configuring actions in the **deviceModels.xml** file, we recommend you name each action something generic so that different device models can use the same name for an equivalent action.

Implementing CoA/DM Support

Step 1: Develop a Deployment Plan

As a mobile or wire-line operator, you can configure the CoA/DM functionality to offer dynamic service changes to reinforce your current service offerings. Before you begin to customize the CoA/DM settings, develop a comprehensive deployment plan based on the solution your company offers to its customers. Have your network architect or someone who has detailed knowledge of your network access servers and service offerings develop this plan.

Depending on the complexity of the services you want to provide, you may need to customize other aspects of your network. For example, if you want to offer a prepaid service, then include in the deployment plan all of the different types of disconnect messages and change of authorization messages that are required to support control of data services where usage is metered by time or traffic volume. When users exhausts their prepaid service quota, the CoA/DM feature can disconnect the users, or redirect them to

a subscriber page to ensure that subscribers do not exceed their purchased limit and allow them to purchase more time or data in mid-session.

NOTE: This type of service interacts with other devices in your network that may require customization beyond the scope of this guide. Make sure your deployment plans take into consideration all aspects of the services you want to provide.

Step 2: Consult Your NAS-Specific Documentation

The information required to identify a session and to process CoA/DM requests depends on the NAS devices in your network. To determine a NAS's capabilities, you must consult the documentation for each NAS, find its make and model, and determine the appropriate attribute packing list (containing the list of attribute-value pairs). What actions is the NAS capable of supporting? What attributes does the NAS require in the request? What attributes does the NAS include in the response? You need to know what type of information the NAS supports because if the list of AVPs cannot satisfy a request, then the request cannot be sent.

You must configure the **deviceModels.xml** file to support your specific NAS according the NAS capabilities, and the specific actions you want to support.

Step 3: Configure Each NAS as a Client in Steel-Belted Radius Carrier

Each device in your network sending requests to the server must be defined as a client in Steel-Belted Radius Carrier. To support RFC 3576 Change of Authorization (CoA), Disconnect Message (DM), or the Cisco proprietary Packet of Disconnect (PoD), you need to configure the following parameters for the client:

- **RFC 3576 CoA/DM port and RFC 3576 CoA/DM Shared secret**
- **POD port and POD Shared secret**

NOTE: For CoA/DM or PoD ports, you must specify the shared secret for the CoA/DM functionality to work. If you do not specify port numbers, Steel-Belted Radius Carrier uses the defaults from the **deviceModels.xml** file.

NOTE: If a NAS client is configured without saving the shared secret, you are prompted to enter the shared secret when the client is subsequently viewed. If unexpected results such as invalid signatures occur, ensure that the shared secret is set correctly.

NOTE: CoA/DM and POD messages do not work for the <ANY> RADIUS client.

For complete details on configuring clients in Steel-Belted Radius Carrier, see [“Administering RADIUS Clients and Client Groups” on page 104.](#)

Step 4: Configure the deviceModels.xml File

The **deviceModels.xml** file contains a list of device models for each *controlled device object* associated with your NAS clients, and defines the *actions* supported by each NAS. The actions supported by each NAS can vary. For example, some devices may use different AVPs as keys when referring to a session, such as Acct-Session-Id, NAS-Port, and NAS-Port-Type. As a result, you need to customize the **deviceModels.xml** file to support the specific NAS and their associated CoA/DM capabilities.

For information about configuring the **deviceModels.xml** file, see [“Configuring the deviceModels.xml File” on page 743.](#)

Step 5: Configure the Current Sessions Table (CST) for Your Environment

To ensure that CoA/DM requests get processed properly, you need to customize the current sessions table (CST) for your network environment. The CST must be customized to include the attributes in session queries that your NAS devices require in a CoA/DM (action) request, and for what SBR Carrier requires in the NAS response. Customizing the CST to include the proper attributes ensures that the *session data*, returned from a Query, includes the appropriate attributes your NAS device requires to process CoA/DM requests.

If you do not customize the CST, then you need to ensure that each DM-Request or CoA-Request includes the appropriate attribute packing list for the request.

For details on customizing the CST, see the *SBR Carrier Installation Guide*.

Processing Dynamic Authorization (CoA/DM) Messages as a Proxy Server

When you configure SBRC as a proxy server, all authentication and accounting requests are forwarded from an upstream device (usually a NAS client) to a downstream device (the proxy target). In the case of CoA/DM functionality, the unsolicited CoA/DM messages are routed from the proxy target to the NAS client that created the session. SBRC as a proxy server listens for unsolicited CoA/DM messages on a configurable UDP port. In addition to listening on a configurable UDP port, SBRC creates the associated threads to handle these requests.

SBRC processes the proxy CoA/DM requests in a similar way that it handles the proxy authentication or accounting requests. SBRC checks the packet for duplicate requests, verifies the shared secret of the sender (which needs to match the configured proxy target), verifies the attributes, and determines the upstream device based on the NAS-Identifier/NAS-IP-Address and Acct-Session-Id attributes in the request, and then the proxy request is sent to the upstream device. When a response to a proxied CoA/DM request is received, the response is matched with an outstanding proxy request and forwarded to the originating device.

Attributes and CoA/DM Forwarding Methods

Attribute filtering and script support is a feature that can be configured to apply to forwarded CoA/DM requests. To enable forwarding of incoming CoA/DM requests to the upstream device that originated the session, a new field in the Current Sessions Table (CST), Sbr_NasClientName, associates a session record with the name of the client that created the session. By default, the Sbr_NasClientName field is disabled, and to enable it you need to add it in the CST. The attribute, Funk-NAS-Identifier, is used by SBRC to refer to the Sbr_NasClientName field.

There are three methods that SBRC uses to forward a received proxy CoA/DM request.

- Method 1—SBRC checks the Current Sessions Table (CST) for a session matching the received attributes (either NAS-Identifier or NAS-IP-Address plus Acct-Session-Id) and forwards the request to the NAS named in the Sbr_NasClientName field. This is the default setting.
- Method 2—SBRC forwards the request directly to the configured client that matches either the NAS-IP-Address or NAS-Identifier attribute in the received proxy CoA/DM request.
- Method 3—This is a combination of methods 1 and 2. If a matching session is not found in the CST (or if there are duplicate matches), an attempt is made to forward the request by matching the attributes with the configured clients.

In all these three methods, if an appropriate NAS target is not found, a CoA/DM NAK response is sent as a reply.

If Reverse Path Forwarding check is enabled, SBRC verifies that the sender of the CoA/DM request is in the proxy realm associated with a session. If Reverse Path Forwarding check is enabled and fails, then a CoA/DM NAK response is sent.

You notice functionality differences if a CoA/DM request is forwarded using the CST, or if it is forwarded directly to a NAS. When a request is forwarded directly, the Reverse Path Forwarding check is disabled because there is no session associated with the request and there is no mechanism to check which proxy realm is associated with a particular request. When a request is forwarded directly and there is no realm associated with it, the realm configuration to use for the attribute filter and CheckMessageAuthenticator settings is a proxy realm the proxy target belongs to. If the proxy target belongs to more than one realm, the realm that is randomly selected remains the same until the proxy realm configuration is changed. If the proxy target does not belong to a proxy realm, a NAK response with an Error-Cause attribute of 505 (Other Proxy Processing Error) is sent in response to a CoA/DM request.

Processing Dynamic Authorization (CoA/DM) Messages as a Proxy Target

SBR Carrier keeps track of the device (either a proxy server or a NAS client) that sent the packet. This ensures that CoA/DM requests are sent to the originating NAS client using the same path (in reverse direction) that the authorization and accounting requests traversed. When a CoA/DM request is generated, the message is sent to the device.

To help the proxy target determine which device model is used for a given session, a new attribute, Funk-Device-Model, is added to forward authentication and accounting requests. The Funk-Device-Model attribute is a string attribute and contains the make or model name of the NAS client associated with a request. The Funk-Device-Model is useful only when the proxy target is also a SBR Carrier server. If a Funk-Device-Model attribute is received as part of an authorization or accounting request, the attribute details are saved in the Sbr_NasDeviceModel field of the CST and passed along without any modification to the proxy target. By default, the Sbr_NasDeviceModel field is disabled, and to enable it you need to add it in the CST. This feature enables SBR Carrier servers acting as proxy targets to determine which attributes to use to send a CoA/DM request through a proxy without having to configure a RADIUS client for every possible NAS client on the network. However, there are configurations in which this information is not required to generate the correct list of attributes, so the Funk-Device-Model attribute is optional, and can be disabled through a configuration variable.

When a CoA/DM request is created, the attributes included in the message are determined by the device model of the originating upstream device (NAS client). The Funk-Device-Model attribute, if present in the proxy authentication and accounting requests, is used to determine the device model.

NOTE: The Funk-Device-Model attribute does not determine the port to send the request to. The port is determined in advance for each NAS client.

If the Funk-Device-Model attribute is not implemented, then you need to create a device model that is a superset of all the possible devices on the network that receives CoA/DM requests. This generic superset device model needs to be configured for any client that receives CoA/DM requests.

Settings to Support the Proxy CoA/DM Functionality

This section explains the settings and the parameters to support the Proxy CoA/DM functionality.

The [DynAuthProxy] section in the **radius.ini** file controls some global Proxy CoA/DM features. For information about [DynAuthProxy] section parameters of the **radius.ini** file, see the *SBR Carrier Reference Guide*.

NOTE: The Proxy CoA/DM functionality is disabled if the Session Control feature license is not set.

In the **radius.ini** file, the **UDPDynAuthPort** parameter in the [Ports] section indicates the ports that SBR Carrier listens on for Proxy CoA/DM messages. This defaults to port 3799. For information about [Ports] section parameters of the **radius.ini** file, see the *SBR Carrier Reference Guide*.

In the [Configuration] section of the **radius.ini** file, a complete set of dynamic authentication thread and flood queue configuration for dynamic authentication works in the same way as the authorization, accounting, and proxy thread and flood queue configuration. The variables added to the [Configuration] section are Dynamic-Auth-Receive-Realtime-Thread-Priority, Max-Dyn-Auth-Threads, Max-Dyn-Auth-Floods, Max-Dyn-Auth-Threads-In-Flood, and Dyn-Auth-Flood-Queue-Shape. Another parameter in the [Configuration] section, DynAuthProxySource, configures the source interface address used to forward Proxy CoA/DM requests, similar to the ProxySource setting. The default setting for this value is 0.0.0.0 (IPv4 unspecified). The thread pool reports in the server log include reports for dynamic authorization threads.

You can change the proxy target configuration to include optional configuration of a CoA/DM secret if it is different from the authorization shared secret.

NOTE: The primary dictionary is used when parsing incoming CoA/DM requests.

The [DynAuth] section in each **realm.pro** file controls the proxy CoA/DM configuration for each proxy realm. The FilterOut parameter in this section, if set to a valid attribute filter name, causes SBR Carrier to apply that filter (and optional JavaScript) when forwarding a CoA/DM request to a NAS client. The **IncludeDeviceModel** parameter can be set to “yes” or “no.” If set to “yes,” then the Funk-Device-Model attribute with the appropriate device name is added to every forwarded proxy request sent to this realm. The RequireMessageAuthenticator setting defaults to 0, or no check. If set to a non-zero value, SBR Carrier requires a Message-Authenticator attribute in incoming CoA/DM requests, and the request is discarded if this attribute is not present.

To prevent performance degradation, if this feature is enabled, the CST fields (Sbr_NasClientName, Sbr_NasDeviceModel, Sbr_ProxyRealm, and Sbr_ProxyState) need to be configured. This can be done by modifying the **CurrentSessions.sql** file for the cluster version, or the **IncludeDynAuth** parameter in the [Configuration] section of **dbclusterlocal.gen** for the standalone version. The **CurrentSessions.sql** file has the fields for Proxy CoA/DM included but commented out, to make it easier to configure these fields in the cluster version.

ShowSessions.sh does not display the Sbr_NasClientName, Sbr_NasDeviceModel, Sbr_ProxyRealm, and Sbr_ProxyState fields, unless you modify the script to do so.

You need to modify the **dbc_mapping.xml** file to add the new Funk-NAS-Identifier, Funk-Device-Model, and Funk-Proxy-State fields to query results. A new version of **dbc_mapping.xml** includes these new fields in the lines that are commented out.

The DynAuth-Thread-Flood-Info parameter in the [Status] section of the **radius.ini** file controls whether the DynAuth thread and flood information is printed in the status log during status reports. It defaults to "no."

For information about configuring the **radius.ini** file to support the CoA/DM functionality, see the *SBR Carrier Reference Guide*.

Using Web GUI to Manage and Control Sessions

IN THIS CHAPTER

- [Current Sessions Overview | 718](#)
- [Searching for Sessions Using Web GUI | 719](#)
- [Setting Session Limits with Web GUI | 724](#)
- [Executing CoA and Disconnect Requests Using Web GUI | 725](#)

This chapter describes how to manage sessions in SBR Carrier using the Web GUI. The following topics are included:

Current Sessions Overview

SBR Carrier tracks the status of user connections that it authenticates in a current sessions table (CST). Because the CST is based on RADIUS accounting data, the list of active sessions is accurate only if all of your NAD are configured to support RADIUS accounting. To accurately track session activity, you need to ensure:

- All clients in your configuration support RADIUS accounting
- All clients are configured to send accounting messages to SBR Carrier

You use the Web GUI to search for all sessions, or specify certain criteria and display only sessions that match that criteria. The Web GUI searches the current session table (CST) to find the sessions.

The **Current Sessions** page is used for managing and controlling sessions. With the basic SBR Carrier license, you can perform the query, view, and delete functions. The **Query** button displays all sessions that meet the criteria defined by the Session Query parameters. The **View** button displays the details for the selected session. The **Delete** button deletes the selected session from the SBR Carrier current sessions table.

NOTE: For tunnel connections, if SBR Carrier was used to authenticate both the user and the tunnel, then two entries are displayed in the **Current Sessions** page: one entry for the authenticated user, and one for the authenticated tunnel.

NOTE: Deleting a session does not disconnect the session, it only deletes the session from the CST. To disconnect a session, see [“Executing CoA and Disconnect Requests Using Web GUI” on page 725](#).

If you are using the optional Session Control module license, you can also issue CoA/DM actions from the **Current Sessions** page. By default, the special actions of Query, View and Delete are predefined in the **deviceModels.xml** file.

When you define an action in the **deviceModels.xml** file, it appears as a button in the **Current Sessions** page. In the example discussed in [Figure 252 on page 710](#), the action *boost* is defined as an action in the **deviceModels.xml** file. If this action is defined in your **deviceModels.xml** file, a button labeled “boost” would be displayed in the **Current Sessions** page.

The Web GUI also contains buttons for the actions Disconnect, InterceptOn and InterceptOff. These buttons reflect CoA actions defined in the **deviceModels.xml** file. The **Disconnect** option or button disconnects one or more selected sessions. The **InterceptOn** and **InterceptOff** options or buttons enable you to comply with Law Enforcement Agency (LEA) requests for lawful intercept. To start the lawful intercept using the Web GUI, select a session and then select **Device Action > InterceptOn**. To stop the lawful intercept using the Web GUI, select the same session and then select **Device Action > InterceptOff**.

NOTE: To support the Disconnect and InterceptOn and InterceptOff actions on a NAD in your network, you must define the make and model of each NAD in the **deviceModels.xml** file. If these are not defined, invoking the action fails.

Searching for Sessions Using Web GUI

Using Web GUI, you can search for all sessions, or specify certain criteria and display only sessions that match that criteria. Web GUI searches the current session table (CST) to find the matching sessions.

NOTE: Sessions must be running on a controlled device and that device must have sent an accounting start message in order to be found in the CST.

You can include wildcards in your search criteria to find sessions that match the leading or ending characters you specify. For example, enter AB* in the **Select by User** field to find all sessions for which the username begins with AB. Wildcards are only accepted for the **Select by User** and **Select by NAS Name** fields (text fields).

Leading wildcards are supported only in a trailing position. For example, "*bcd" matches any value and "abc*def" matches any value beginning with "abc".

TIP: To find all sessions, enter an * in the **Select by User** field with nothing entered for other fields.

Session Query Fields and Searchable Attributes

Table 104 on page 720 describes the fields that appear in the **Current Sessions** page. You can use these fields as criteria when searching for sessions. For example, you can search for all sessions being run by a user with the username you specify.

Table 104: Session Query Parameters

Field	RADIUS Attribute	Description
Limit Sessions Returned to	Not applicable	Type the upper limit of number of sessions to be found. The result is the lesser of this number or the system session limit. See “Setting Session Limits with Web GUI” on page 724 for more information about the effect of the Limit Sessions returned to field.
User-Name	User-Name	RADIUS username.
NAS Name	NAS-Identifier	Identifier set by the carrier when the device is deployed.
NAS IP	NAS-IP-Address	IP address of the controlled device.

Table 104: Session Query Parameters *(continued)*

Field	RADIUS Attribute	Description
Assigned IP	Framed-IP-Address	IPv4 address assigned to the user.
Framed-IPv6-Prefix	Framed-IPv6-Prefix	IPv6 prefix assigned to the user.
Framed-IPv6-Address	Framed-IPv6-Address	IPv6 address assigned to the user.
Session ID	Acct-Session-Id	Unique identifier assigned to a session.
Multi session ID	Acct-Multisession-Id	Identifier assigned to multiple sessions to link them together.
Calling Station	Calling-Station-ID	Station identifier (usually a telephone number or MAC address) of the user.
Called Station	Called-Station-ID	Station identifier of the NAD (the telephone number dialed by the user or the MAC address of the NAD)

To find sessions using the Web GUI:

1. Select **RADIUS Configuration > Current Sessions**.

The **Current Sessions** page (Figure 253 on page 722) appears.

For more information about the parameters in the **Current Sessions** page, see Table 104 on page 720.

Figure 253: Current Sessions Page

User-Name	NAS Name	NAS IP	Assigned IP	Framed-IPv6-Prefix	Framed-IPv6-Address	Session ID	Multisession ID	Calling Station	Called Station
TEST	NAS1	1.2.3....	1.2.3.4		{hex}2008dbcaabcd000000000000000000000001	123456789		00381123456	mms
TEST	NAS1	1.2.3....	1.2.3.4		{hex}2008dbcaabcd000000000000000000000003	123456789		00381123456	mms

2. Enter a value in the **Limit Sessions Returned to** field if you want to limit the number of sessions found.

For more information about the effect of the **Limit Sessions Returned to** field, see “[Setting Session Limits with Web GUI](#)” on page 724.

3. Move your cursor on the parameter column header you want to use in the search and click the **Down** arrow. Then, move your cursor over the **Filters** check box and enter the searching criteria in the text box.

NOTE: If you select or enter multiple criteria, the search finds sessions that meet all the criteria (Boolean AND).

4. Click **Query**.

The **Current Sessions** page (Figure 253 on page 722) displays the sessions that match the search criteria.

Viewing Session Detail

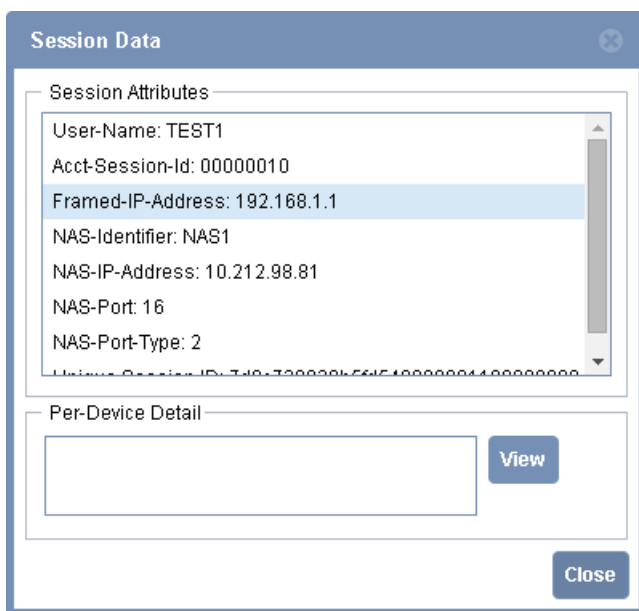
To view session details using the Web GUI:

1. In the **Current Sessions** page (Figure 253 on page 722), select the session for which you want to view the details and click **View**.

The **Session Data** dialog box (Figure 254 on page 723) displays additional RADIUS attributes about the session.

The **Session Data** dialog box has a different appearance for different device makes and models. The attributes that appear in the dialog box vary depending on the make and model of the controlled device. Different devices send different lists of attributes and values.

Figure 254: Session Data Dialog



The screenshot shows a dialog box titled "Session Data". It contains two main sections: "Session Attributes" and "Per-Device Detail".

Session Attributes: This section lists several attributes with their values:

- User-Name: TEST1
- Acct-Session-Id: 00000010
- Framed-IP-Address: 192.168.1.1 (This line is highlighted with a blue background)
- NAS-Identifier: NAS1
- NAS-IP-Address: 10.212.98.81
- NAS-Port: 16
- NAS-Port-Type: 2
- Username-IP: 10.10.10.10

Per-Device Detail: This section contains a large empty text box and a "View" button.

At the bottom right of the dialog box is a "Close" button.

2. Click **Close** after viewing the session details.

Deleting Sessions

Normally, the system maintains the information in the sessions list based on accounting information received from the RADIUS client. However, a user who has logged off may still be identified as active in the sessions list if communications between the RADIUS client and SBR Carrier fail or if either the RADIUS client or SBR Carrier is taken down for a period of time.

In most cases, SBR Carrier can correct such anomalies itself. For example, if a new user dials in to the same port on the same RADIUS client, SBR Carrier infers that the previous user must have disconnected and removes the entry.

You can manually correct the sessions list by selecting any entry and clicking **Delete**. This removes the user from the list and decrements the user's connection count (if it is being tracked) by one. Any pooled IP address assigned to the deleted user is returned to the appropriate pool.

NOTE: Deleting a session deletes the session from the SBR Carrier Current Sessions Table (CST). However, the session remains active. To inactivate a session, you must perform disconnect which requires the optional Session Control module. For more information, see [“Overview of the Optional Session Control Module” on page 700](#).

Setting Session Limits with Web GUI

You can control the maximum number of sessions that are returned when you execute an action. You may want to limit the number of sessions returned if you are concerned that too many sessions may be returned.

Factors Affecting the Number of Sessions Returned

The number of sessions returned in response to a request depends on four factors:

- The number you type in the **Limit Sessions Returned to** field. If you leave the field blank, it defaults to the system-wide session limit.
- The system-wide session limit. This number defaults to 10,000 but may be changed with the assistance of your Juniper Networks Technical Support representative.
- The Effective Session Limit. This number is always the lesser of the system-wide session limit and the number you enter in the **Limit Sessions Returned to** field.
- The number of sessions found that meet the criteria you enter.

Number of Sessions Returned

The number of sessions returned by SBR Carrier is always the lesser of the Effective Session Limit and the number of sessions found that match the criteria you enter.

Table 105 on page 725 provides examples in which the session limit rules apply.

Table 105: Number of Sessions Found for Various Examples

Value Entered in the "Limit Sessions Returned to" Field	System-Wide Session Limit	Effective Session Limit	Sessions Found	Sessions Returned	Reason
100	10,000	100	200	100	Effective session limit prevails
100	10,000	100	50	50	Number of sessions found is within effective limit
10	10,000	10	200	10	Effective session limit prevails
blank	10,000	10,000	200	200	Number of sessions found is within effective limit
20,000	10,000	10,000	25,000	10,000	Effective session limit prevails

Executing CoA and Disconnect Requests Using Web GUI

NOTE: CoA/DM support is only available with the optional Session Control license.

Because the make and model of NADs differ in each network environment, the default *actions* defined in the **deviceModels.xml** file for Change of Authorization (CoA) and Disconnect Message (DM) requests are for example purposes only. To issue CoA or DM (action) requests to NADs in your network, you need to define supporting *actions* in the **deviceModels.xml** file for each NAD in your network, and specify the appropriate attribute packing list for each NAD make and model.

After an action is defined in the **deviceModels.xml** file, a button labeled with the name of the action becomes available in the **Current Sessions** page in Web GUI. To execute the action, you simply perform a query to locate and select one or more sessions, and then select the action button to execute the action.

When configuring actions in the **deviceModels.xml** file, be consistent about what you name the actions for the various makes and models of NADs in your network. Naming actions consistently enables you to simply invoke the action without regard for which NAD is handling the session you want to take action on.

Example of Executing a Disconnect Action

NOTE: The procedure in this section is an example only, and assumes you have already created the action titled Disconnect in the **deviceModels.xml** file.

After you have found a list of sessions matching the criteria you specify, you can select one or more of the sessions to disconnect. The effect of disconnecting sessions on the current session table (CST) is described in [Table 106 on page 726](#).

Table 106: Effect on CST of Disconnecting Sessions

Action	Result	Effect on Current Session Table
Request to disconnect is sent and the disconnect is acknowledged (ACK)	Controlled device sends a RADIUS Accounting-Request (Stop) to the accounting client, which, in turn, sends a Stop to the SBR Carrier.	Session is deleted from the CST
Request to disconnect is sent and the disconnect is not acknowledged (NAK)	Not applicable	Session is deleted from the CST because the session does not exist
Request to disconnect is sent and the disconnect times out	Not applicable	No effect on the CST

NOTE: The procedure in this section is an example only. The exact fields that are displayed change depending on the actions and the make and model of the NAD devices defined in your **deviceModels.xml** file.

To disconnect sessions using the Web GUI:

1. Find the sessions that meet your search criteria. (See “[Searching for Sessions Using Web GUI](#)” on [page 719](#).)
2. In the **Current Sessions** page ([Figure 253 on page 722](#)), select one or more sessions.
To select more than one session, hold down the Ctrl key and click on the sessions.
3. Select **Device Action > Disconnect**.
A **Disconnect** dialog box appears ([Figure 255 on page 727](#)).

Figure 255: Example Disconnect Dialog

Disconnect

Request Parameters

Request Execution

Disconnect Result Message:
<not executed>

Successes	Failures	Timeouts	Errors								
<table border="1"> <thead> <tr> <th>User-Name ↑</th> <th>NAS-Identif...</th> <th>Session-Id</th> <th>View</th> </tr> </thead> <tbody> <tr> <td colspan="4">No Successes</td> </tr> </tbody> </table>	User-Name ↑	NAS-Identif...	Session-Id	View	No Successes						
User-Name ↑	NAS-Identif...	Session-Id	View								
No Successes											

Close

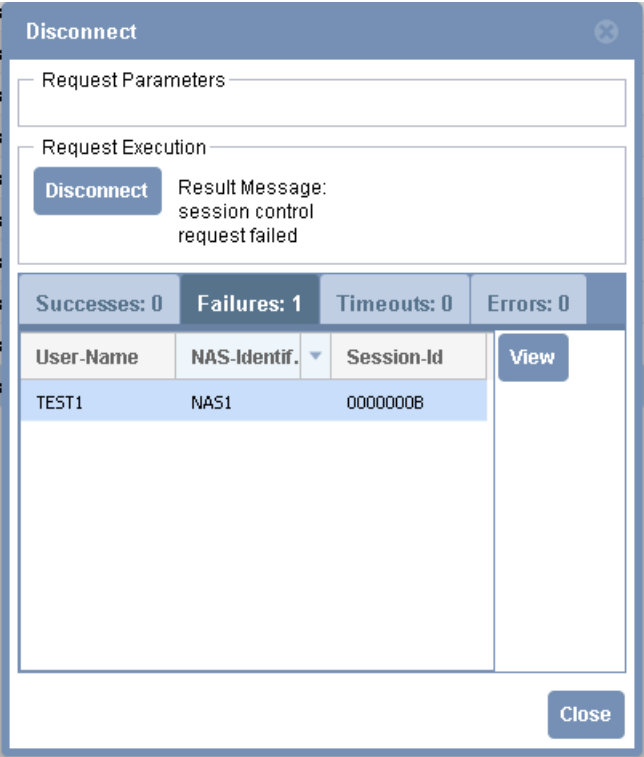
4. In the **Request Parameters** area, leave the **Acct-Session-Id** check box cleared and leave the area as **(using session data)**. The example shown in [Figure 255 on page 727](#) does not show the information for the **Request Parameters** area.

This area of the dialog box indicates that the session identifiers are taken from the sessions you selected.

5. Click **Disconnect**.

A watch icon appears while the disconnect request is processed. One or more of the result tabs becomes active (**Successes**, **Failures**, **Timeouts**, **Errors**), depending on the result of the disconnect attempt ([Figure 256 on page 728](#)).

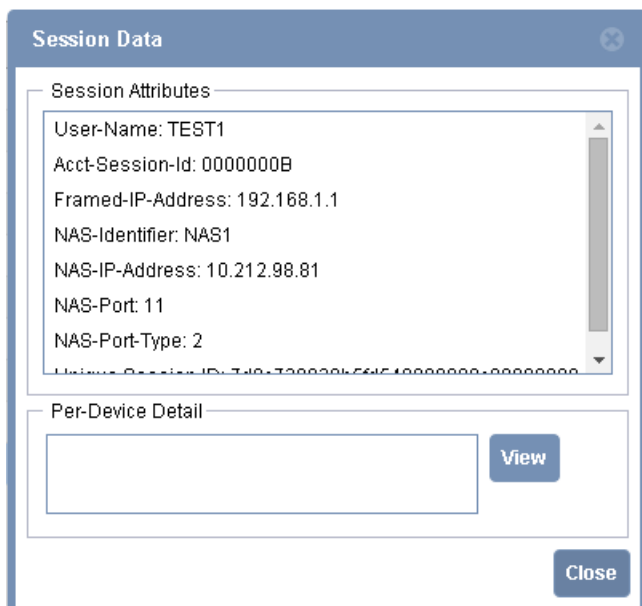
Figure 256: Example Result Dialog



6. Select an active result tab and a session, and click **View**.

A **Session Data** dialog box (Figure 257 on page 729) appears. The **Session Data** dialog box has a different appearance for different device makes and models. The attributes that appear in the **Session Data** dialog box vary depending on the make and model of the device. Different devices send different attribute packing lists and values.

Figure 257: Example Session Data Dialog



The image shows a 'Session Data' dialog box with a blue title bar and a close button. It contains two main sections: 'Session Attributes' and 'Per-Device Detail'. The 'Session Attributes' section has a list of session details with a vertical scrollbar. The 'Per-Device Detail' section has an empty text box and a 'View' button. A 'Close' button is located at the bottom right of the dialog.

Session Attributes	
User-Name:	TEST1
Acct-Session-Id:	0000000B
Framed-IP-Address:	192.168.1.1
NAS-Identifier:	NAS1
NAS-IP-Address:	10.212.98.81
NAS-Port:	11
NAS-Port-Type:	2
Unique-Session-ID:	7-18-7388381561518888888-88888888

Per-Device Detail

View

Close

7. Select the device in the **Per-Device Detail** area and click **View**.

Information from the device running the session you disconnected (or attempted to disconnect) is displayed.

8. Click **Close** for each open dialog box.

Using the Command Line Utility to Manage and Control Sessions

IN THIS CHAPTER

- [Command Line Utility Overview | 730](#)
- [Access Control Arguments | 731](#)
- [Action Arguments | 733](#)
- [Setting Session Limits Using the Command Line Utility | 737](#)
- [Examples of Issuing CoA/DM Requests Using the Command Line Utility | 739](#)
- [Shortcut Arguments | 741](#)
- [Finding All Sessions Using the Command Line Utility | 742](#)

This chapter describes how to manage and control sessions in SBR Carrier using the command line utility. The following topics are included:

Command Line Utility Overview

A command line utility is available for managing and controlling sessions in Steel-Belted Radius Carrier. By default, you can use the command line utility to perform actions such as query, view, and delete. If you have purchased the optional Session Control module license, you can use the command line utility to issue CoA/DM client requests (*actions*).

The command line utility is useful if you want to run a command quickly. You can run commands in batch mode at night or in off hours. For example, you might create a set of batch files to periodically find all the sessions on a server so that you can later review the server activity.

NOTE: The command line utility is designed only to be used to manage and control sessions, not as a primary management interface.

You can use the command line utility to administer CoA/DM functions locally (on the server running Steel-Belted Radius Carrier) or remotely (on another computer that has a network connection to the Steel-Belted Radius Carrier server).

Starting the Command Line Utility

To start the command line utility:

- 1. Go to the directory in which you installed the Steel-Belted Radius Carrier software.
- 2. From this directory, type **SessionControl.sh**.

Example

```
$ cd /opt/JNPRsbrha
$ SessionControl.sh
```

Using Command Line Arguments

Table 107 on page 731 lists the types of arguments used by the command line utility. Each of these types of arguments is discussed in the following sections.

Table 107: Types of Command Line Arguments

Argument Type	Description
Access Control	Control security between the server and the client.
Action	Indicate the type of action you want to execute and the attributes that need to be specified.
Shortcuts	Abbreviated way to indicate attributes.

Access Control Arguments

The Access Control arguments control the access between the server and client.

Syntax

```
-authuser adminname -authpass adminpass [-s server] -p port [-tls ver] -rootca certfile [-ciphersuites ciphersuites]
```

Arguments

Table 108 on page 732 describes the functions of each Access Control argument. Mandatory arguments must be included in the **access_arguments** section of the command line utility.

Table 108: Access Control Arguments

Access Arguments	Description	Mandatory or Optional	Example
-authuser <i>adminname</i>	Administrator's username (of Admin level or Operator level access) for accessing the server.	mandatory	-authuser my_user_name
-authpass <i>adminpass</i>	Password of the administrator.	mandatory	-authpass my_password
-s <i>server</i>	Server name or IP address if the server is not the local host. If this attribute is not used, the localhost is assumed to be the server.	optional	-s my_server_name -s 123.45.6.78
-p <i>port</i>	TCP port on the server into which the command line utility connects to the server.	mandatory	-p 1814
-rootca <i>certfile</i>	Filename containing the certificate name that must be passed by the client to the server to validate the client's right to access the server. The filename is usually root.cer. The file always has a .cer extension. Include the entire path of the file.	mandatory	-rootca /opt/JNPRsbr/root /autoGeneratedRoot.cer
-tls <i>ver</i>	TLS protocol version on which the server expects the client to initiate the handshake process. Default value is 31. If you set a value other than 31, 32, or 33, then the default TLS protocol version 1.0 (31) is considered.	optional	-tls 32
-ciphersuites <i>ciphersuites</i>	TLS cipher suites (in order of preference) that the server is to use. Default value is 0xC00A, 0xC014, 0xC019, 0xC009, 0xC013. NOTE: If you are specifying more than one cipher suite, the cipher suites must be enclosed with double quotation marks ("").	optional	-ciphersuites "0xC014,0xC019"

Example

The following example shows the command line entry for accessing the server and querying for all users:

```
$ SessionControl.sh -authuser my_user -authpass my_password -s my_server_name -p 1814 -rootca
/opt/JNPRsbr/root/autoGeneratedRoot.cer -m query -u ""
```

Action Arguments

Action arguments in the command line indicate the type of action requested, search conditions, and additional arguments needed by the controlled device (NAS).

Syntax

```
-m Action -a Attr -v Value [-a Attr -v Value...][-a Attr -x hex-value...][-sl session_limit][-force]
```

NOTE: The asterisk character (*) represents a wildcard. Wildcards are only accepted for the User and Device Name text attributes. Leading wildcards may be used for User.

Arguments

Table 109 on page 734 describes the functions of each Action argument.

Table 109: Action Arguments

Action Argument	Description	Example
-m Action	<p>Specifies the name of the action you want to perform. Each action must be defined in the deviceModels.xml file.</p> <p>By default, the actions query, view and delete are defined in the deviceModels.xml file. After you define other actions in the deviceModels.xml file, you can use those names as arguments.</p> <p>Valid actions (by default):</p> <p>query</p> <p>view</p> <p>delete</p>	-m query

Table 109: Action Arguments (continued)

Action Argument	Description	Example
-a Attr -v Val	<p>Attribute name and value for which you want to search</p> <p>or</p> <p>The attribute name and value to pass to a NAS if the NAS requires an additional attribute.</p> <p>The valid (searchable) attributes are:</p> <p>Acct-Multi-Session-Id</p> <p>Acct-Session-Id*</p> <p>Called-Station-Id</p> <p>Calling-Station-Id</p> <p>Funk-Attribute-Range</p> <p>Funk-Session-Handle</p> <p>Framed-IP-Address</p> <p>NAS-Identifier</p> <p>NAS-IP-Address</p> <p>User-Name</p>	<p>-a Framed-IP-Address -v 127.0.0.1</p> <p>NOTE: IP addresses may be in either IPv6 or IPv4 format.</p>
-a Attr -x hex-val	Specifies the information being submitted is in hexadecimal format.	-a NAS-IPv6-Address -x 12cdef125f3d
-sl session_limit	Session limit. Upper limit of number of sessions to be found. (See -force to learn how the -force variable affects the interpretation of the session limit.)	-sl 10

Table 109: Action Arguments (continued)

Action Argument	Description	Example
-force	<p>Presence of the -force argument specifies use of the session limit even if the number of found sessions exceeds the session limit. The absence of the -force argument specifies that no sessions be returned or acted upon if the number of matching sessions exceeds the session limit.</p> <p>See “Setting Session Limits Using the Command Line Utility” on page 737 for information about the -force argument.</p>	-sl 10 -force

*The Acct-Session-Id attribute is commonly used in CoA/DM packing lists for router requirements. Packing lists represent the router's requirements for a valid DM or CoA request. Attributes that are not in the CST must come from the actual query to fulfill the request.

To have data available for queries using the command line utility (**SessionControl.sh**), you must configure this attribute in the packing list. If the Sbr_AcctSessionId does not exist in the CST or is not found by the query itself, the request fails.

NOTE: The RADIUS attributes for a controlled device are contained in a file with a **.dic** or **.dct** extension on the machine on which CoA/DM is installed. Open these files to find the names of additional attributes that you might want to pass to the controlled device.

IP Address Ranges

For IP address ranges, specify **rangeBegin** and **rangeEnd** as shown in the following example. Quotation marks are necessary because of the space between **rangeBegin** and **rangeEnd**.

```
-a Funk-Attribute-Range
-v "attribute=NAS-IP-Address rangeBegin=172.16.0.84 rangeEnd=172.16.0.90"
```

Unique Session IDs

For a specific session in the CST, specify the internal session handle returned from a previous query.

```
-a Funk-Session-Handle -v 42A0BA6500000002
```

Setting Session Limits Using the Command Line Utility

You can control the maximum number of sessions that are returned when you execute a query. You may want to limit the number of sessions returned if you are concerned that too many sessions may be found with your query and may impact performance. For example, when you execute a query that does not contain any supported search attributes, then the query is unconstrained and may find too many sessions.

Factors Affecting the Number of Sessions Returned

The number of sessions returned from a query depends on five factors:

- The number you type next to the **-sl** (session limit) argument.
- The system-wide session limit. This number defaults to 10,000 but may be changed with the assistance of your Juniper Networks Technical Support representative.
- The Effective Session Limit. This number is always the lesser of the system-wide session limit and the number you enter next to **-sl** for the session limit.
- The number of sessions found to meet the criteria you enter. (For example, if your criteria is `User-Name = "A*"`, then the sessions found are those with usernames beginning with the letter A or a.)
- Whether or not you use the **-force** argument. The presence of the **-force** argument specifies use of the session limit even if the number of found sessions exceeds the session limit. The absence of **-force** specifies that no sessions be returned or acted upon if the number of matching sessions exceeds the session limit. (See [“Number of Sessions Returned” on page 737](#) and [“Examples of Limiting the Number of Sessions Returned Using the Command Line Utility” on page 738](#) for more information about the use of **-force**.)

Number of Sessions Returned

If you use the **-force** argument, the number of sessions returned is always the lesser of the effective session limit or the number of sessions found. For example, if the session-wide limit is 10,000, you enter a session limit of 100, and the number of sessions found to match your criteria is 150, then 100 sessions (the effective session limit) is returned.

If you do not use the **-force** argument, the number of sessions returned is the actual number of sessions found if the number of sessions found is less than the effective session limit. If the number of sessions found exceeds the effective session limit, then no sessions are returned. For example, if the session-wide limit is 10,000, you enter a session limit of 100, and the number of sessions found to match your criteria is 150, then no sessions are returned because the effective session limit is exceeded.

Examples of Limiting the Number of Sessions Returned Using the Command Line Utility

Table 110 on page 738 provides examples in which the session limit rules apply.

Table 110: Number of Sessions Found for Various Examples

-force Argument Present or Absent	Session Limit You Enter	Session- Wide Session Limit	Effective Session Limit	Sessions Found	Sessions Returned	Reason
Present	100	10,000	100	200	100	Effective limit prevails
Present	100	10,000	100	50	50	Number of sessions found is within effective limit
Present	10	10,000	10	200	10	Effective limit prevails
Present	Nothing	10,000	10,000	200	200	Number of sessions found is within effective limit
Present	20,000	10,000	10,000	25,000	10,000	Effective limit prevails
Absent	100	10,000	100	200	0	Effective limit exceeded
Absent	100	10,000	100	50	50	Number of sessions found is within effective limit

Table 110: Number of Sessions Found for Various Examples (continued)

-force Argument Present or Absent	Session Limit You Enter	Session Limit	Effective Session Limit	Sessions Found	Sessions Returned	Reason
Absent	10	10,000	10	200	10	Effective limit exceeded
Absent	Nothing	10,000	10,000	200	200	Number of sessions found is within effective limit
Absent	20,000	10,000	10,000	25,000	0	Effective limit exceeded

Examples of Issuing CoA/DM Requests Using the Command Line Utility

Query Example Using Wildcard

Query for sessions running on controlled devices with identifiers beginning with the letter L. Return up to 10 sessions.

```
$ SessionControl.sh -authuser my_user -authpass my_password -s my_server_name -p 1814 -rootca
/opt/JNPRsbr/radius/root/autoGeneratedRoot.cer -m query -a User-Name -v "L*" -sl 10 -force
```

Disconnect Example

Disconnect the session that has a session ID of 1234.

```
$ SessionControl.sh -authuser my_user -authpass my_password -s my_server_name -p 1814 -rootca
/opt/JNPRsbr/radius/root/autoGeneratedRoot.cer -m disconnect -a Acct-Session-Id -v 1234
```

NOTE: Acct-Session-Id is a required attribute to execute a disconnect. This attribute is commonly used in CoA/DM packing lists for router requirements. Packing lists represent the router's requirements for a valid DM or CoA request. Attributes that are not in the Current Sessions Table (CST) must come from the actual query to fulfill the request.

To have data available for queries using the command line client, you must configure this attribute in the packing list. If the Sbr_AcctSession-Id does not exist in the CST or is not found by the query itself, the request fails.

For details on customizing the CST, see the *SBR Carrier Installation Guide*.

NOTE: The CoA/DM requests require the SCM license.

Lawful Intercept Example

Intercept the session with the accounting session ID of 12345 and send log information about the session to the specified device.

```
$ SessionControl.sh -authuser my_user -authpass my_password -s my_server_name -p 1814 -rootca
/opt/JNPRsbr/radius/root/autoGeneratedRoot.cer -m interceptOn
-a Acct-Session-Id -v 12345
-a Lawful-Intercept-Action -v 1
-a Lawful-Intercept-Identifier 2468
-a Lawful-Intercept-Mediation-Device-Port-Number -v 4567
-a Lawful-Intercept-Mediation-Device-Port-IP-Address
-v 123.45.6.78
```

NOTE: The attributes needed to execute a lawful intercept are unique to the make and model of the controlled device. Some devices may require the entire word **LegalIntercept** as part of its configuration, and some devices may only use the abbreviation **LI** for their configuration. See the technical documentation of the device for a list of attributes required to execute a lawful intercept or other type of request.

Shortcut Arguments

Shortcut arguments are a quick way to express commands. They are equivalent to certain request arguments.

Syntax

```
-shortcut val
```

Arguments

Table 111 on page 741 describes the functions of each shortcut argument.

Table 111: Shortcut Arguments

Shortcut Argument	Equivalent to	Description	Example
-u <i>username</i>	-a <i>User-Name</i> -v <i>username</i>	Username in the session data	-u "f*" or -u fred
-n <i>NASID</i>	-a <i>NAS-Identifier</i> -v <i>NASID</i>	Identifier required by the controlled device	-n Nas1A
-i <i>sessionid</i>	-a <i>Acct-Session-Id</i> -v <i>sessionid</i>	Accounting session ID used in the CST	-i Session8
-fsh <i>FSH</i>	-a <i>Funk-Session-Handle</i> -v <i>FSH</i>	Internal session handle returned from a previous query against the CST, used to select a specific session in the CST	-fsh 42A0BA6500000002
-calling <i>callID</i>	-a <i>Calling-Station-ID</i> -v <i>callID</i>	Station identifier (usually a telephone number or MAC address) of the user	-calling ID 6175551234

Table 111: Shortcut Arguments (continued)

Shortcut Argument	Equivalent to	Description	Example
-called <i>callID</i>	-a Called-Station-ID -v <i>callID</i>	Station identifier of the NAS (the telephone number dialed by the user or the MAC address of the NAS)	-called ID 6175551234

Disconnect Example with Shortcut

Disconnect the session that has a session ID of 6789.

```
$ SessionControl.sh -authuser my_user -authpass my_password -s my_server_name -p 1814 -rootca /opt/JNPRsbr/radius/root/autoGeneratedRoot.cer -m disconnect -i 6789
```

Finding All Sessions Using the Command Line Utility

You can use the wildcard to find all sessions that are running. The wildcard character is an asterisk enclosed in double quotation marks.

NOTE: The asterisk (wildcard character) and all blank spaces or special characters in the command line client that need to be escaped must be enclosed in double quotation marks.

Example of Finding All Sessions

```
$ SessionControl.sh -authuser my_user -authpass my_pass -s my_server_name -p 1814 -rootca root.cer -m query -u ""
```

Configuring the deviceModels.xml File

IN THIS CHAPTER

- [Summary of Allowed Elements in the deviceModels.xml File | 744](#)
- [Element: action | 747](#)
- [Element: actions | 748](#)
- [Element: attributes | 748](#)
- [Element: controlledDeviceModel | 749](#)
- [Element: controlledDeviceModels | 751](#)
- [Element: defaultAttribute | 752](#)
- [Element: localSessionQuery | 752](#)
- [Element: onFailure | 753](#)
- [Element: onSuccess | 753](#)
- [Element: onTimeout | 754](#)
- [Element: overrideAttribute | 755](#)
- [Element: radiusPort | 755](#)
- [Element: radiusPorts | 756](#)
- [Element: radiusRequest | 757](#)
- [Element: requiredAttribute | 758](#)
- [Element: sessionStop | 758](#)

The **deviceModels.xml** file specifies the CoA/DM actions supported by the various devices in your network. It contains a list of makes and models for each *controlled device object* associated with your device clients. You must configure this file for each device in your network that supports RFC 3576 Change of Authorization (CoA), Disconnect Message (DM), or the Cisco proprietary Packet of Disconnect (PoD).

The following topics are included:

Summary of Allowed Elements in the deviceModels.xml File

Table 112 on page 744 describes the elements allowed in the **deviceModels.xml** file.

Table 112: Allowed Elements in the deviceModels.xml File

Element Name	Comment
action	<p>This release of Steel-Belted Radius Carrier only supports actions that can be expressed as a RADIUS CoA, DM, or PoD requests. It also supports the special action of querying the current sessions table (CST), and returning its contents. All actions must be of type radiusRequest or localSessionQuery. An action request includes some number of attributes, which can be key attributes used to find the session, or action attributes that instruct the device what to do (disconnect, change, and so forth). You can invoke an action through Web GUI, command line utility, or through an XML request sent over the HTTPS port. Any XML application used to issue CoA/DM must conform to the XML interface described in “XML over HTTPS Interface” on page 760.</p>
actions	<p>Contains the list of actions that can be performed against this device model. Different packing lists of AVPs (attribute-value pairs) that cause a particular effect, such as reconfiguring a user's bandwidth or disconnecting a user.</p> <p>NOTE: Actions should be named carefully so that equivalent actions defined against different device models have the same name.</p>
attributes	<p>Contains the actual attribute packing list for the RADIUS request. Each required attribute listed here must be populated from either the client request or the contents of the CST for each session. There are also <i>default</i> and <i>override</i> attributes that can be defined, which are explained in the respective sections.</p> <p>NOTE: Subattributes are not supported in CoA/DM actions.</p>

Table 112: Allowed Elements in the deviceModels.xml File (continued)

Element Name	Comment
controlledDeviceModel	<p>Defines the model for a controlled device object. The controlledDeviceModel is used only if a NAS is configured that has a vendor-product matching the id attribute of this element.</p> <p>The dictionary attribute must be set to the name of one of the dictionaries in the radius directory.</p> <p>The vendor attribute is used only in logging; you must set the attribute with a descriptive value.</p> <p>As shown in the schema in “Element: controlledDeviceModel” on page 749, the controlledDeviceModel is defined in terms of a set of RADIUS ports and a set of actions. The attributes for the controlledDeviceModel are:</p> <ul style="list-style-type: none"> • id—Device name • model—Model name of the device. The value for this attribute must match the model name you specify when configuring the client. • vendor—Brand of the device • port—See radiusPorts element • actions—See actions element
controlledDeviceModels	Defines models for a controlled device object.
defaultAttribute	May be populated either by finding it in the CST or in the client request (action). If it cannot be found, the attribute is populated from the default value specified here.
localSessionQuery	A special action type that uses the attributes specified in the client request, such as User-Name="bo*", to query the current session table (CST). Any matching sessions, up to the currently active safety limit, are returned as the response to the request. No change is made to the session records and no RADIUS request is sent.
onFailure	Session fails.
onSuccess	Session succeeds.
onTimeout	Session times out.

Table 112: Allowed Elements in the deviceModels.xml File (continued)

Element Name	Comment
overrideAttribute	<p>Always set to the value specified in this element, even if it was found in the CST, or in the (action) client request. It can be used to specify constant values that need to be present in the CoA or DM request. For example, an action for a lawful intercept might require that a VSA always be present in the request with the value <i>enable intercept</i>.</p>
radiusPort	<p>Defines a named RADIUS port on which the device receives CoA/DM or PoD requests. The names of these ports are then referenced in the <i>action</i> declarations, so that different actions can be routed to different port numbers.</p> <p>NOTE: The implementation of this element is limited in that each RADIUS port <i>must</i> be named either "RFC3576" or "CiscoPOD" so that at most these two ports exist for each device model.</p>
radiusPorts	<p>Defines named RADIUS ports where the device receives CoA/DM or PoD requests.</p>
radiusRequest	<p>An action type used to define CoA, DM, and PoD requests. When such an action is executed, the attributes specified in the request, such as User-Name="bo*", are used to query the current session table (CST). Any matching sessions, up to the currently active safety limit, are processed by the CoA/DM engine. Using the attributes present in the request, plus any attributes found in the session record for each session, the CoA/DM engine tries to satisfy the attribute packing list for this request. If it can do so, then the RADIUS request is formatted and sent to the device.</p> <p>As shown in the schema in “Element: radiusRequest” on page 757, the radiusRequest is defined in terms of an attributes section, which is required, plus three optional sections, which define the behavior of the server in the event that the request fails, times out, or succeeds. The attributes section is the actual attribute packing list for the request.</p>
requiredAttribute	<p>Must be populated either by finding it in the CST or in the (action) client request. If it cannot be found, the action fails.</p>
sessionStop	<p>Session stops.</p>

Element: action

Actions that are expressed as a RADIUS CoA, DM, or PoD request, and the special action of querying the current sessions table (CST) and returning its contents are supported. All actions must be of type **radiusRequest** or **localSessionQuery**.

NOTE: Name each action something generic so that different vendor devices can use the same names for equivalent actions.

XML Instance Representation

```
<action
description="anySimpleType [0..1]"
name=" xs:NCName [1]">
  Start Choice [1]
    <localSessionQuery> ... </localSessionQuery> [1]
    <radiusRequest> ... </radiusRequest> [1]
  End Choice
</action>
```

Schema Component Representation

```
<xs:element name="action">
  <xs:complexType>
    <xs:choice>
      <xs:element ref=" localSessionQuery "/>
      <xs:element ref=" radiusRequest "/>
    </xs:choice>
    <xs:attribute name="description"/>
    <xs:attribute name="name" type=" xs:NCName " use="required"/>
  </xs:complexType>
</xs:element>
```

Element: actions

This element contains the list of actions that can be performed against this device model.

XML Instance Representation

```
<actions
  <action> ... </action> [0..*]
</actions>
```

Schema Component Representation

```
<xs:element name="actions">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" action " minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: attributes

This element contains the actual attribute packing list for the RADIUS request. Each required attribute listed here must be populated from either the client request or the contents of the CST for each session. You can also define default and override attributes. For more information, see [“Element: defaultAttribute” on page 752](#) and [“Element: overrideAttribute” on page 755](#).

NOTE: Subattributes are not supported in CoA/DM actions.

XML Instance Representation

```
<attributes>
  Start Choice [0..*]
  <defaultAttribute> ... </defaultAttribute> [1]
```

```

        <overrideAttribute> ... </overrideAttribute> [1]
        <requiredAttribute> ... </requiredAttribute> [1]
    End Choice
</attributes>

```

Schema Component Representation

```

<xs:element name="attributes">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref=" defaultAttribute "/>
      <xs:element ref=" overrideAttribute "/>
      <xs:element ref=" requiredAttribute "/>
    </xs:choice>
  </xs:complexType>
</xs:element>

```

Element: controlledDeviceModel

The `controlledDeviceModel` element defines the device model for a controlled device object. It is used only if a device is configured that has a vendor-product matching the `id` attribute of this element.

The **dictionary** attribute must be set to the name of one of the dictionaries in the radius directory. Use the `vendor` and `model` attributes only for logging, and set them to descriptive values.

To use VSAs for a controlled device model, you must explicitly define the VSAs in the corresponding `.dic` file. For example, if you want to use the Juniper VSA “Unisphere-Egress-Policy-Name”, which is defined in the `juniper.dct` file as follows:

```

ATTRIBUTE Unisphere-Egress-Policy-Name          ERX-VSA(11, string)      c

```

Then, you must define the Unisphere-Egress-Policy-Name VSA in the `enterprise_juniper_4874.dic` file as follows:

```

<attribute id="Unisphere-Egress-Policy-Name" type="11" format="string" vendor
="JUNIPER_4874" />

```

You must reboot the SBR Carrier server for the changes to take effect.

As shown in the schema, the `controlledDeviceModel` is defined in terms of a set of RADIUS ports, and a set of actions. For more information, see [“Element: radiusPort” on page 755](#), and [“Element: actions” on page 748](#).

XML Instance Representation

```
<controlledDeviceModel
dictionary=" xs:NCName [0..1]"
id="anySimpleType [1]"
model="anySimpleType [1]"
vendor="anySimpleType [1]">
  <radiusPorts> ... </radiusPorts> [1]
  <actions> ... </actions> [1]
</controlledDeviceModel>
```

Schema Component Representation

```
<xs:element name="controlledDeviceModel">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" radiusPorts "/>
      <xs:element ref=" actions "/>
    </xs:sequence>
    <xs:attribute name="dictionary" type=" xs:NCName "/>
    <xs:attribute name="id" use="required"/>
    <xs:attribute name="model" use="required"/>
    <xs:attribute name="vendor" use="required"/>
  </xs:complexType>
</xs:element>
```

NOTE: You must use a cURL command if you are sending multiple values for a VSA attribute in a single CoA message. For example, in the following XML file named “example.xml”, if you want to send the **Jnpr-Cos-Scheduler-Pmt-Type** attribute that is defined with two different values, you can use the cURL command “**curl -s -u root:password -H \"Content-Type:text/xml\" -d @example.xml -X POST https://10.212.10.120:1814/scs/request/ -k**”.

```
<attributes>
  <attribute enabled="true" name="User-Name" value="test" />
  <attribute enabled="true" name="Acct-Session-Id" value="132ssfdr23" />
  <attribute enabled="true" name="Jnpr-Cos-Scheduler-Pmt-Type"
value="DHCP_DATA 2345" />
  <attribute enabled="true" name="Jnpr-Cos-Scheduler-Pmt-Type"
value="DHCP_DATA 8765" />
</attributes>
```

Element: controlledDeviceModels

This element contains a list of controlledDeviceModels. For more information, see [“Element: controlledDeviceModel” on page 749](#).

XML Instance Representation

```
<controlledDeviceModels
xsi:noNamespaceSchemaLocation="[1]">
  <controlledDeviceModel> ... </controlledDeviceModel> [1..*]
</controlledDeviceModels>
```

Schema Component Representation

```
<xs:element name="controlledDeviceModels">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" controlledDeviceModel " maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute ref=" xsi:noNamespaceSchemaLocation " use="required"/>
  </xs:complexType>
</xs:element>
```

```

    </xs:complexType>
</xs:element>

```

Element: defaultAttribute

A default attribute may be populated either by retrieving it from the CST or from the client request. If it cannot be found, the attribute is populated from the default value specified here.

XML Instance Representation

```

<defaultAttribute
  name=" xs:NCName [1]"
  value="anySimpleType [1]"/>

```

Schema Component Representation

```

<xs:element name="defaultAttribute">
  <xs:complexType>
    <xs:attribute name="name" type=" xs:NCName " use="required"/>
    <xs:attribute name="value" use="required"/>
  </xs:complexType>
</xs:element>

```

Element: localSessionQuery

This special action type uses the attributes specified in the client request, such as **User-Name="bo*"** to query the CST. Any matching sessions, up to the currently active safety limit (defined by session limit), are returned as the response to this action. No change is made to the session records and no RADIUS request is sent.

XML Instance Representation

```

<localSessionQuery

```

```
description="anySimpleType [1]"/>
```

Schema Component Representation

```
<xs:element name="localSessionQuery">
  <xs:complexType>
    <xs:attribute name="description" use="required"/>
  </xs:complexType>
</xs:element>
```

Element: onFailure

This element defines what you want SBR Carrier to do when the NAS responds with a NAK.

XML Instance Representation

```
<onFailure>
  <sessionStop> ... </sessionStop> [0..1]
</onFailure>
```

Schema Component Representation

```
<xs:element name="onFailure">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" sessionStop " minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: onSuccess

This element defines what you want SBR Carrier to do when the NAS responds with an ACK.

XML Instance Representation

```
<onSuccess>
  <sessionStop> ... </sessionStop> [0..1]
</onSuccess>
```

Schema Component Representation

```
<xs:element name="onSuccess">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" sessionStop " minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: onTimeout

This element defines what you want SBR Carrier to do when the request times out.

XML Instance Representation

```
<onTimeout>
  <sessionStop> ... </sessionStop> [0..1]
</onTimeout>
```

Schema Component Representation

```
<xs:element name="onTimeout">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" sessionStop " minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```


Element: overrideAttribute

An overrideAttribute is always set to the value specified in this element, even if it is found in the CST, or in the action client request. You can use the overrideAttribute to specify constant values that are required in the CoA or DM (action) request. For example, a request for a lawful intercept action may require that a VSA always be present in the request with the value: *enable intercept*.

XML Instance Representation

```
<overrideAttribute
  name=" xs:NCName [1]"
  value="anySimpleType [1]"/>
```

Schema Component Representation

```
<xs:element name="overrideAttribute">
  <xs:complexType>
    <xs:attribute name="name" type=" xs:NCName " use="required"/>
    <xs:attribute name="value" use="required"/>
  </xs:complexType>
</xs:element>
```

Element: radiusPort

This element defines a named RADIUS port where the device can receive CoA/DM or PoD requests. The names of these ports are then referenced in the action declarations so that different actions can be routed to different port numbers.

NOTE: You must name each RADIUS port either **RFC3576** or **CiscoPOD** so that at most, these two ports exist for each device model. You cannot configure port numbers or shared secrets for any ports other than these.

XML Instance Representation

```
<radiusPort
description="anySimpleType [0..1]"
name=" xs:NCName [1]"
port=" xs:integer [1]"/>
```

Schema Component Representation

```
<xs:element name="radiusPort">
  <xs:complexType>
    <xs:attribute name="description"/>
    <xs:attribute name="name" type=" xs:NCName " use="required"/>
    <xs:attribute name="port" type=" xs:integer " use="required"/>
  </xs:complexType>
</xs:element>
```

Element: radiusPorts

This element defines a list of RADIUS port where the device can receive CoA/DM or PoD requests.

XML Instance Representation

```
<radiusPorts>
  <radiusPort> ... </radiusPort> [1..*]
</radiusPorts>
```

Schema Component Representation

```
<xs:element name="radiusPorts">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" radiusPort " maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element: radiusRequest

This action type is used to define CoA, DM, and PoD requests. When such an action is executed, the attributes specified in the request, such as **User-Name="bo*"**, are used to query the CST. Any matching sessions, up to the currently active safety limit, are processed by the CoA/DM engine. Using the attributes present in the client request, plus any attributes found in the session record for each session, the CoA/DM engine attempts to satisfy the attribute packing list for this request. If it can do so, then the RADIUS request is formatted and sent to the device.

The **radiusRequest** is defined in terms of an **attributes** section (required), and three optional sections which define the behavior of SBR Carrier in the event the request fails, times out, or succeeds. The **attributes** section is the actual packing list for the request.

XML Instance Representation

```
<radiusRequest
code=" xs:NCName [1]"
description="anySimpleType [0..1]"
dictionary=" xs:NCName [0..1]"
portName=" xs:NCName [1]">
  <attributes> ... </attributes> [1]
  <onSuccess> ... </onSuccess> [0..1]
  <onFailure> ... </onFailure> [0..1]
  <onTimeout> ... </onTimeout> [0..1]
</radiusRequest>
```

Schema Component Representation

```
<xs:element name="radiusRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" attributes " />
      <xs:element ref=" onSuccess " minOccurs="0"/>
      <xs:element ref=" onFailure " minOccurs="0"/>
      <xs:element ref=" onTimeout " minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="code" type=" xs:NCName " use="required"/>
    <xs:attribute name="description"/>
    <xs:attribute name="dictionary" type=" xs:NCName " />
    <xs:attribute name="portName" type=" xs:NCName " use="required"/>
  </xs:complexType>
</xs:element>
```

```

    </xs:complexType>
  </xs:element>

```

Element: requiredAttribute

A required attribute must be populated either by finding it in the CST or in the action request. If it is not found, the action fails.

XML Instance Representation

```

<requiredAttribute
  name=" xs:NCName [1]" />

```

Schema Component Representation

```

<xs:element name="requiredAttribute">
  <xs:complexType>
    <xs:attribute name="name" type=" xs:NCName " use="required"/>
  </xs:complexType>
</xs:element>

```

Element: sessionStop

You can use the sessionStop element in conjunction with the onFailure, onSuccess and onTimeout elements to instruct SBR Carrier to stop the session when these conditions occur. The sessionStop element is relevant when performing a disconnect action. In particular, the sessionStop element can be used to stop the session when the NAS does not send an Accounting-Request (stop).

XML Instance Representation

```

<sessionStop
  description="anySimpleType [1]" />

```

Schema Component Representation

```
<xs:element name="sessionStop">
  <xs:complexType>
    <xs:attribute name="description" use="required"/>
  </xs:complexType>
</xs:element>
```

XML over HTTPS Interface

IN THIS CHAPTER

- XML over HTTPS Interface Overview | 760
- XML Statement Construction | 761
- Client Request Schema Example | 762
- Client Request Elements | 763
- Client Request Examples | 767
- Client Response Schema Example | 770
- Client Response Elements | 774
- Client Response Examples | 789

This chapter provides the XML schema and format for the application programming interface (API) used to actuate the CoA subsystem in Steel-Belted Radius Carrier. If you want to develop your own management client to issue requests to the SBR Carrier CoA subsystem, your management application must adhere to this API. The following topics are included:

XML over HTTPS Interface Overview

Steel-Belted Radius Carrier includes an application programming interface (API), which is a proprietary XML client request and response interface that runs over an HTTPS connection (HTTP over TLS). The protocol used for this interface is mirrored on the Simple Object Access Protocol (SOAP), though it is not identical to it. The Web GUI and command line utility (SessionControl.sh) management clients interact with this interface when issuing CoA/DM client requests, and when receiving CoA/DM client responses. The connection for XML client requests over HTTPS to the session control module need to be made to <https://<server-address>:1814/scs/request>.

If you choose to develop your own management client to issue CoA/DM actions, all client requests must be in the XML format specified in “[Client Request Schema Example](#)” on page 762. In addition, your client management application must be capable of handling client responses from Steel-Belted Radius Carrier in the XML format specified in “[Client Response Schema Example](#)” on page 770. Refer to

“Overview of the Optional Session Control Module” on page 700 for a discussion on how Steel-Belted Radius Carrier processes CoA/DM requests.

Transport Protocol

This proprietary XML request/response interface runs over an HTTPS connection (HTTP over TLS). Most languages (Java, perl, and the like) have a free HTTPS client that can be used for the management client. The SBR Carrier HTTPS implementation is fully compliant with HTTP 1.1
<http://www.w3.org/Protocols/rfc2616/rfc2616.html>.

XML Statement Construction

Your XML client management application must construct CoA/DM client requests in the proper XML format, and must be capable of handling the XML client responses sent to it. The schemas for both the client request and client response are detailed in the following sections.

For client requests, all attributes sent over this interface are directly specified in XML. The CoA/DM action to be carried out (disconnect, lawful intercept, or whatever action you decide to design and specify in the **deviceModels.xml** file) is also specified in XML (for example `<request action=disconnect>`).

The client response XML is very verbose and includes all of the following (the XML element name is shown in ()):

- The request that was sent (*clientRequest*)
- One or more sessions that were matched by the query (*sessionData*)
- The request, as customized for **each** session (*sessionRequest*)
- One or more CoA or DM messages (if any) that were sent to devices (*deviceRequest*)
- Any responses received for those device requests (*deviceResponse*)
- The resulting per-session response to the request (*sessionResponse*)
- In the event of a failure to emit a request because of missing data, the specification of which attributes is required in order to emit the request (*deviceRequestSpec*) various result codes

In the following sections, refer to <http://www.w3.org/XML/Schema> for documentation of the XML schema language.

Client Request Schema Example

The following example shows an XML schema for client requests, which you can use to machine-validate the input to the XML interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="envelope">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="header" />
        <xs:element ref="body" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="header">
    <xs:complexType/>
  </xs:element>
  <xs:element name="body">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="request" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="request">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="attributes" />
      </xs:sequence>
      <xs:attribute name="action" use="required" type="xs:string" />
      <xs:attribute name="force" type="xs:boolean" />
      <xs:attribute name="session_limit" type="xs:integer" />
    </xs:complexType>
  </xs:element>
  <xs:element name="attributes">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="attribute" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```



```

</xs:element>
<xs:element name="attribute">
  <xs:complexType>
    <xs:attribute name="enabled" type="xs:boolean"/>
    <xs:attribute name="name" use="required" type="xs:string"/>
    <xs:attribute name="value" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Client Request Elements

This section provides a detailed description of each element in a CoA/DM client request. A description is provided for each element, along with both the XML format and schema representation.

Element: attribute

In the CoA/DM client request, attributes are not differentiated by type (required, default) because they are used to provide values for both key attributes and action attributes. All attributes are represented as `<attribute>` nodes with a value. If the name of the attribute matches the name of an attribute in a dictionary on Steel-Belted Radius Carrier, then the value must conform to the data type of that attribute. For example, if the data type is integer, then the value must be a quoted integer. If the data type is octets or hex, the value must be either a quoted string (to be interpreted as ASCII), or a quoted string of the form `"{hex}0123456789abcdef"` to directly specify the binary payload of the attribute.

If the enabled attribute is set to false, then the attribute is ignored. This syntax is supported only for the convenience of Web GUI, so that attributes can be ignored without having to take them out of the XML. Using this feature is not normally required because you can simply omit the attribute.

XML Instance Representation

```

<attribute
  enabled="boolean [0..1]"
  name="string [1]"
  value="string [1]"/>

```

Schema Component Representation

```
<element name="attribute">
  <complexType>
    <attribute name="enabled" type=" boolean " />
    <attribute name="name" type=" string " use="required"/>
    <attribute name="value" type=" string " use="required"/>
  </complexType>
</element>
```

Element: attributes

This element groups a set attribute nodes. The request can include any number of attributes nodes in order to group attributes in some arbitrary way. These groupings have no semantic meaning, but can make it easier to structure the client request. You do not need to have more than one attribute node.

XML Instance Representation

```
<attributes>
  <attribute> ... </attribute> [1..*]
</attributes>
```

Schema Component Representation

```
<element name="attributes">
  <complexType>
    <sequence>
      <element ref=" attribute " maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
```

Element: body

This element has no semantic function and is placed in the output to mirror the structure of a SOAP request.

XML Instance Representation

```
<body>
<request> ... </request> [1]
</body>
```

Schema Component Representation

```
<element name="body">
  <complexType>
    <sequence>
      <element ref=" request " />
    </sequence>
  </complexType>
</element>
```

Element: envelope

This element has no semantic function and is placed in the output to mirror the structure of a SOAP request.

XML Instance Representation

```
<envelope>
<header> ... </header> [1]
<body> ... </body> [1]
</envelope>
```

Schema Component Representation

```
<element name="envelope">
  <complexType>
    <sequence>
      <element ref=" header " />
      <element ref=" body " />
    </sequence>
  </complexType>
</element>
```

Element: header

This element has no semantic function and is placed in the output to mirror the structure of a SOAP request.

XML Instance Representation

```
<header/>
```

Schema Component Representation

```
<element name="header">
  <complexType/>
</element>
```

Element: request

This element represents the client request. Its contents become the clientRequest element in the response. (See [“Client Response Schema Example” on page 770](#).) The *action* attribute specifies the name of an action that must match one of the actions defined in the **deviceModels.xml** file. For more information about action names see [“How Steel-Belted Radius Carrier Processes CoA/DM Messages” on page 702](#).

The *force* argument specifies whether to execute the action if the number of affected sessions exceeds the defined *session limit*. See [“Setting Session Limits Using the Command Line Utility” on page 737](#) for information about the effects of setting the *force* argument in addition to session limits.

The *session limit* argument specifies the number of sessions to be maximally affected by the action. For example, a Disconnect action for user “b*” disconnects at most the number of sessions defined by *session limit*. If *force* is not specified, the action is terminated if too many sessions are matched. If *force* is specified, the action proceeds and only the number of sessions specified by *session limit* are affected.

XML Instance Representation

```
<request
  action="string [1]"
  force="boolean [0..1]"
  session_limit="integer [0..1]">
  <attributes> ... </attributes> [1..*]
</request>
```

Schema Component Representation

```
<element name="request">
  <complexType>
    <sequence>
      <element ref=" attributes " maxOccurs="unbounded"/>
    </sequence>
    <attribute name="action" type=" string " use="required"/>
    <attribute name="force" type=" boolean "/>
    <attribute name="session_limit" type=" integer "/>
  </complexType>
</element>
```

Client Request Examples

The following examples show the XML format of a client request.

Example: Query

This example performs a query for up to 1000 sessions with any username:

```
<envelope>
  <header />
  <body>
    <request action="query" force="true" session_limit="1000">
      <attributes>
        <attribute enabled="true" name="User-Name" value="*" />
      </attributes>
    </request>
  </body>
</envelope>
```

Example: Query

This example performs a query for up to 1000 sessions with any username beginning with the letter 'A':

```
<envelope>
  <header />
```

```

<body>
  <request action="query" force="true" session_limit="1000">
    <attributes>
      <attribute enabled="true" name="User-Name" value="A*" />
    </attributes>
  </request>
</body>
</envelope>

```

Example: RADIUS Disconnect

This example performs a RADIUS Disconnect against a specific session, using a handle previously obtained in a query:

```

<envelope>
  <header />
  <body>
    <request action="disconnect">
      <attributes>
        <attribute name="Funk-Session-Handle" value="41f7e1780000000a" />
      </attributes>
    </request>
  </body>
</envelope>

```

Example: RADIUS Disconnect

This example performs a RADIUS Disconnect against all users whose names start with “Test” unless there are more than 10 such users, in which case no action is taken:

```

<envelope>
  <header />
  <body>
    <request action="disconnect" force="false" session_limit="10">
      <attributes>
        <attribute enabled="true" name="User-Name" value="Test*" />
      </attributes>
    </request>
  </body>
</envelope>

```

Example: RADIUS Disconnect

This example performs a RADIUS Disconnect against 3 specific sessions identified by handles obtained in previous queries:

```
<envelope>
  <header />
  <body>
    <request action="disconnect">
      <attributes>
        <attribute name="Funk-Session-Handle" value="41f7e178000000003" />
        <attribute name="Funk-Session-Handle" value="41f7e178000000005" />
        <attribute name="Funk-Session-Handle" value="41f7e178000000008" />
      </attributes>
    </request>
  </body>
</envelope>
```

Example: (CoA) Action Called Intercept

This example performs an action called Intercept for up to 10 sessions of the user, named "Bob":

```
<envelope>
  <header />
  <body>
    <request action="intercept" force="true" session_limit="10" >
      <attributes>
        <attribute enabled="true" name="User-Name" value="Bob" />
      </attributes>
    </request>
  </body>
</envelope>
```

NOTE: COA/DM disconnect multiple users via XML is supported only using wildcard entries (for all attributes in the session), and for specific session, multiple disconnect can be performed using "Funk-Session-Handle". Funk-Session-Handle refers to the "UniqueSessionId" in the session table.

Client Response Schema Example

The following example shows an XML schema for client responses, which you can use to machine-validate the input to the XML interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="envelope">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="header"/>
        <xs:element ref="body"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="header">
    <xs:complexType/>
  </xs:element>
  <xs:element name="body">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="clientResults"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="clientResults">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="clientResult"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="clientResult">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="clientRequest"/>
        <xs:element ref="clientResponse"/>
      </xs:sequence>
        <xs:attribute name="sequence" use="required" type="xs:integer"/>
      </xs:complexType>
    </xs:element>
  <xs:element name="clientRequest">
    <xs:complexType>
```



```

    <xs:sequence>
      <xs:element minOccurs="0" ref="attributes"/>
    </xs:sequence>
    <xs:attribute name="action" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="clientResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="sessionResults"/>
    </xs:sequence>
    <xs:attribute name="resultCode" use="required" type="xs:integer"/>
    <xs:attribute name="resultMessage" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="sessionResults">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="sessionResult"/>
    </xs:sequence>
    <xs:attribute name="type" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="sessionResult">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" ref="sessionData"/>
      <xs:element ref="sessionRequest"/>
      <xs:element minOccurs="0" ref="deviceResults"/>
      <xs:element ref="sessionResponse"/>
    </xs:sequence>
    <xs:attribute name="sequence" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="sessionData">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="attributes"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="sessionRequest">
  <xs:complexType>
    <xs:sequence>

```

```

        <xs:element minOccurs="0" ref="attributes"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="deviceResults">
    <xs:complexType>
        <xs:sequence>
            <xs:element maxOccurs="unbounded" ref="deviceResult"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="deviceResult">
    <xs:complexType>
        <xs:choice minOccurs="0">
            <xs:element ref="deviceRequestSpec"/>
            <xs:sequence>
                <xs:element ref="deviceRequest"/>
                <xs:element ref="deviceResponse"/>
            </xs:sequence>
        </xs:choice>
        <xs:attribute name="device" use="required" type="xs:string"/>
        <xs:attribute name="sequence" use="required" type="xs:integer"/>
    </xs:complexType>
</xs:element>
<xs:element name="deviceRequestSpec">
    <xs:complexType>
        <xs:sequence>
            <xs:element maxOccurs="unbounded" ref="requiredAttribute"/>
            <xs:element maxOccurs="unbounded" ref="optionalAttribute"/>
            <xs:element maxOccurs="unbounded" ref="defaultAttribute"/>
            <xs:element maxOccurs="unbounded" ref="overrideAttribute"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="requiredAttribute">
    <xs:complexType>
        <xs:attribute name="name" use="required" type="xs:string"/>
        <xs:attribute name="sequence" use="required" type="xs:integer"/>
    </xs:complexType>
</xs:element>
<xs:element name="optionalAttribute">
    <xs:complexType>
        <xs:attribute name="name" use="required" type="xs:string"/>
        <xs:attribute name="sequence" use="required" type="xs:integer"/>
    </xs:complexType>
</xs:element>

```

```

    </xs:complexType>
</xs:element>
<xs:element name="defaultAttribute">
  <xs:complexType>
    <xs:attribute name="name" use="required" type="xs:string"/>
    <xs:attribute name="sequence" use="required" type="xs:integer"/>
    <xs:attribute name="value" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="overrideAttribute">
  <xs:complexType>
    <xs:attribute name="name" use="required" type="xs:string"/>
    <xs:attribute name="sequence" use="required" type="xs:integer"/>
    <xs:attribute name="value" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="deviceRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="attributes"/>
    </xs:sequence>
    <xs:attribute name="address" use="required" type="xs:string"/>
    <xs:attribute name="code" use="required" type="xs:string"/>
    <xs:attribute name="port" use="required" type="xs:integer"/>
    <xs:attribute name="type" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="deviceResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="attributes"/>
    </xs:sequence>
    <xs:attribute name="resultCode" use="required" type="xs:integer"/>
    <xs:attribute name="resultMessage" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="sessionResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" ref="attributes"/>
    </xs:sequence>
    <xs:attribute name="resultCode" use="required" type="xs:integer"/>
    <xs:attribute name="resultMessage" use="required" type="xs:string"/>
  </xs:complexType>

```

```

</xs:element>
<xs:element name="attributes">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="attribute"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="attribute">
  <xs:complexType>
    <xs:attribute name="name" use="required" type="xs:string"/>
    <xs:attribute name="sequence" use="required" type="xs:integer"/>
    <xs:attribute name="value" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Client Response Elements

This section provides a detailed description of each element in the client response. A description is provided for each element, along with both the XML and schema representation.

Element: attribute

This element represents a RADIUS attribute or VSA that is returned from a query or other operation.

XML Instance Representation

```

<attribute
name="string [1]"
sequence="integer [1]"
value="string [1]"/>

```

Schema Component Representation

```

<element name="attribute">
  <complexType>
    <attribute name="name" type=" string " use="required"/>
    <attribute name="sequence" type=" integer " use="required"/>
  </complexType>
</element>

```

```
<attribute name="value" type=" string " use="required"/>
</complexType>
</element>
```

Element: attributes

This element is simply a grouping node to collect a set of <attribute> nodes in the returned structure.

NOTE: Subattributes are not supported in CoA/DM actions.

XML Instance Representation

```
<attributes>
<attribute> ... </attribute> [1..*]
</attributes>
```

Schema Component Representation

```
<element name="attributes">
<complexType>
<sequence>
<element ref=" attribute " maxOccurs="unbounded"/>
</sequence>
</complexType>
</element>
```

Element: body

This element has no semantic function and is placed in the output to mirror the structure of a SOAP response.

XML Instance Representation

```
<body>
<clientResults> ... </clientResults> [1]
</body>
```

Schema Component Representation

```
<element name="body">
  <complexType>
    <sequence>
      <element ref=" clientResults " />
    </sequence>
  </complexType>
</element>
```

Element: clientRequest

This element contains the query and action attributes sent by the client in the previous client request. It is presented in the client response for reference only.

XML Instance Representation

```
<clientRequest
action="string [1]">
  <attributes> ... </attributes> [0..1]
</clientRequest>
```

Schema Component Representation

```
<element name="clientRequest">
  <complexType>
    <sequence>
      <element ref=" attributes " minOccurs="0" />
    </sequence>
    <attribute name="action" type=" string " use="required" />
  </complexType>
</element>
```

Element: clientResponse

This top-level element contains the entire response to the client request. It aggregates multiple per-session responses for all affected sessions.

XML Instance Representation

```

<clientResponse
  resultCode="integer [1]"
  resultMessage="string [1]">
  <sessionResults> ... </sessionResults> [0..*]
</clientResponse>

```

Schema Component Representation

```

<element name="clientResponse">
  <complexType>
    <sequence>
      <element ref=" sessionResults " minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="resultCode" type=" integer " use="required"/>
    <attribute name="resultMessage" type=" string " use="required"/>
  </complexType>
</element>

```

Element: clientResult

This element is merely a grouping of a clientRequest and related clientResponse and has no semantic meaning.

XML Instance Representation

```

<clientResult
  sequence="integer [1]">
  <clientRequest> ... </clientRequest> [1]
  <clientResponse> ... </clientResponse> [1]
</clientResult>

```

Schema Component Representation

```

<element name="clientResult">
  <complexType>
    <sequence>
      <element ref=" clientRequest "/>
      <element ref=" clientResponse "/>
    </sequence>

```

```

<attribute name="sequence" type=" integer " use="required"/>
</complexType>
</element>

```

Element: clientResults

This element groups multiple clientResult elements. This release of Steel-Belted Radius Carrier supports only a single clientResult object per response, so this element is for future expansion.

XML Instance Representation

```

<clientResults>
<clientResult> ... </clientResult> [1..*]
</clientResults>

```

Schema Component Representation

```

<element name="clientResults">
<complexType>
<sequence>
<element ref=" clientResult " maxOccurs="unbounded"/>
</sequence>
</complexType>
</element>

```

Element: defaultAttribute

This element appears in a deviceRequestSpec element, and specifies that this attribute may be provided in the clientRequest; otherwise it is defaulted to the specified value.

NOTE: Subattributes are not supported in CoA/DM actions.

XML Instance Representation

```

<defaultAttribute
name="string [1]"
sequence="integer [1]"

```



```
value="string [1]"/>
```

Schema Component Representation

```
<element name="defaultAttribute">
  <complexType>
    <attribute name="name" type=" string " use="required"/>
    <attribute name="sequence" type=" integer " use="required"/>
    <attribute name="value" type=" string " use="required"/>
  </complexType>
</element>
```

Element: deviceRequest

This element contains the exact request sent to a device, if any. By searching this element, the client management application can determine the actual attributes sent to the device, and whether they were populated from the client request or from the session database (CST).

XML Instance Representation

```
<deviceRequest
address="string [1]"
code="string [1]"
port="integer [1]"
type="string [1]">
  <attributes> ... </attributes> [1]
</deviceRequest>
```

Schema Component Representation

```
<element name="deviceRequest">
  <complexType>
    <sequence>
      <element ref=" attributes "/>
    </sequence>
    <attribute name="address" type=" string " use="required"/>
    <attribute name="code" type=" string " use="required"/>
    <attribute name="port" type=" integer " use="required"/>
    <attribute name="type" type=" string " use="required"/>
```

```

</complexType>
</element>

```

Element: deviceRequestSpec

This element is only present when a request fails because it required attributes that were present in neither the clientRequest or the session table (CST).

XML Instance Representation

```

<deviceRequestSpec>
  <requiredAttribute> ... </requiredAttribute> [1..*]
  <optionalAttribute> ... </optionalAttribute> [1..*]
  <defaultAttribute> ... </defaultAttribute> [1..*]
  <overrideAttribute> ... </overrideAttribute> [1..*]
</deviceRequestSpec>

```

Schema Component Representation

```

<element name="deviceRequestSpec">
  <complexType>
    <sequence>
      <element ref=" requiredAttribute " maxOccurs="unbounded"/>
      <element ref=" optionalAttribute " maxOccurs="unbounded"/>
      <element ref=" defaultAttribute " maxOccurs="unbounded"/>
      <element ref=" overrideAttribute " maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>

```

Element: deviceResponse

This element contains the attributes received in a response from the device, if any. Usually, there are not any attributes, except in the case of a NAK with an error result. The attribute *resultCode* carries a numeric code that is meaningful to Juniper Networks Technical Support, and the *resultMessage* carries a human-readable result message. [Table 113 on page 781](#) lists all result codes with the corresponding result messages for the deviceResponse element.

Table 113: Result Codes and Result Messages for deviceResponse

Result Codes	Result Messages
3000000	unknown session control result code
3010000	session control request complete
3010100	session control request successful
3010200	session control request failed
3010201	session control request rejected by device
3010202	number of session affected by the session control request exceeds the limit
3020303	need more information to format session control request
3020000	session control request did not complete
3020100	malformed session control request
3020101	session control request from unauthorized source
3020102	session control request discarded by plug-in
3020103	unknown action specified or invalid license for session control request
3020104	unknown attribute specified in session control request
3020200	server ran out of resources while processing session control request
3020201	server is too busy to process session control request
3020202	session control request timed out
3020300	session control request cannot be processed on this server
3020301	no matching session for session control request
3020302	no matching controlled device for session control request
3020600	session control request is retry of request currently in progress

XML Instance Representation

```

<deviceResponse
  resultCode="integer [1]"
  resultMessage="string [1]">
  <attributes> ... </attributes> [1]
</deviceResponse>

```

Schema Component Representation

```

<element name="deviceResponse">
  <complexType>
    <sequence>
      <element ref=" attributes " />
    </sequence>
    <attribute name="resultCode" type=" integer " use="required"/>
    <attribute name="resultMessage" type=" string " use="required"/>
  </complexType>
</element>

```

Element: deviceResult

This element represents a CoA or DM operation that was executed or attempted to execute against a particular device. It contains either a pair of deviceRequest and associated deviceResponse, or only a deviceRequestSpec in cases where the request could not be sent because the required attributes were not available.

XML Instance Representation

```

<deviceResult
  device="string [1]"
  sequence="integer [1]">
  Start Choice [0..1]
  <deviceRequestSpec> ... </deviceRequestSpec> [1]
  <deviceRequest> ... </deviceRequest> [1]
  <deviceResponse> ... </deviceResponse> [1]
  End Choice
</deviceResult>

```

Schema Component Representation

```
<element name="deviceResult">
  <complexType>
    <choice minOccurs="0">
      <element ref=" deviceRequestSpec " />
      <sequence>
        <element ref=" deviceRequest " />
        <element ref=" deviceResponse " />
      </sequence>
    </choice>
    <attribute name="device" type=" string " use="required"/>
    <attribute name="sequence" type=" integer " use="required"/>
  </complexType>
</element>
```

Element: deviceResults

This element groups multiple deviceResult elements, indicating the actual CoA or DM requests that were sent to devices in order to execute the requested action for a particular session.

This release of Steel-Belted Radius Carrier supports only a single deviceRequest for each session. However, we recommend you design your client management application to handle multiple deviceRequest nodes to take advantage of this element in future Steel-Belted Radius Carrier releases.

XML Instance Representation

```
<deviceResults>
  <deviceResult> ... </deviceResult> [1..*]
</deviceResults>
```

Schema Component Representation

```
<element name="deviceResults">
  <complexType>
    <sequence>
      <element ref=" deviceResult " maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
```

Element: envelope

This element has no semantic meaning and is placed here only to mirror the structure of a SOAP response.

XML Instance Representation

```
<envelope>
  <header> ... </header> [1]
  <body> ... </body> [1]
</envelope>
```

Schema Component Representation

```
<element name="envelope">
  <complexType>
    <sequence>
      <element ref=" header " />
      <element ref=" body " />
    </sequence>
  </complexType>
</element>
```

Element: header

This element has no semantic meaning and is placed here only to mirror the structure of a SOAP response.

XML Instance Representation

```
<header/>
```

Schema Component Representation

```
<element name="header">
  <complexType/>
</element>
```

Element: optionalAttribute

This element appears in a deviceRequestSpec and specifies that this attribute may optionally be provided in the clientRequest. For information regarding this element, see [“Configuring the deviceModels.xml File” on page 743.](#)

XML Instance Representation

```
<optionalAttribute
  name="string [1]"
  sequence="integer [1]"/>
```

Schema Component Representation

```
<element name="optionalAttribute">
  <complexType>
    <attribute name="name" type=" string " use="required"/>
    <attribute name="sequence" type=" integer " use="required"/>
  </complexType>
</element>
```

Element: overrideAttribute

This element appears in a deviceRequestSpec and specifies that this attribute is always used with the specified value, even if it was present in the clientRequest. For information regarding this element, see [“Configuring the deviceModels.xml File” on page 743.](#)

XML Instance Representation

```
<overrideAttribute
  name="string [1]"
  sequence="integer [1]"
  value="string [1]"/>
```

Schema Component Representation

```
<element name="overrideAttribute">
  <complexType>
    <attribute name="name" type=" string " use="required"/>
    <attribute name="sequence" type=" integer " use="required"/>
```

```
<attribute name="value" type=" string " use="required"/>
</complexType>
</element>
```

Element: `requiredAttribute`

This element appears in a `deviceRequestSpec` and specifies that this attribute must be provided in the `clientRequest`; otherwise the request cannot be executed. For information regarding this element, see ["Configuring the deviceModels.xml File" on page 743](#).

XML Instance Representation

```
<requiredAttribute
name="string [1]"
sequence="integer [1]"/>
```

Schema Component Representation

```
<element name="requiredAttribute">
  <complexType>
    <attribute name="name" type=" string " use="required"/>
    <attribute name="sequence" type=" integer " use="required"/>
  </complexType>
</element>
```

Element: `sessionData`

This element contains the attributes found in the session table (CST). For each session matched by the `clientRequest`, a `sessionData` element is present.

XML Instance Representation

```
<sessionData>
  <attributes> ... </attributes> [1]
</sessionData>
```


Schema Component Representation

```

<element name="sessionData">
  <complexType>
    <sequence>
      <element ref=" attributes " />
    </sequence>
  </complexType>
</element>

```

Element: sessionRequest

This element contains the clientRequest customized for a particular session, by filling in any missing information from the sessionData element. There is one sessionRequest for each session matched by the clientRequest.

XML Instance Representation

```

<sessionRequest>
  <attributes> ... </attributes> [0..1]
</sessionRequest>

```

Schema Component Representation

```

<element name="sessionRequest">
  <complexType>
    <sequence>
      <element ref=" attributes " minOccurs="0" />
    </sequence>
  </complexType>
</element>

```

Element: sessionResponse

This element contains the response to the sessionRequest for a particular session. There is one sessionRequest for each session matched by the clientRequest.

XML Instance Representation

```

<sessionResponse

```

```

resultCode="integer [1]"
resultMessage="string [1]">
<attributes> ... </attributes> [0..1]
</sessionResponse>

```

Schema Component Representation

```

<element name="sessionResponse">
  <complexType>
    <sequence>
      <element ref=" attributes " minOccurs="0"/>
    </sequence>
    <attribute name="resultCode" type=" integer " use="required"/>
    <attribute name="resultMessage" type=" string " use="required"/>
  </complexType>
</element>

```

Element: sessionResult

This element groups all the activity for a particular session matched by the clientRequest.

XML Instance Representation

```

<sessionResult
sequence="integer [1]">
  <sessionData> ... </sessionData> [0..1]
  <sessionRequest> ... </sessionRequest> [1]
  <deviceResults> ... </deviceResults> [0..1]
  <sessionResponse> ... </sessionResponse> [1]
</sessionResult>

```

Schema Component Representation

```

<element name="sessionResult">
  <complexType>
    <sequence>
      <element ref=" sessionData " minOccurs="0"/>
      <element ref=" sessionRequest "/>
      <element ref=" deviceResults " minOccurs="0"/>
      <element ref=" sessionResponse "/>
    </sequence>
  </complexType>
</element>

```

```

</sequence>
<attribute name="sequence" type=" integer " use="required"/>
</complexType>
</element>

```

Element: sessionResults

This element groups all the sessionResult nodes for the sessions matched by the clientRequest.

XML Instance Representation

```

<sessionResults
  type="string [1]">
  <sessionResult> ... </sessionResult> [1..*]
</sessionResults>

```

Schema Component Representation

```

<element name="sessionResults">
  <complexType>
    <sequence>
      <element ref=" sessionResult " maxOccurs="unbounded"/>
    </sequence>
    <attribute name="type" type=" string " use="required"/>
  </complexType>
</element>

```

Client Response Examples

The following examples show the XML format of a client response.

Example: Client Response to Query for Username 'bob'

This response shows the results of a query for any sessions for username 'bob'. The response shows the request failed because no sessions were found for that username, as indicated by: **resultMessage = no session found**.

```
<envelope>
  <header/>
  <body>
    <clientResults>
      <clientResult sequence = '1'>
        <clientRequest action = 'query'>
          <attributes>
            <attribute name = 'User-Name' value = 'bob' sequence = '1'>
          </attribute>
          </attributes>
        </clientRequest>
        <clientResponse resultCode = '03010101'
          resultMessage = 'no session found'>
          <sessionResults type = 'success'>
          </sessionResults>
          <sessionResults type = 'failure'>
          </sessionResults>
          <sessionResults type = 'timeout'>
          </sessionResults>
          <sessionResults type = 'incomplete'>
          </sessionResults>
        </clientResponse>
      </clientResult>
    </clientResults>
  </body>
</envelope>
```

Example: Client Response to Query for Any Username Using Wildcard

This example shows the response to a successful query request for all sessions with *any* username using the * wildcard. The response retrieved sessions for usernames: TestUser8, TestUser9, and TestUser10. All three sessions are being handled by a device with NAS-Identifier= Building1.Room1.AP1.

```
<envelope>
  <header/>
```

```

<body>
  <clientResults>
    <clientResult sequence = '1'>
      <clientRequest action = 'query'>
        <attributes>
          <attribute name = 'User-Name' value = '*' sequence = '1'>
          </attribute>
        </attributes>
      </clientRequest>
      <clientResponse resultCode = '00010100'
        resultMessage = 'request completed successfully'>
        <sessionResults type = 'success'>
          <sessionResult sequence = '1'>
            <sessionData handle = '41d088e300000009'>
              <attributes>
                <attribute name = 'User-Name' value = 'TestUser8'
                  sequence = '1'>
                </attribute>
                <attribute name = 'NAS-Identifier' value =
                  'Building1.Room1.AP1' sequence = '2'>
                </attribute>
                <attribute name = 'Acct-Session-Id' value = 'Session 8'
                  sequence = '3'>
                </attribute>
                <attribute name = 'Acct-Multi-Session-Id' value = 'Multi
                  Session 8' sequence = '4'>
                </attribute>
              </attributes>
            </sessionData>
            <sessionRequest>
            </sessionRequest>
          </sessionResult>
          <deviceResults>
            <deviceResult device = 'Building1.Room1.Ap1' sequence = '1'>
              <deviceRequest>
              </deviceRequest>
              <deviceResponse resultCode = '00010100' resultMessage =
                'request completed successfully'>
              </deviceResponse>
            </deviceResult>
          </deviceResults>
          <sessionResponse resultCode = '00010100' resultMessage =
            'request completed successfully'>
          </sessionResponse>
        </sessionResult>
      </clientResponse>
    </clientResult>
  </clientResults>
</body>

```

```

<sessionResult sequence = '2'>
  <sessionData handle = '41f7e17800000000a'>
    <attributes>
      <attribute name = 'User-Name' value = 'TestUser9'
        sequence = '1'>
      </attribute>
      <attribute name = 'NAS-Identifier' value =
        'Building1.Room1.AP1' sequence = '2'>
      </attribute>
      <attribute name = 'Acct-Session-Id' value = 'Session 9'
        sequence = '3'>
      </attribute>
      <attribute name = 'Acct-Multi-Session-Id' value = 'Multi
        Session 9' sequence = '4'>
      </attribute>
    </attributes>
  </sessionData>
  <sessionRequest>
</sessionRequest>
  <deviceResults>
    <deviceResult device = 'Building1.Room1.Ap1' sequence = '1'>
      <deviceRequest>
</deviceRequest>
      <deviceResponse resultCode = '00010100' resultMessage =
        'request completed successfully'>
      </deviceResponse>
    </deviceResult>
  </deviceResults>
  <sessionResponse resultCode = '00010100' resultMessage =
    'request completed successfully'>
  </sessionResponse>
</sessionResult>
<sessionResult sequence = '3'>
  <sessionData handle = '41f7e17800000000b'>
    <attributes>
      <attribute name = 'User-Name' value = 'TestUser10'
        sequence = '1'>
      </attribute>
      <attribute name = 'NAS-Identifier' value =
        'Building1.Room1.AP1' sequence = '2'>
      </attribute>
      <attribute name = 'Acct-Session-Id' value = 'Session 10'
        sequence = '3'>
      </attribute>

```

```

        <attribute name = 'Acct-Multi-Session-Id' value = 'Multi
        Session 10' sequence = '4'>
    </attribute>
</attributes>
</sessionData>
<sessionRequest>
</sessionRequest>
<deviceResults>
    <deviceResult device = 'Building1.Room1.Ap1' sequence = '1'>
        <deviceRequest>
        </deviceRequest>
        <deviceResponse resultCode = '00010100' resultMessage =
        'request completed successfully'>
        </deviceResponse>
    </deviceResult>
</deviceResults>
    <sessionResponse resultCode = '00010100' resultMessage =
    'request completed successfully'>
    </sessionResponse>
</sessionResult>
</sessionResults>
<sessionResults type = 'failure'>
</sessionResults>
<sessionResults type = 'timeout'>
</sessionResults>
<sessionResults type = 'incomplete'>
</sessionResults>
</clientResponse>
</clientResult>
</clientResults>
</body>
</envelope>

```

Example: Client Response to Request for Action Called “foo” on Username TestUser9

This example shows the response to a client request to perform an action called “foo” on username=TestUser9. The results show that although a session was found for username TestUser9, the request failed because the action “foo” is not defined in the **deviceModels.xml** file for the device

(NAS-Identifier= Building1.Room1.AP1) handling the session. The failure is indicated by the **resultMessage**: unknown action specified in Device Control request.

```
<envelope>
  <header/>
  <body>
    <clientResults>
      <clientResult sequence = '1'>
        <clientRequest action = 'foo'>
          <attributes>
            <attribute name = 'User-Name' value = 'TestUser9' sequence = '1'>
          </attribute>
          </attributes>
        </clientRequest>
        <clientResponse resultCode = '03010200' resultMessage = 'Device
        Control request failed'>
          <sessionResults type = 'success'>
        </sessionResults>
          <sessionResults type = 'failure'>
            <sessionResult sequence = '1'>
              <sessionData handle = '41f7e17800000000a'>
                <attributes>
                  <attribute name = 'User-Name' value = 'TestUser9'
                  sequence = '1'>
                </attribute>
                  <attribute name = 'NAS-Identifier' value =
                  'Building1.Room1.AP1' sequence = '2'>
                </attribute>
                  <attribute name = 'Acct-Session-Id' value =
                  'Session 9' sequence = '3'>
                </attribute>
                  <attribute name = 'Acct-Multi-Session-Id' value =
                  'Multi Session 9' sequence = '4'>
                </attribute>
                </attributes>
              </sessionData>
            <sessionRequest>
          </sessionRequest>
          <deviceResults>
            <deviceResult device = 'Building1.Room1.Ap1' sequence = '1'>
              <deviceRequest>
                <attributes>
                  <attribute name = 'User-Name' value = 'TestUser9'
                  sequence = '1'>
```



```

        </attribute>
        <attribute name = 'NAS-Identifier' value =
        'Building1.Room1.AP1' sequence = '2'>
        </attribute>
        <attribute name = 'Acct-Session-Id' value =
        'Session 9' sequence = '3'>
        </attribute>
        <attribute name = 'Acct-Multi-Session-Id' value =
        'Multi Session 9' sequence = '4'>
        </attribute>
    </attributes>
</deviceRequest>
<deviceResponse resultCode = '03020103' resultMessage =
    'unknown action specified in Device
    Control request'>
</deviceResponse>
</deviceResult>
</deviceResults>
<sessionResponse resultCode = '03020103' resultMessage =
    'unknown action specified in Device
    Control request'>
</sessionResponse>
</sessionResult>
</sessionResults>
<sessionResults type = 'timeout'>
</sessionResults>
<sessionResults type = 'incomplete'>
</sessionResults>
</clientResponse>
</clientResult>
</clientResults>
</body>
</envelope>

```

Example: Client Response to Request for Action Called “foo” on Username TestUser99

This response shows the results to a client request to perform an action called “foo” on username=TestUser99. In this example, the request failed as indicated by: **resultMessage = 'no matching**

controlled device for Device Control request'. This failure indicates that the NAS handling the session is a model that has no controlledDeviceModel defined in the **deviceModels.xml** file.

```
<envelope>
  <header/>
  <body>
    <clientResults>
      <clientResult sequence = '1'>
        <clientRequest action = 'foo'>
          <attributes>
            <attribute name = 'User-Name' value = 'TestUser99'
              sequence = '1'>
            </attribute>
          </attributes>
        </clientRequest>
        <clientResponse resultCode = '03010200' resultMessage =
          'Device Control request failed'>
          <sessionResults type = 'success'>
          </sessionResults>
          <sessionResults type = 'failure'>
            <sessionResult sequence = '1'>
              <sessionData>
              </sessionData>
              <sessionRequest>
              </sessionRequest>
              <deviceResults>
                <deviceResult sequence = '1'>
                  <deviceRequest>
                  </deviceRequest>
                  <deviceResponse resultCode = '03020302' resultMessage =
                    'no matching controlled device for Device Control
                      request'>
                  </deviceResponse>
                </deviceResult>
              </deviceResults>
              <sessionResponse resultCode = '03020302' resultMessage =
                'no matching controlled device for Device Control request'>
              </sessionResponse>
            </sessionResult>
          </sessionResults>
          <sessionResults type = 'timeout'>
          </sessionResults>
          <sessionResults type = 'incomplete'>
          </sessionResults>
        </clientResponse>
      </clientResult>
    </clientResults>
  </body>
</envelope>
```

```

        </clientResponse>
    </clientResult>
</clientResults>
</body>
</envelope>

```

Example: Client Response to RADIUS Disconnect

This response shows the results to a client request to perform a disconnect action on username TestUser9. The response shows that the request timed out, as indicated by the resultMessage: **Device Control request timed out**.

```

<envelope>
  <header/>
  <body>
    <clientResults>
      <clientResult sequence = '1'>
        <clientRequest action = 'disconnect'>
          <attributes>
            <attribute name = 'User-Name' value = 'TestUser9'
              sequence = '1'>
          </attribute>
        </attributes>
      </clientRequest>
      <clientResponse resultCode = '03020202' resultMessage =
        'Device Control request timed out'>
        <sessionResults type = 'success'>
        </sessionResults>
        <sessionResults type = 'failure'>
        </sessionResults>
        <sessionResults type = 'timeout'>
          <sessionResult sequence = '1'>
            <sessionData handle = '41f7e1780000000a'>
              <attributes>
                <attribute name = 'User-Name' value = 'TestUser9'
                  sequence = '1'>
                </attribute>
                <attribute name = 'NAS-Identifier' value =
                  'Building1.Room1.AP1' sequence = '2'>
                </attribute>
                <attribute name = 'Acct-Session-Id' value =
                  'Session 9' sequence = '3'>

```

```

        </attribute>
        <attribute name = 'Acct-Multi-Session-Id' value =
            'Multi Session 9' sequence = '4'>
        </attribute>
    </attributes>
</sessionData>
<sessionRequest>
    <attributes>
        <attribute name = 'User-Name' value = 'TestUser9'
            sequence = '1'>
        </attribute>
        <attribute name = 'NAS-Identifier' value =
            'Building1.Room1.AP1' sequence = '2'>
        </attribute>
        <attribute name = 'Acct-Session-Id' value = 'Session 9'
            sequence = '3'>
        </attribute>
        <attribute name = 'Acct-Multi-Session-Id' value =
            'Multi Session 9' sequence = '4'>
        </attribute>
    </attributes>
</sessionRequest>
<deviceResults>
    <deviceResult device = 'Building1.Room1.Ap1' sequence = '1'>
        <deviceRequest type="radiusRequest" address=
            "198.186.160.84" port="3799" >
            <attributes>
                <attribute name = 'User-Name' value = 'TestUser9'
                    sequence = '1'>
                </attribute>
                <attribute name = 'Funk-Realm-Name' value = 'users'
                    sequence = '2'>
                </attribute>
                <attribute name = 'Acct-Session-Id' value = 'Session 9'
                    sequence = '3'>
                </attribute>
            </attributes>
        </deviceRequest>
        <deviceResponse resultCode = '03020202' resultMessage =
            'Device Control request timed out'>
        </deviceResponse>
    </deviceResult>
</deviceResults>
<sessionResponse resultCode = '03020202' resultMessage =

```

```

        'Device Control request timed out'>
      </sessionResponse>
    </sessionResult>
  </sessionResults>
  <sessionResults type = 'incomplete'>
  </sessionResults>
</clientResponse>
</clientResult>
</clientResults>
</body>
</envelope>

```

Example: Client Response to Action Intercept

This example shows the client response to a request to perform an action called Intercept on username=TestUser9. In this example, the request failed as indicated by: **resultMessage = 'no matching controlled device for Device Control request'**. This failure indicates that the NAS handling the session is a model that has no controlledDeviceModel defined in the **deviceModels.xml** file.

```

<envelope>
  <header/>
  <body>
    <clientResults>
      <clientResult sequence = '1'>
        <clientRequest action = 'intercept'>
          <attributes>
            <attribute name = 'User-Name' value = 'TestUser9'
              sequence = '1'>
            </attribute>
          </attributes>
        </clientRequest>
        <clientResponse resultCode = '03010200' resultMessage =
          'Device Control request failed'>
          <sessionResults type = 'success'>
          </sessionResults>
          <sessionResults type = 'failure'>
            <sessionResult sequence = '1'>
              <sessionData>
              </sessionData>
              <sessionRequest>
              </sessionRequest>
              <deviceResults>

```

```

        <deviceResult sequence = '1'>
            <deviceRequest>
            </deviceRequest>
            <deviceResponse resultCode = '03020302' resultMessage =
                'no matching controlled device for Device Control
                request'>
            </deviceResponse>
        </deviceResult>
    </deviceResults>
    <sessionResponse resultCode = '03020302' resultMessage =
        'no matching controlled device for Device Control request'>
    </sessionResponse>
</sessionResult>
</sessionResults>
<sessionResults type = 'timeout'>
</sessionResults>
<sessionResults type = 'incomplete'>
</sessionResults>
</clientResponse>
</clientResult>
</clientResults>
</body>
</envelope>

```

Example: Client Response to Action Intercept

This example shows the client response to a request to perform an action called Intercept on username TestUser8. In this case a session was found, but the request failed because the action Intercept was not defined for the device handling the session, as indicated by: **resultMessage: unknown action specified in Device Control request.**

```

<envelope>
    <header/>
    <body>
        <clientResults>
            <clientResult sequence = '1'>
                <clientRequest action = 'intercept'>
                    <attributes>
                        <attribute name = 'User-Name' value = 'TestUser8'
                            sequence = '1'>
                        </attribute>
                    </attributes>

```

```

</clientRequest>
<clientResponse resultCode = '03010200' resultMessage =
  'Device Control request failed'>
  <sessionResults type = 'success'>
  </sessionResults>
  <sessionResults type = 'failure'>
    <sessionResult sequence = '1'>
      <sessionData handle = '41d088e300000009'>
        <attributes>
          <attribute name = 'User-Name' value = 'TestUser8'
            sequence = '1'>
          </attribute>
          <attribute name = 'NAS-Identifier' value =
            'Building1.Room1.AP1' sequence = '2'>
          </attribute>
          <attribute name = 'Acct-Session-Id' value =
            'Session 8' sequence = '3'>
          </attribute>
          <attribute name = 'Acct-Multi-Session-Id' value =
            'Multi Session 8' sequence = '4'>
          </attribute>
        </attributes>
      </sessionData>
    </sessionRequest>
  </sessionRequest>
  <deviceResults>
    <deviceResult device = 'Building1.Room1.Ap1'
      sequence = '1'>
      <deviceRequest>
        <attributes>
          <attribute name = 'User-Name' value = 'TestUser8'
            sequence = '1'>
          </attribute>
          <attribute name = 'NAS-Identifier' value =
            'Building1.Room1.AP1' sequence = '2'>
          </attribute>
          <attribute name = 'Acct-Session-Id' value =
            'Session 8' sequence = '3'>
          </attribute>
          <attribute name = 'Acct-Multi-Session-Id' value =
            'Multi Session 8' sequence = '4'>
          </attribute>
        </attributes>
      </deviceRequest>
    </deviceResult>
  </deviceResults>
</clientResponse>

```

```

        <deviceResponse resultCode = '03020103' resultMessage =
            'unknown action specified in Device Control request'>
        </deviceResponse>
    </deviceResult>
</deviceResults>
    <sessionResponse resultCode = '03020103' resultMessage =
        'unknown action specified in Device Control request'>
    </sessionResponse>
</sessionResult>
</sessionResults>
<sessionResults type = 'timeout'>
</sessionResults>
<sessionResults type = 'incomplete'>
</sessionResults>
</clientResponse>
</clientResult>
</clientResults>
</body>
</envelope>

```

Example: Client Response to Action Intercept

This example shows the client response to a request to perform an action called Intercept on any usernames beginning with: TestUser, as indicated by the * wildcard (TestUser*). In this case, an active session was found for TestUser10, but the request failed because the action Intercept was not defined for the device handling the session, as indicated by: **resultMessage: unknown action specified in Device Control request**.

```

<envelope>
    <header/>
    <body>
        <clientResults>
            <clientResult sequence = '1'>
                <clientRequest action = 'intercept'>
                    <attributes>
                        <attribute name = 'User-Name' value = 'TestUser*'
                            sequence = '1'>
                        </attribute>
                    </attributes>
                </clientRequest>
            <clientResponse resultCode = '03010200' resultMessage =
                'Device Control request failed'>
                <sessionResults type = 'success'>

```



```

</sessionResults>
<sessionResults type = 'failure'>
  <sessionResult sequence = '1'>
    <sessionData handle = '41f7e1780000000b'>
      <attributes>
        <attribute name = 'User-Name' value = 'TestUser10'
          sequence = '1'>
        </attribute>
        <attribute name = 'NAS-Identifier' value =
          'Building1.Room1.AP1' sequence = '2'>
        </attribute>
        <attribute name = 'Acct-Session-Id' value = 'Session 10'
          sequence = '3'>
        </attribute>
        <attribute name = 'Acct-Multi-Session-Id' value = 'Multi
          Session 10' sequence = '4'>
        </attribute>
      </attributes>
    </sessionData>
    <sessionRequest>
  </sessionRequest>
  <deviceResults>
    <deviceResult device = 'Building1.Room1.Ap1' sequence = '1'>
      <deviceRequest>
        <attributes>
          <attribute name = 'User-Name' value = 'TestUser10'
            sequence = '1'>
          </attribute>
          <attribute name = 'NAS-Identifier' value =
            'Building1.Room1.AP1' sequence = '2'>
          </attribute>
          <attribute name = 'Acct-Session-Id' value =
            'Session 10' sequence = '3'>
          </attribute>
          <attribute name = 'Acct-Multi-Session-Id' value =
            'Multi Session 10' sequence = '4'>
          </attribute>
        </attributes>
      </deviceRequest>
      <deviceResponse resultCode = '03020103' resultMessage =
        'unknown action specified in Device Control request'>
      </deviceResponse>
    </deviceResult>
  </deviceResults>

```

```

    <sessionResponse resultCode = '03020103' resultMessage =
      'unknown action specified in Device Control request'>
    </sessionResponse>
  </sessionResult>
  <sessionResult sequence = '2'>
    <sessionData handle = '41d088e300000009'>
      <attributes>
        <attribute name = 'User-Name' value = 'TestUser8'
          sequence = '1'>
        </attribute>
        <attribute name = 'NAS-Identifier' value =
          'Building1.Room1.AP1' sequence = '2'>
        </attribute>
        <attribute name = 'Acct-Session-Id' value = 'Session 8'
          sequence = '3'>
        </attribute>
        <attribute name = 'Acct-Multi-Session-Id' value =
          'Multi Session 8' sequence = '4'>
        </attribute>
      </attributes>
    </sessionData>
    <sessionRequest>
  </sessionRequest>
  <deviceResults>
    <deviceResult device = 'Building1.Room1.Ap1'
      sequence = '1'>
      <deviceRequest>
        <attributes>
          <attribute name = 'User-Name' value = 'TestUser8'
            sequence = '1'>
          </attribute>
          <attribute name = 'NAS-Identifier' value =
            'Building1.Room1.AP1' sequence = '2'>
          </attribute>
          <attribute name = 'Acct-Session-Id' value =
            'Session 8' sequence = '3'>
          </attribute>
          <attribute name = 'Acct-Multi-Session-Id' value =
            'Multi Session 8' sequence = '4'>
          </attribute>
        </attributes>
      </deviceRequest>
      <deviceResponse resultCode = '03020103' resultMessage =
        'unknown action specified in Device Control request'>

```

```
        </deviceResponse>
      </deviceResult>
    </deviceResults>
    <sessionResponse resultCode = '03020103' resultMessage =
      'unknown action specified in Device Control request'>
    </sessionResponse>
  </sessionResult>
</sessionResults>
<sessionResults type = 'timeout'>
</sessionResults>
<sessionResults type = 'incomplete'>
</sessionResults>
</clientResponse>
</clientResult>
</clientResults>
</body>
</envelope>
```

Example CoA/DM Configuration

IN THIS CHAPTER

- Requirements of the CoA/DM Requests | 806
- Requirements for Supporting the Attributes in CoA/DM Requests | 808
- Configuring the Attribute Handling Parameters | 811
- Example Result | 812
- Configuring Lawful-Intercept between SBR Carrier and ERX Device | 813

This chapter provides an example of using the CoA/DM capability of Steel-Belted Radius Carrier in a WiMAX deployment. The following topics are included:

Requirements of the CoA/DM Requests

This example configuration is for a WiMAX deployment using two different dynamic authorization requests: a Disconnect-Request and a CoA-Request. Both requests are executed against all sessions that match a particular Acct-Multi-Session-Id attribute value specified in the CoA/DM client requests.

Requirements for the Disconnect Message Request

The first client request requirement in this example is for a RADIUS Disconnect Message (DM) request. You execute the Disconnect-Request when you want to disconnect a user from the network. The attribute packing list for the Disconnect-Request is determined by the WiMAX equipment used to gain access to the network. In this example, the RADIUS attributes listed in [Table 114 on page 807](#) must be contained in the Disconnect-Request message:

Table 114: Required Attributes for the Disconnect-Request

Attribute	Description
User-Name	<p>The User-Name specified in the Accounting-Start message, which in this example is a pseudonym such as a fast re-authentication identity if EAP-SIM or EAP-AKA authentication is used, or an anonymous outer username if EAP-TTLS is used.</p> <p>NOTE: This username is different from the username normally found in the current session table (CST), which is always the real (inner) identity of the user.</p>
Calling-Station-Id	<p>The Calling-Station-Id AVP from the Access-Request. Because this is a WiMAX deployment, this AVP may contain bytes with the value=0 (Null). Conventional RADIUS devices do not use null in the Calling-Station-Id attribute, so it is usually interpreted as a null-terminated string. In this example, the Calling-Station-Id contains the MAC address of the WiMAX network access device in binary form, so it may contain nulls.</p>
WiMAX-AAA-Session-Id	The WiMAX-AAA-Session-Id from the Access-Accept response.
WiMAX-DM-Action-Code	Set to the 4 byte integer value=0.

Requirements for the CoA (Hotline) Request

The second client request requirement in this example is for a RADIUS Change of Authorization (CoA) request. The CoA-Request is executed when a user needs be redirected to a hotlining or lawful intercept device. The attribute packing list for the CoA-Request is determined by the WiMAX equipment used to access the network. In this example, the following RADIUS attributes must be contained in the CoA-Request:

Table 115: Required Attributes for the CoA-Request

Attribute	Description
User-Name	<p>The User-Name specified in the Accounting-Start message, which in this example is a pseudonym such as the fast re-authentication identity if EAP-SIM or EAP-AKA authentication is used, or an anonymous outer username if EAP-TTLS is used.</p> <p>NOTE: This username is different from the username normally found in the current session table (CST) which is always the real (inner) identity of the user.</p>
Calling-Station-Id	<p>The Calling-Station-Id AVP from the Access-Request. Because this is a WiMAX deployment, this AVP may contain bytes with the value=0 (Null). Conventional RADIUS devices do not use null in the Calling-Station-Id attribute, so it is usually interpreted as a null-terminated string. In this example, the Calling-Station-Id contains the MAC address of the WiMAX network access device in binary form, so it may contain nulls.</p>
WiMAX-AAA-Session-Id	<p>The WiMAX-AAA-Session-Id from the Access Accept response.</p>
WiMAX-Hotline-Profile-ID	<p>Set to the string value=1.</p>

Requirements for Supporting the Attributes in CoA/DM Requests

This section describes the requirements for supporting the attributes in the CoA and DM requests described in [“Requirements of the CoA/DM Requests” on page 806](#).

Dictionaries

In order to be used in CoA or DM messages, all attributes in the packing lists identified in [“Requirements of the CoA/DM Requests” on page 806](#) must be in Steel-Belted Radius Carrier dictionaries (files with the

extension *.dic.) In this example, all attributes appear in the **radius.dic** or **wimax.dic** file, so no additional attributes need to be added.

If any of the required attributes do not exist in a *.dic file, you must add them to the dictionary file. For information about adding attributes to a dictionary file, see *Attribute Processing Files* in the *SBR Carrier Reference Guide*.

deviceModels.xml

The requirements for the attribute packing lists are directly mapped to the **deviceModels.xml** file. Each network access device processing CoA/DM requests needs to be configured as a RADIUS client in SBR Carrier. The Make/Model setting defined for each of these RADIUS clients must match the *controlledDeviceModel id* of the corresponding section in the **deviceModels.xml** file. If no such section exists in **deviceModels.xml** file, you must create one.

The following shows a valid *controlledDeviceModel* XML section for the CoA and DM requests:

```
<controlledDeviceModel id="WiMax"
vendor="juniper"
model="WiMax"
dictionary="wimax">
  <radiusPorts>
    <!--specifies default port, can be changed per device -->
    <radiusPort name="RFC3576" port="3799"/>
  </radiusPorts>
  <actions>
    <action name="query">
      <localSessionQuery description="return local session data"/>
    </action>
    <action name="disconnect">
      <radiusRequest code="DM"
        portName="RFC3576"
        dictionary="wimax">
        <attributes>
          <requiredAttribute name="User-Name"/>
          <requiredAttribute name="Calling-Station-Id"
            convertToBinary="true"/>
          <requiredAttribute name="WiMAX-AAA-Session-Id"/>
          <defaultAttribute name="WiMAX-DM-Action-Code"
            value="0"/>
        </attributes>
      </radiusRequest>
    </action>
  </actions>
  <onSuccess>
    <!--this device does not send Stop when we
      disconnect someone -->
```

```

        <sessionStop
            description="Simulated Session Stop"/>
    </onSuccess>
    <onFailure>
        <!--assume bad session record -->
        <sessionStop description="Cleaning Session Database"/>
    </onFailure>
    <onTimeout/>
</radiusRequest>
</action>
<action name="interceptOn">
    <radiusRequest code="CoA"
        portName="RFC3576"
        dictionary="wimax">
        <attributes>
            <requiredAttribute name="User-Name"/>
            <requiredAttribute name="Calling-Station-Id"
                convertToBinary="true"/>
            <requiredAttribute name="WiMAX-AAA-Session-Id"/>
            <overrideAttribute name="WiMAX-Hotline-Profile-ID"
                value="1"/>
        </attributes>
    </radiusRequest>
</action>
</actions>
</controlledDeviceModel>

```

As you can see in this example, three actions are defined: Query, Disconnect-Request, and a CoA-Request called "interceptOn".

NOTE: Internally, SBR Carrier assumes that the Calling-Station-Id attribute is a readable string. Therefore, when operating in an environment where this attribute can contain nulls, SBR Carrier must convert the attribute to a hexadecimal string after it is received, to encode any non-readable characters. As a result, the value for this attribute must be restored to its binary form before it is sent in a CoA/DM request. To trigger this behavior, the `convertToBinary` attribute is applied to the Calling-Station-Id in this example *controlledDeviceModel*. This flag, if set to true, causes the CoA/DM subsystem to convert the attribute from hexadecimal back to binary before emitting the CoA/DM packet.

Configuring the Attribute Handling Parameters

This section describes how to configure the attribute handling parameters required for this example.

radius.ini

As discussed in [“Requirements of the CoA/DM Requests” on page 806](#), this example requires a binary Calling-Station-Id (the Calling-Station-Id RADIUS AVP may contain nulls). The **ConvertCallingStationId** parameter in the **radius.ini** file enables the conversion of the Calling-Station-Id to hexadecimal form when it is received, even though it is marked and processed as an attribute of type *string*. To enable this feature, set the following configuration in the **radius.ini** file:

```
[Configuration]
ConvertCallingStationId = 1
```

When enabled, this causes the Calling-Station-Id always to internally contain the hexadecimal representation of the received bytes, instead of those bytes directly. By using this feature, all existing SBR Carrier functionality that normally requires Calling-Station-Id as a human-readable string can still be used.

classmap.ini

In this application, the inner username needs to be included in the CST in order to identify active sessions. In addition, because network access devices (RADIUS clients) only identify users by their outer identity, the outer username also needs to be included in the CST, so that it can be included in the CoA/DM requests. To make sure both the outer and inner usernames are available in the Accounting-Request, you need to tunnel the inner username through the class attribute. This is common for many scenarios. However, you also need to preserve the outer username presented in the Accounting-Requests, so that it can also be written to the CST.

You accomplish this by changing **classmap.ini** file to copy the username in Accounting-Requests to the VSA Funk-Outer-User-Name before replacing it with the value tunneled inside the class attribute. Include the following in the **classmap.ini** file:

```
[Class]
add = class
add = User-Name

[User-Name]
replace = User-Name, Funk-Outer-User-Name
```

To test this portion of the configuration, add the Funk-Outer-User-Name attribute to the [Attributes] section of the **account.ini** file. If an Accounting-Request is received that includes this attribute, it is logged

in the accounting log file. If the Funk-Outer-User-Name attribute does not appear in the accounting log file, you may need to add it to the **radius.dct** dictionary file. This attribute is included by default in the **radius.dct** file, so normally you do not need to add it.

Example Result

After you have made all the changes described in the previous sections, you can execute the following requests assuming a session exists with a matching Acct-Multi-Session-Id. These requests can be issued by using the command line utility or Web GUI.

First, issue the Query request, then select a session and issue the interceptOn action (CoA-Request). This request sends log information about the session to another device so that the session activity that occurs on one controlled device can be tracked at a different location. Finally, issue the Disconnect-Request to disconnect the session.

```
hadm@host:~> ./ShowSessions.sh -a
===== [1] xxx xxx xx xx:xx:xx (TZ=+00:00) xxxx =====
CurrentSessions:
+ --- + (1)
CORE
  UniqueSessionId: '01020304050000000000000000000000'x
  CreationTime: xxxx-xx-xx xx:xx:xx (TZ=+00:00)
  ExpirationTime: xxxx-xx-xx xx:xx:xx (TZ=+00:00)
  Ipv4Address: 1.2.3.15
  IpAddrPool: (n u l l)
  NasName: "TESTING-123"
  Status: Active (2)
  UserConcurrencyId: (n u l l)
  MobileIpType: 3
  3gpp2ReqType: 0
  WimaxClientType: 1
  WimaxAcctFlows: (n u l l)
  Ipv6Address: "1100:dbca:abcd:0000:0000:0000:0000:0002"
FEATURE
OPTIONAL
  UserName: "test"
  AcctSessionId: "2"
  CallingStationId: "8573776"
  CalledStationId: (n u l l)
RADATTR
  WimaxSessionId: "xxxxxxxx:00000000:00000000"
  AcctMultiSessionId: "01020304:05000000:00000000"
```

```

    FunkOuterUserName: "anonymous1234567890@demo.test"
PRIVATE
+ ----- + (end)

root@host:~> ./SessionControl.sh -authuser my_user -authpass my_password -s my_server_name -p 1814 \
-m disconnect \
-a Acct-Multi-Session-Id -v 01020304:05000000:00000000

```

In the preceding example, the **SessionControl.sh** command line executes a Disconnect-Request against the user “anonymous1234567890@demo.test” on the RADIUS client “TESTING-123”. The Calling-Station-Id is present in the disconnect message with the exact value that was present in the authentication request, even if it contains nulls.

Configuring Lawful-Intercept between SBR Carrier and ERX Device

During a lawful intercept request, the ERX device records the username as a case-sensitive string. SBR Carrier converts the username to an upper case format (all capitals) and saves it in the database. When a lawful intercept request is sent, the request is rejected by the ERX device because the username retrieved from the database is in upper case format.

To make the ERX device interoperate with SBR Carrier for lawful intercept, you need to configure the following procedures to retain the case sensitivity of the original username received from the ERX device:

1. Create a duplicate entry in the **radius.dic** file for storing the Original-User-Name.

Example:

```

ATTRIBUTE   Original-User-Name  1    string  r
ATTRIBUTE   User-Name           1    string  c
ATTRIBUTE   User-Password       2    string  c

```

2. Update the [AuthRequest] section of the **sessionTable.ini** file to map the username to the FunkOuterUserName attribute.

Example:

```

[AuthRequest]
FunkOuterUserName = User-Name

```

3. Map the FunkOuterUserName attribute to the Original-User-Name in the **dbc_mapping.xml** file.

Example:

```

<attributeMapping field="Sbr_NasIpv4Address"          attribute="NAS-IP-Address">
<queryAttribute name="NAS-IP-Address" />
</attributeMapping>
<attributeMapping field="FunkOuterUserName"          attribute="Original-User-Name">
</attributeMapping>
</dbcMapping>

```

4. Update the InterceptOn section in the **deviceModels.xml** file by setting the requiredAttribute as the Original-User-Name for the ERX device.

NOTE: In the case of InterceptOn, the following attributes need to be configured:

- Original-User-Name
- Acct-Session-Id
- Unisphere-Med-Ip-Address
- Unisphere-LI-Action

In the case of InterceptOff, the following attributes need to be configured:

- Acct-Session-Id
- Unisphere-LI-Action

The following is a sample **deviceModels.xml** file for ERX device:

```

</controlledDeviceModel>
<controlledDeviceModel id="Juniper-ERX 10.2" vendor="Juniper" model="Juniper-ERX
10.2" dictionary="juniper">
  <radiusPorts>
    <!--specifies default port -->
    <radiusPort name="RFC3576" port="1700"/>
  </radiusPorts>
  <actions>
    <action name="query">
      <localSessionQuery description="return local session data"/>
    </action>
    <action name="disconnect">
      <radiusRequest description="ERX Packet of Disconnect" code="DM" portName="RFC3576">
        <attributes>
          <requiredAttribute name="Acct-Session-Id"/>
        </attributes>
      </radiusRequest>
    </action>
  </actions>
</controlledDeviceModel>

```

```

    <onSuccess>
      <!--this device does not send Stop when you knock someone off -->
      <sessionStop description="Simulated Session Stop"/>
    </onSuccess>
    <onFailure>
      <!--assume bad session record -->
      <sessionStop description="Cleaning Session Database"/>
    </onFailure>
    <onTimeout/>
  </radiusRequest>
</action>
<action name="interceptOn" description="Mirror all IP traffic to specified device">
  <radiusRequest code="CoA" portName="RFC3576">
    <attributes>
      <requiredAttribute name="Original-User-Name"/>
      <requiredAttribute name="Acct-Session-Id"/>
      <requiredAttribute name="Unisphere-Med-Ip-Address"/>
      <overrideAttribute name="Unisphere-LI-Action" value="1"/>
    </attributes>
  </radiusRequest>
</action>
<action name="interceptOff" description="Stop mirroring IP traffic to specified
device">
  <radiusRequest code="CoA" portName="RFC3576">
    <attributes>
      <requiredAttribute name="Acct-Session-Id"/>
      <overrideAttribute name="Unisphere-LI-Action" value="0"/>
    </attributes>
  </radiusRequest>
</action>
</actions>
</controlledDeviceModel>

```

NOTE: You must to restart SBR Carrier to make these changes into effect.

11

PART

Statistics and Reporting

Displaying Statistics | 817

Logging and Reporting | 834

Displaying Statistics

IN THIS CHAPTER

- [Displaying Authentication Statistics | 817](#)
- [Displaying Accounting Statistics | 820](#)
- [Displaying Proxied Request Statistics | 823](#)
- [Displaying RADIUS Client Statistics | 825](#)
- [Displaying RADIUS Proxy Targets Statistics | 829](#)
- [Displaying IP Address Pool Statistics | 833](#)

You use the Web GUI to display summary statistics for authentication, accounting, and proxy forwarding transactions. You can also use the Web GUI to see how long SBR Carrier has been running and to display a list of the users currently connected through a NAD or tunnel. This chapter contains these topics:

NOTE: The statistics described are per second, and the statistics displays are updated every 10 seconds.

NOTE: Only the standard IETF RADIUS statistics are available from the Web GUI. To access SBR Carrier extended statistics, you must use other utilities. See [“Statistics Variables” on page 529](#).

Displaying Authentication Statistics

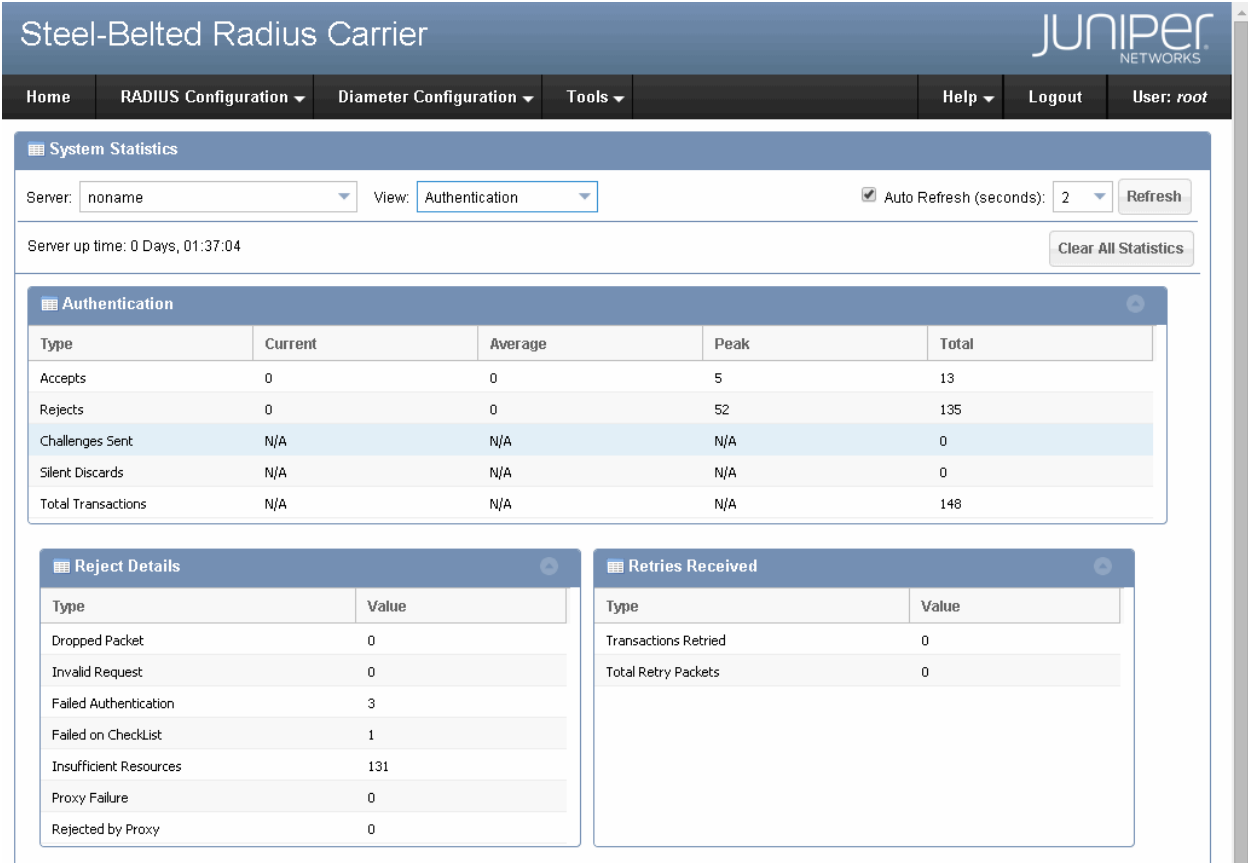
Authentication statistics summarize the number of authentication acceptances and rejections, with summary totals for each type of rejection or retry.

To display authentication statistics using the Web GUI:

1. Select **RADIUS Configuration > Statistics > System**.

The **System Statistics** page (Figure 258 on page 818) appears.

Figure 258: System Statistics Page—Authentication



2. Select the server entry for which you want to view the authentication statistics from the **Server** list and select **Authentication** from the **View** list.

The **System Statistics** page (Figure 258 on page 818) displays authentication statistics for the selected server.

Table 116 on page 819 explains the authentication statistics fields and describes possible causes for authentication rejections.

3. Optionally, select the **Auto Refresh (seconds)** check box and select a time period for refreshing the display from the drop-down list. Default value is 2 seconds.

By default, the auto-refresh feature is enabled.

You can click the **Refresh** button to refresh the display.

4. Optionally, click **Clear All Statistics** to clear all authentication statistics.

NOTE: SBR Carrier calculates the rate of authentication requests as they are received from the socket and displays all the incoming requests from the client in the Authentication Request statistics. Other statistics information such as Rejects are measured as the incoming requests are processed. If a request is not processed immediately and is handled by the flood queue, the processing is delayed and this is reflected in the Authentication statistics for that time period.

Table 116: Authentication Statistics

Authentication Statistic	Meaning
Authentication	
Accepts	The current, average, and peak number of RADIUS transactions that resulted in an accept response.
Rejects	The current, average, and peak number of RADIUS transactions that resulted in a reject response. These are detailed in the Reject Details section.
Challenges Sent	The number of challenges sent.
Silent Discards	The number of discarded authentication requests. Requests can be discarded due to unknown clients or other factors, such as inability to assign resources, if so configured. The clients cannot not be identified if a RADIUS client entry cannot be found for a device with the name or IP address of a device requesting authentication services.
Total Transactions	The sum of accept, reject, and silent discard totals.
Reject Details	
Dropped Packet	The number of RADIUS authentication packets dropped by Steel-Belted Radius Carrier because the server was flooded with more packets than it was able to handle.
Invalid Request	<p>The number of invalid RADIUS requests made.</p> <p>Invalid requests can be a result of a device sending incorrectly formed packets to Steel-Belted Radius Carrier because it is configured incorrectly, or because the device does not conform to the RADIUS standard.</p>

Table 116: Authentication Statistics *(continued)*

Authentication Statistic	Meaning
Failed Authentication	<p>The number of failed authentication requests, where the failure is due to invalid username or password.</p> <p>If all transactions are failing authentication, the problem might be that the shared secret entered into Steel-Belted Radius Carrier does not match the shared secret entered on the client device.</p>
Failed on CheckList	The number of requests that were authenticated but failed to meet the check list requirements.
Insufficient Resources	The number of rejects due to a server resource problem.
Proxy Failure	The number of rejects that had to be issued because Proxy forwarding to another RADIUS server failed.
Rejected by Proxy	The number of rejects due to receiving a reject response from a proxy RADIUS target server.
Retries Received	
Transactions Retried	The number of requests for which one or more duplicates was received.
Total Retry Packets	The number of duplicate packets received.

Displaying Accounting Statistics

Accounting statistics provide information such as the number of transaction starts and stops and the reasons for rejecting attempted transactions. The start and stop numbers rarely match, because many transactions can be in progress at any given time.

To display accounting statistics using the Web GUI:

1. Select **RADIUS Configuration > Statistics > System**.

The **System Statistics** page (Figure 258 on page 818) appears.

2. Select the server entry for which you want to view the accounting statistics from the **Server** list and select **Accounting** from the **View** list.

The **System Statistics** page (Figure 259 on page 821) displays accounting statistics for the selected server.

Figure 259: System Statistics Page—Accounting

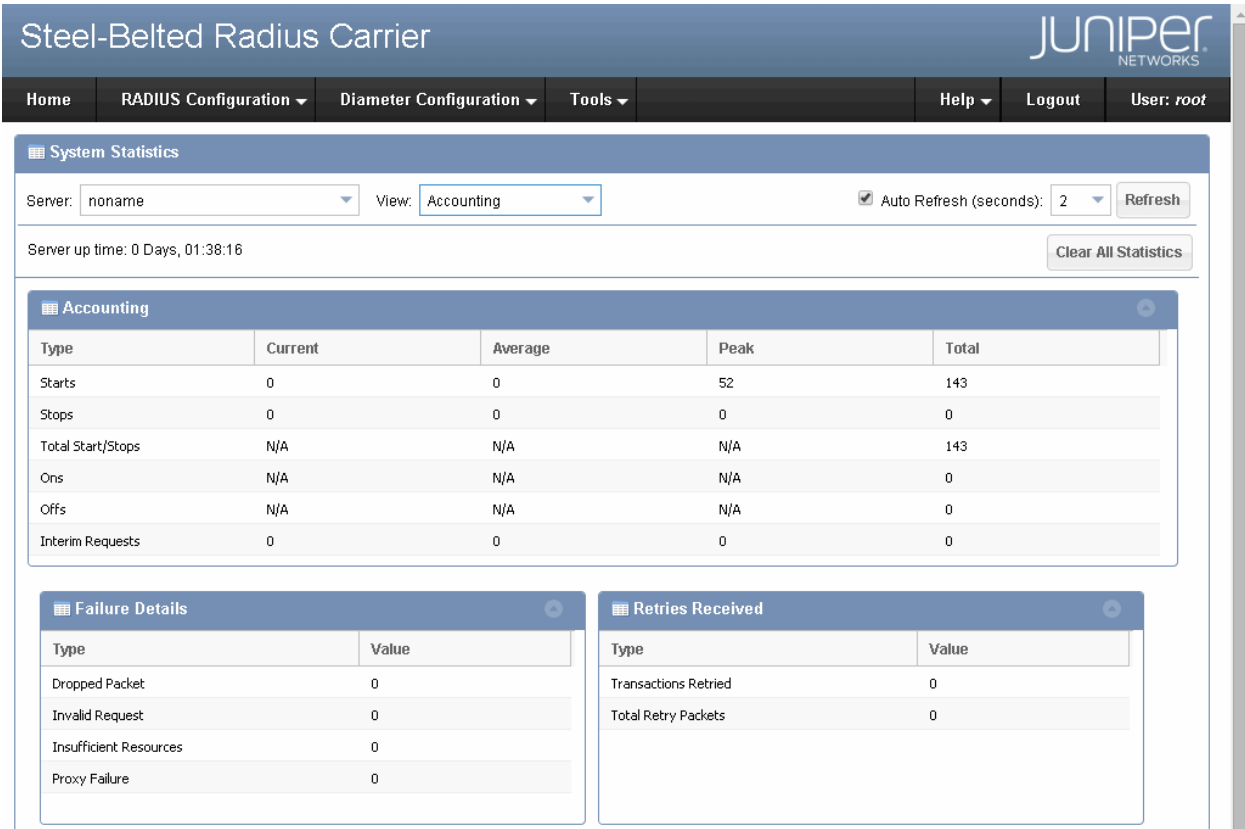


Table 117 on page 822 describes the accounting statistics fields and describes the possible causes for accounting errors.

3. Optionally, select the **Auto Refresh (seconds)** check box and select a time period for refreshing the display from the drop-down list. Default value is 2 seconds.

By default, the auto-refresh feature is enabled.

You can click the **Refresh** button to refresh the display.

4. Optionally, click **Clear All Statistics** to clear all accounting statistics.

NOTE: SBR Carrier calculates the rate of accounting requests as they are received from the socket and displays all the incoming requests from the client in Accounting Request statistics. Other statistics information such as Rejects are measured as the incoming requests are processed. If a request is not processed immediately and is handled by the flood queue, the processing will be delayed and this will be reflected in the Accounting statistics for that time period.

Table 117: Accounting Statistics

Statistic	Meaning
Accounting	
Starts	The current, average, and peak number of transactions in which a dial-in connection was started following a successful authentication.
Stops	The current, average, and peak number of transactions in which a dial-in connection was terminated.
Total Start/Stops	The sum of the start, stop, on and off totals.
Ons	The number of Accounting-On messages received, indicating that a RADIUS client has restarted.
Offs	The number of Accounting-Off messages received, indicating that a RADIUS client has shut down.
Interim Requests	The number of interim accounting packets received.
Failure Details	
Dropped Packet	The number of RADIUS accounting packets dropped by Steel-Belted Radius Carrier because the server was flooded with more packets than it was able to handle.
Invalid Request	<p>The number of invalid RADIUS requests made.</p> <p>Invalid requests can be a result of a device sending incorrectly formed packets to Steel-Belted Radius Carrier because it is incorrectly configured, or because the device does not conform to the RADIUS standard.</p>

Table 117: Accounting Statistics (continued)

Statistic	Meaning
Insufficient Resources	The number of requests discarded due to a server resource problem.
Proxy Failure	The number of times that proxy RADIUS forwarding failed.
Retries Received	
Transactions Retried	The number of requests for which one or more duplicates was received.
Total Retry Packets	The number of duplicate packets received.

Displaying Proxied Request Statistics

Proxied request statistics provide information such as the number of proxy authentication or accounting requests and the reasons for any transaction failures that occur.

To display proxied request statistics using the Web GUI:

1. Select **RADIUS Configuration > Statistics > System**.

The **System Statistics** page ([Figure 258 on page 818](#)) appears.

2. Select the server entry for which you want to view the proxied request statistics from the **Server** list and select **Proxied Requests** from the **View** list.

The **System Statistics** page ([Figure 260 on page 824](#)) displays proxied request statistics for the selected server.

Figure 260: System Statistics Page—Proxied Requests

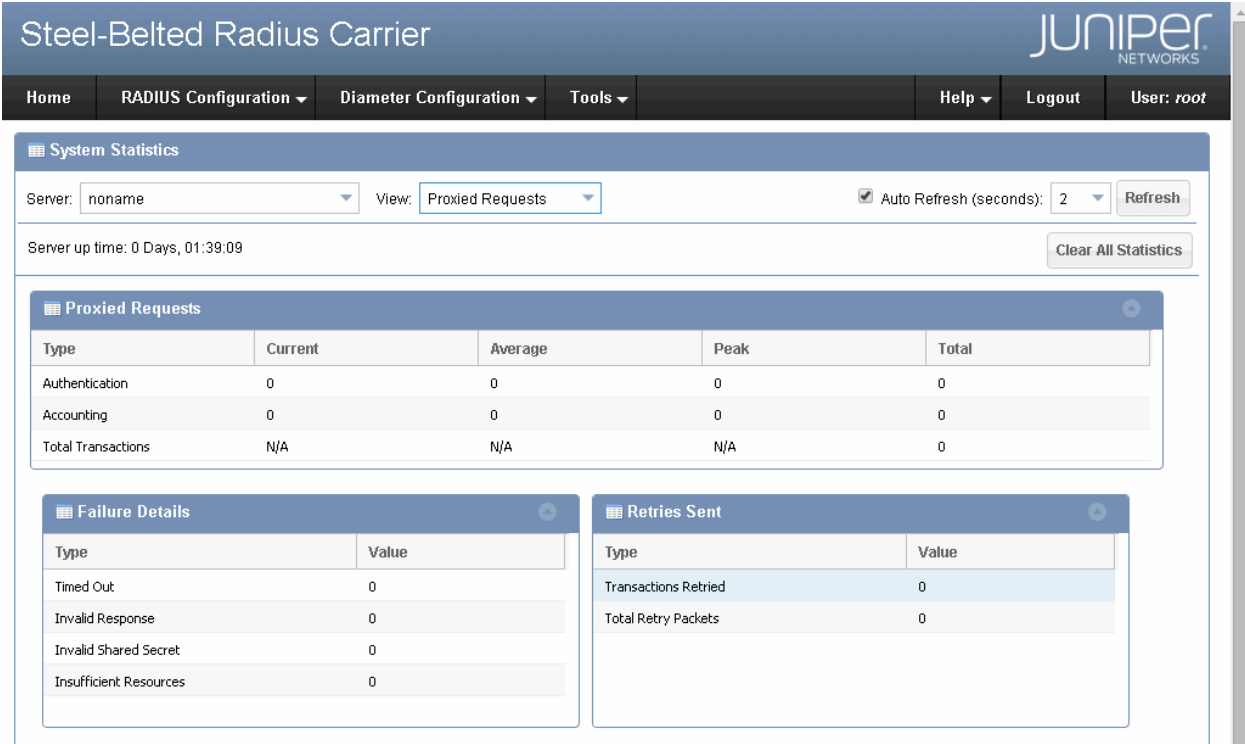


Table 118 on page 824 describes the proxied request statistics.

- Optionally, select the **Auto Refresh (seconds)** check box and select a time period for refreshing the display from the drop-down list. Default value is 2 seconds.

By default, the auto-refresh feature is enabled.

You can click the **Refresh** button to refresh the display.

- Optionally, click **Clear All Statistics** to clear all proxied request statistics.

Table 118: Proxy Statistics

Proxy Statistic	Meaning
Proxied Requests	
Authentication	The number of authentication transactions between the proxy and target RADIUS servers.
Accounting	The number of accounting transactions between the proxy and target RADIUS servers.
Total Transactions	The sum of the authentication and accounting transaction totals.

Table 118: Proxy Statistics *(continued)*

Proxy Statistic	Meaning
Failure Details	
Timed Out	The number of RADIUS transactions that timed out, even after all retry attempts were made.
Invalid Response	<p>The number of invalid RADIUS responses received.</p> <p>A target is sending incorrectly formed packets to Steel-Belted Radius Carrier; there is a configuration error, the target RADIUS server does not conform to the RADIUS standard, or Steel-Belted Radius Carrier did not receive a proxy state echo in the received packet.</p>
Invalid Shared Secret	<p>The number of packets for which an incorrect digital signature was received.</p> <p>The shared secret does not match between Steel-Belted Radius Carrier and the target; or some unauthorized rogue device is attempting to compromise RADIUS security.</p>
Insufficient Resources	The number of rejects or dropped packets due to a server resource problem.
Retries Sent	
Transactions Retried	The number of requests for which one or more retried transmissions was performed.
Total Retry Packets	The number of duplicate packets received.

Displaying RADIUS Client Statistics

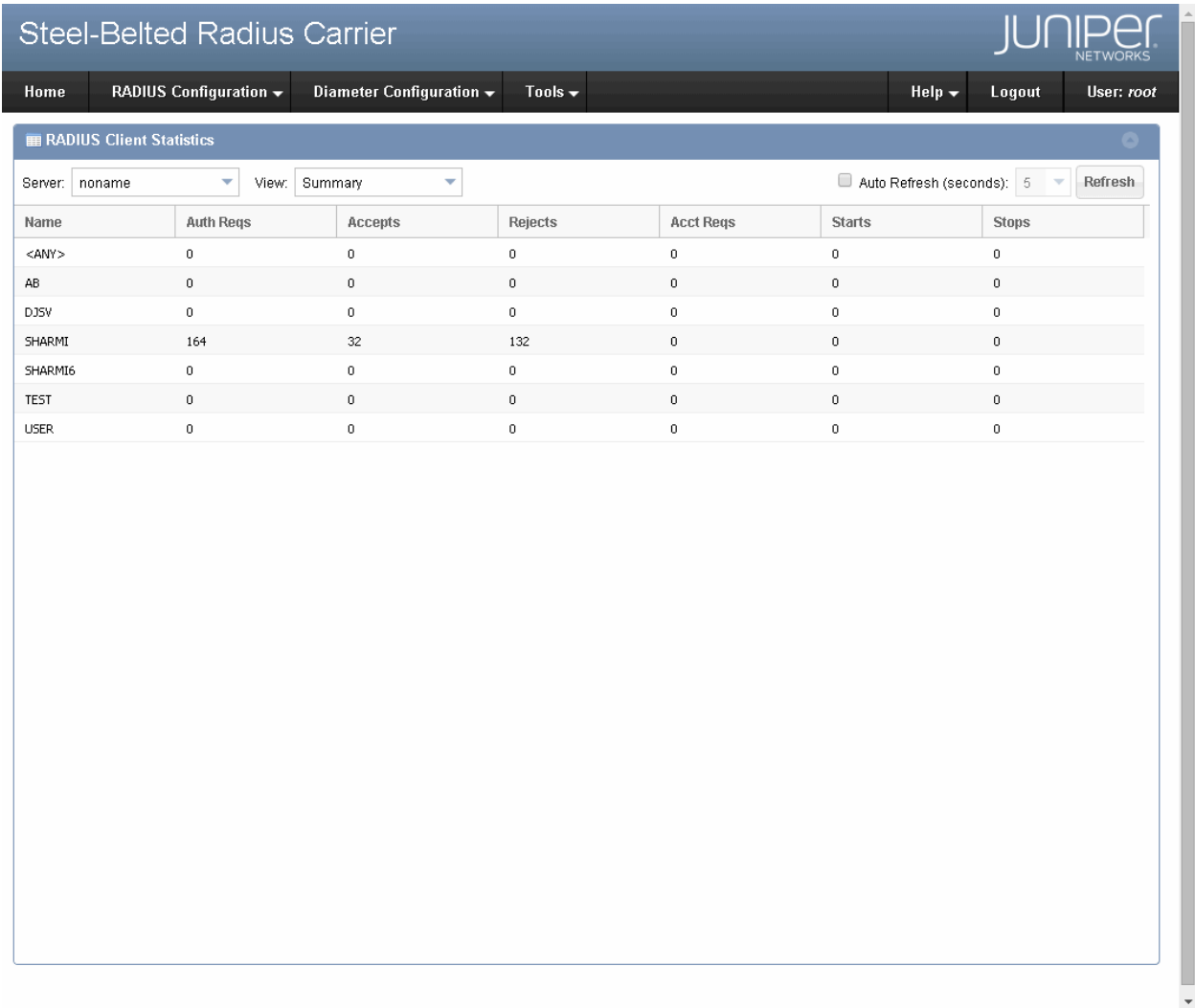
RADIUS client statistics provide information about the number of authentication and accounting requests by client.

To display statistics for RADIUS clients using the Web GUI:

1. Select **RADIUS Configuration > Statistics > RADIUS Clients**.

The **RADIUS Client Statistics** page (Figure 261 on page 826) appears.

Figure 261: RADIUS Client Statistics Page—Summary

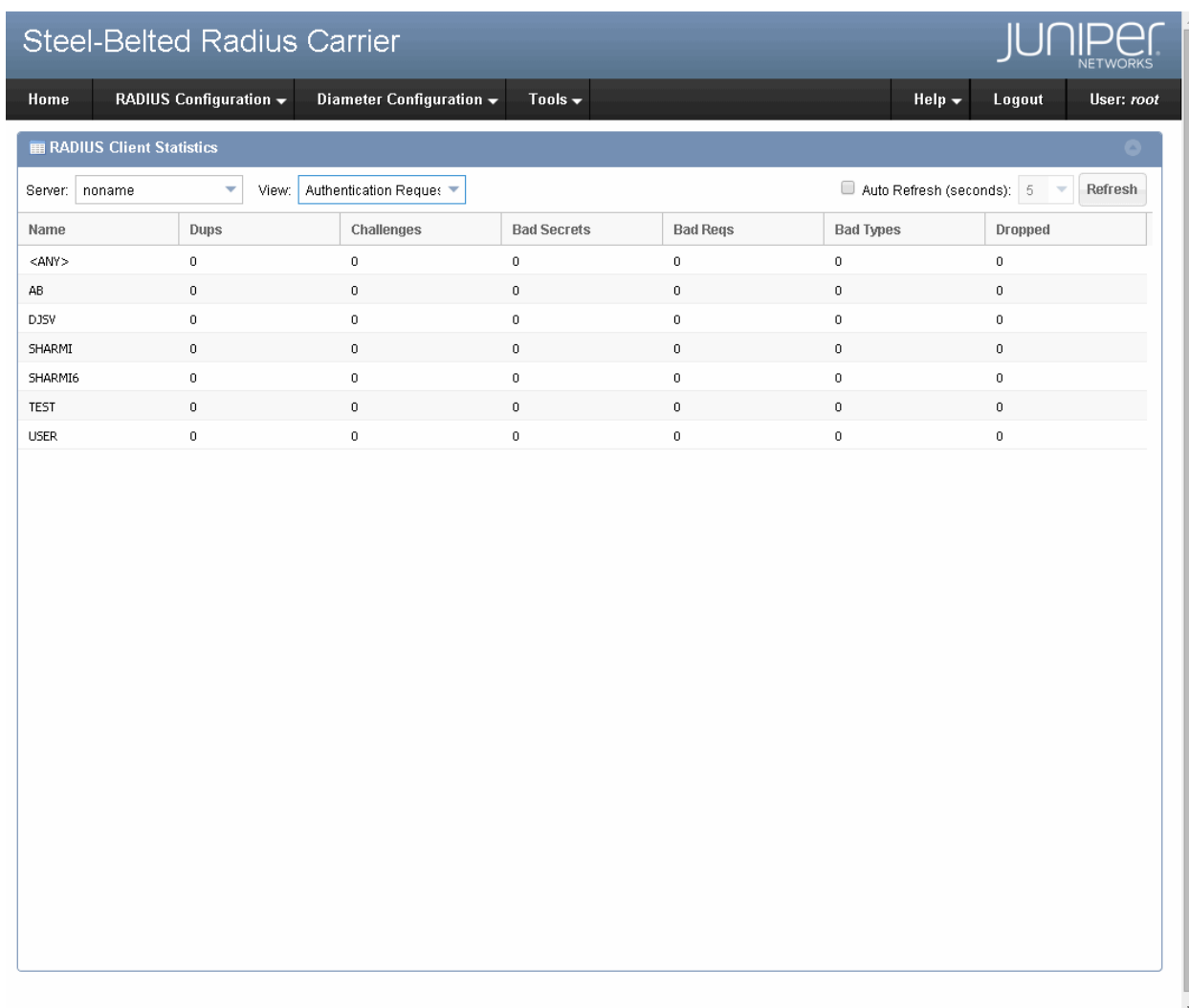


2. Select the server entry for which you want to view statistics for RADIUS clients from the **Server** list.

3. Use the **View** list to display the type of statistics you want to display.

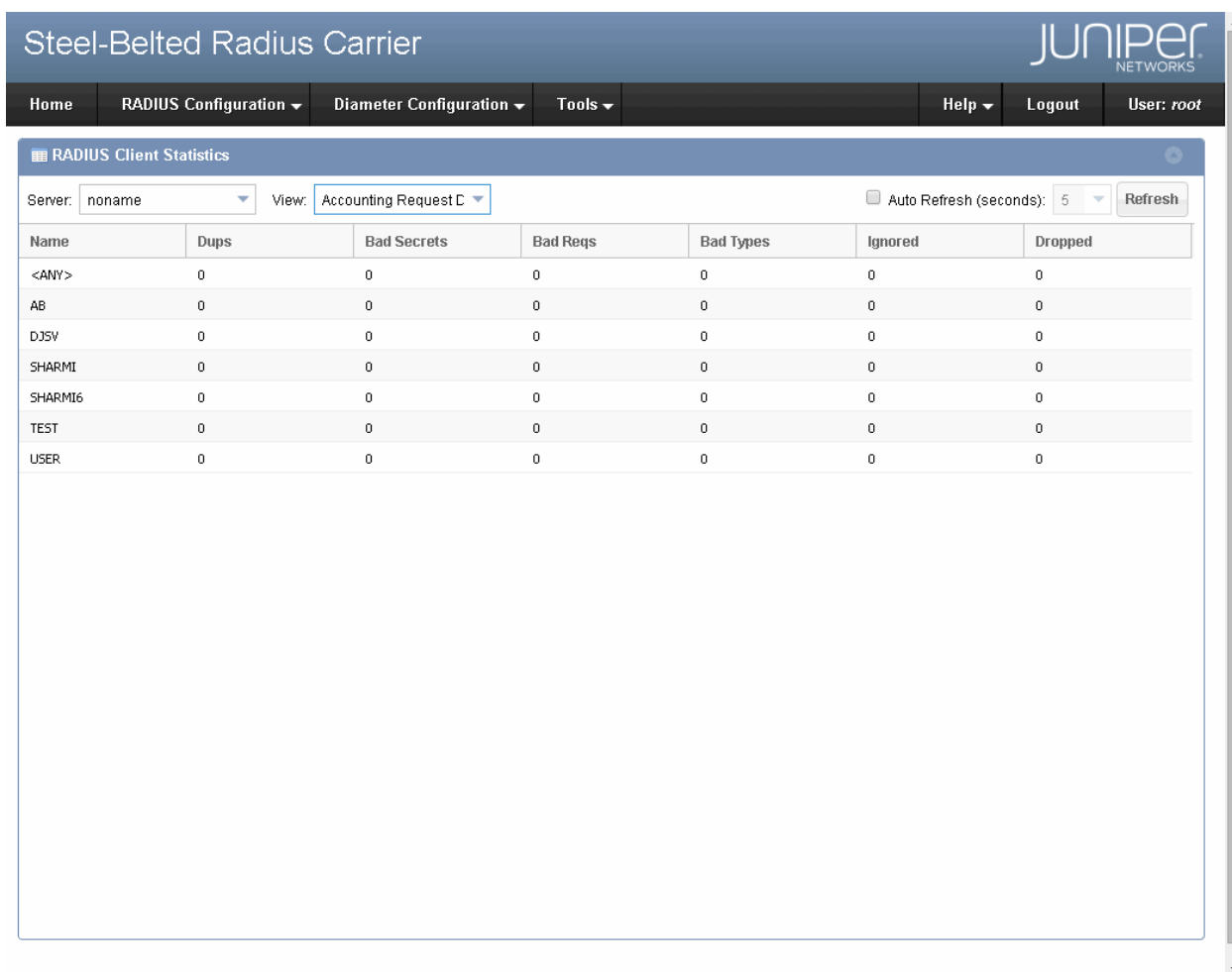
- **Summary** (Figure 261 on page 826)—Displays the number of authentication requests, Access-Accepts, and Accept-Reject messages and the total number of accounting requests, starts, and stops for each RADIUS client.
- **Authentication Request Diagnostics** (Figure 262 on page 827)—Displays the number of duplicate messages, challenges, messages containing invalid authentication information, bad authentication requests, bad types, and dropped requests for each RADIUS client.

Figure 262: RADIUS Client Statistics Page—Authentication Request Diagnostics



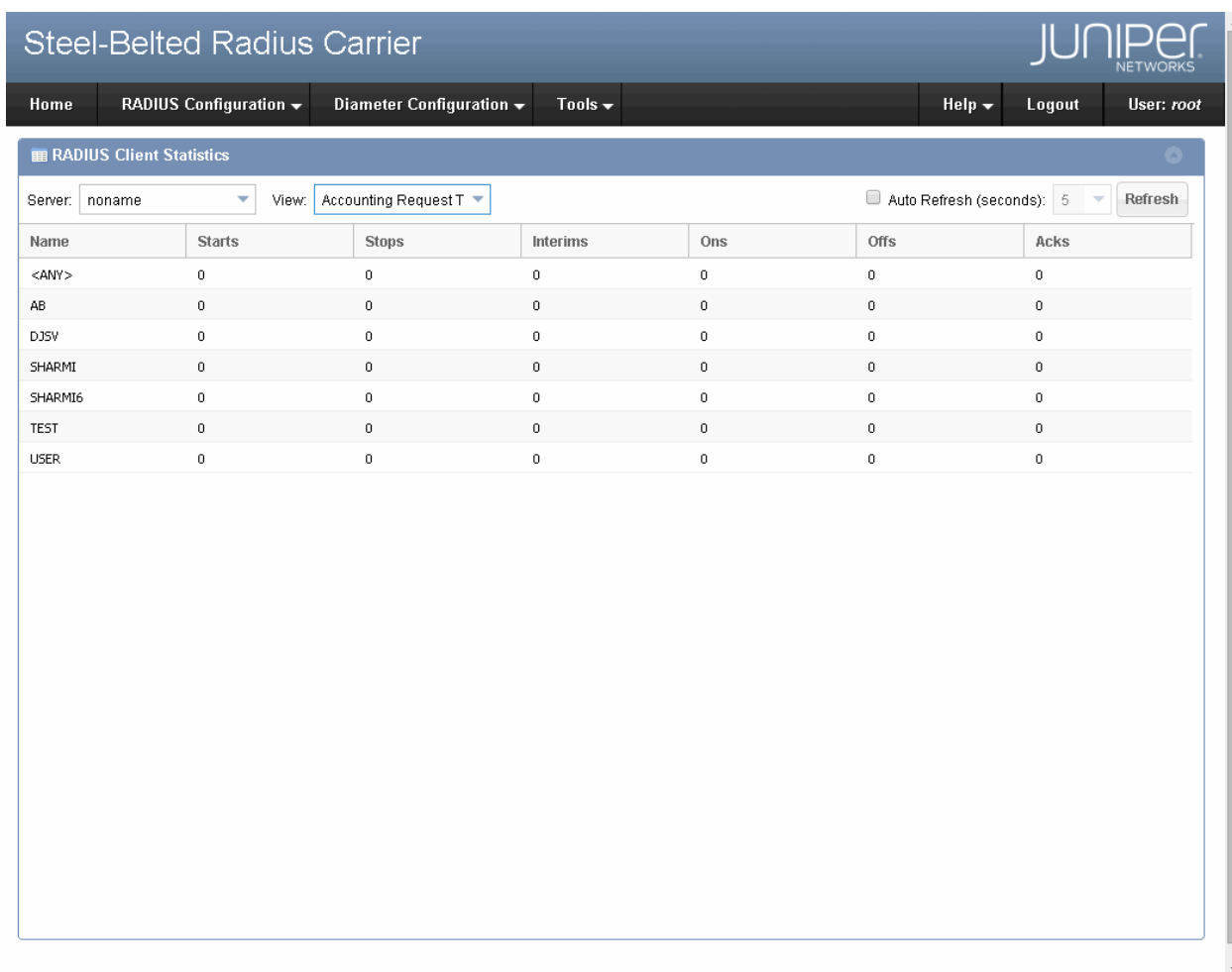
- **Accounting Request Diagnostics** (Figure 263 on page 828)—Displays the number of duplicate messages, messages with invalid secrets, malformed messages, messages with incorrect types, ignored messages, and dropped requests for each RADIUS client.

Figure 263: RADIUS Client Statistics Page—Accounting Request Diagnostics



- **Accounting Request Types** (Figure 264 on page 829)—Displays the number of accounting start messages, accounting stop messages, interim messages, Accounting-On messages, Accounting-Off messages, and acknowledgement messages sent for each RADIUS client.

Figure 264: RADIUS Client Statistics Page—Accounting Request Types



- Optionally, select the **Auto Refresh (seconds)** check box and select a time period for refreshing the display from the drop-down list. Default value is 5 seconds.

By default, the auto-refresh feature is disabled.

You can click the **Refresh** button to refresh the display.

Displaying RADIUS Proxy Targets Statistics

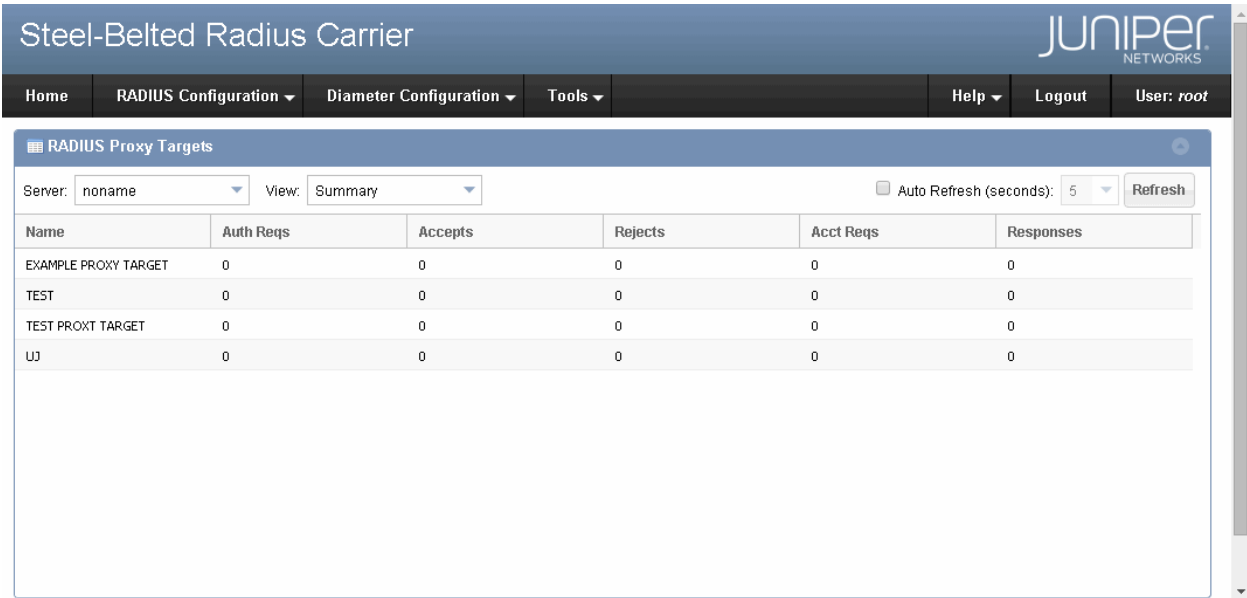
RADIUS proxy target statistics provide information about the number of authentication and accounting transactions associated with each proxy target.

To display statistics for RADIUS proxy targets using the Web GUI:

1. Select **RADIUS Configuration > Statistics > RADIUS Proxy Targets**.

The **RADIUS Proxy Targets** page (Figure 265 on page 830) appears.

Figure 265: RADIUS Proxy Targets Page—Summary



2. Select the server entry for which you want to view statistics for RADIUS proxy targets from the **Server** list.
3. Use the **View** list to display the type of statistics you want to display.
 - **Summary** (Figure 265 on page 830)—Displays the number of authentication requests, accepts and reject messages, and the number of accounting requests and responses for each RADIUS proxy target.
 - **Authentication Request Details** (Figure 266 on page 831)—Displays the number of outstanding messages, retransmitted messages, and challenges, along with the most recent response time for the proxy target.

Figure 266: RADIUS Proxy Targets Page—Authentication Request Details

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

RADIUS Proxy Targets

Server: noname View: Authentication Request Auto Refresh (seconds): 5 Refresh

Name	Outstanding	Retransmitted	Challenges	Latest Response (ms)
EXAMPLE PROXY TARGET	0	0	0	0
TEST	0	0	0	0
TEST PROXT TARGET	0	0	0	0
UJ	0	0	0	0

- **Authentication Request Diagnostics** (Figure 267 on page 831)—Displays the number of timeouts, invalid secrets, incorrect requests, requests with invalid types, and dropped messages for each proxy target.

Figure 267: RADIUS Proxy Targets Page—Authentication Request Diagnostics

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

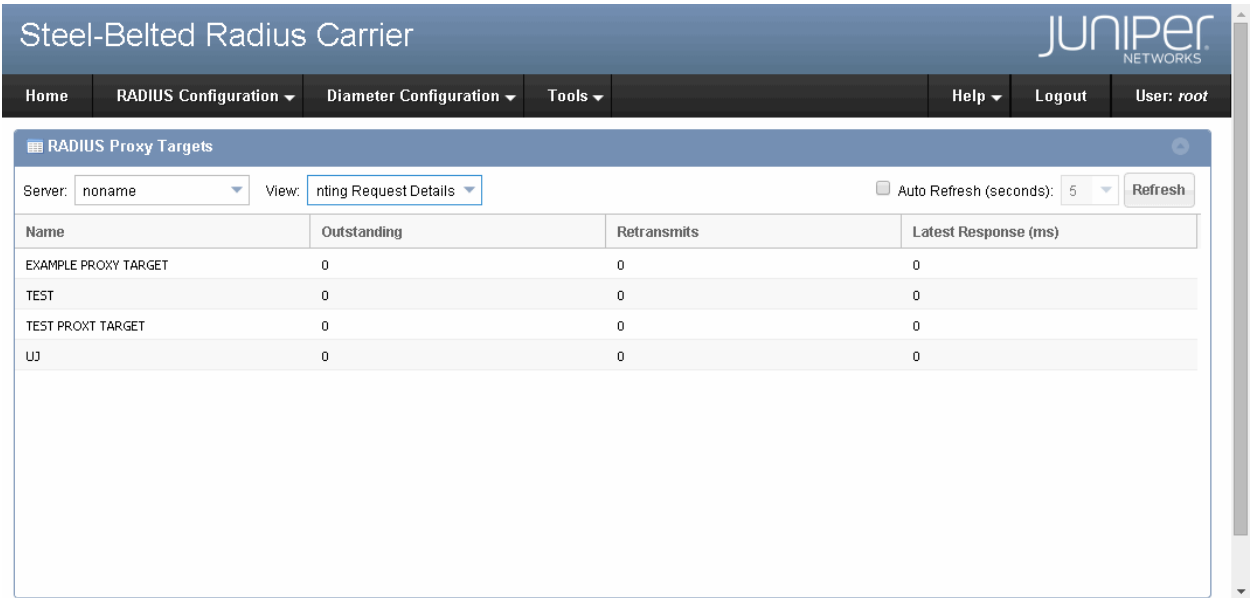
RADIUS Proxy Targets

Server: noname View: Authentication Request Auto Refresh (seconds): 5 Refresh

Name	Timeouts	Bad Secrets	Bad Reqs	Bad Types	Dropped
EXAMPLE PROXY TARGET	0	0	0	0	0
TEST	0	0	0	0	0
TEST PROXT TARGET	0	0	0	0	0
UJ	0	0	0	0	0

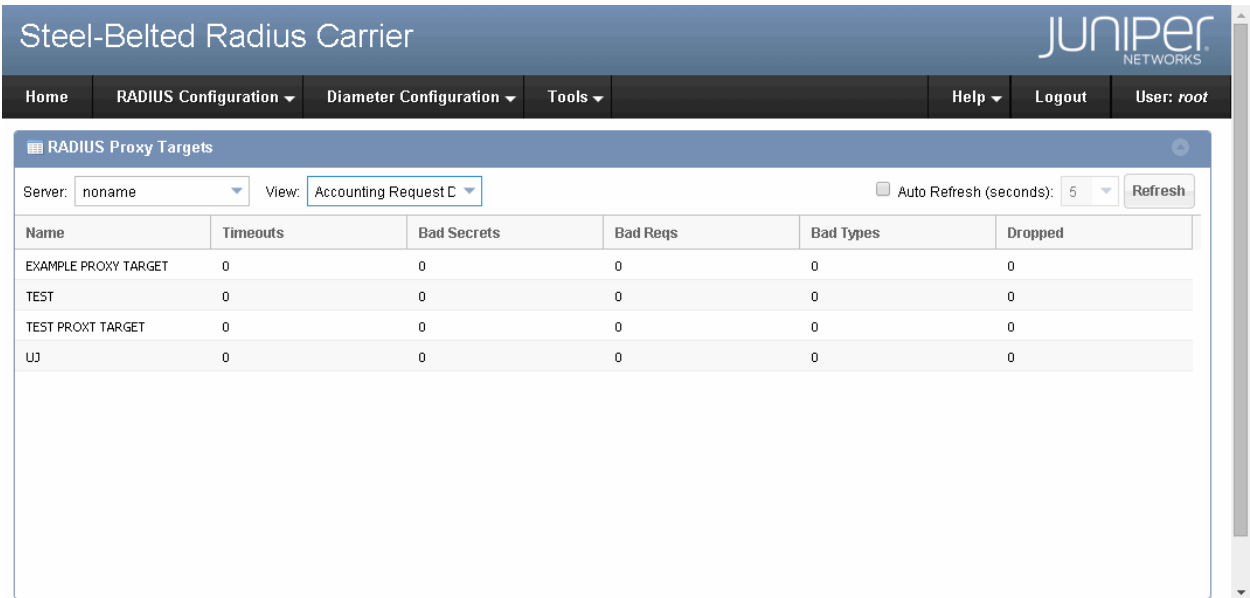
- **Accounting Request Details** (Figure 268 on page 832)—Displays the number of outstanding messages and retransmitted messages, along with the most recent response time for the proxy target.

Figure 268: RADIUS Proxy Targets Page—Accounting Request Details



- **Accounting Request Diagnostics** (Figure 269 on page 832)—Displays the number of timeouts, invalid secrets, incorrect requests, requests with invalid types, and dropped messages for each proxy target.

Figure 269: RADIUS Proxy Targets Page—Accounting Request Diagnostics



4. Optionally, select the **Auto Refresh (seconds)** check box and select a time period for refreshing the display from the drop-down list. Default value is 5 seconds.
- By default, the auto-refresh feature is disabled.
- You can click the **Refresh** button to refresh the display.

Displaying IP Address Pool Statistics

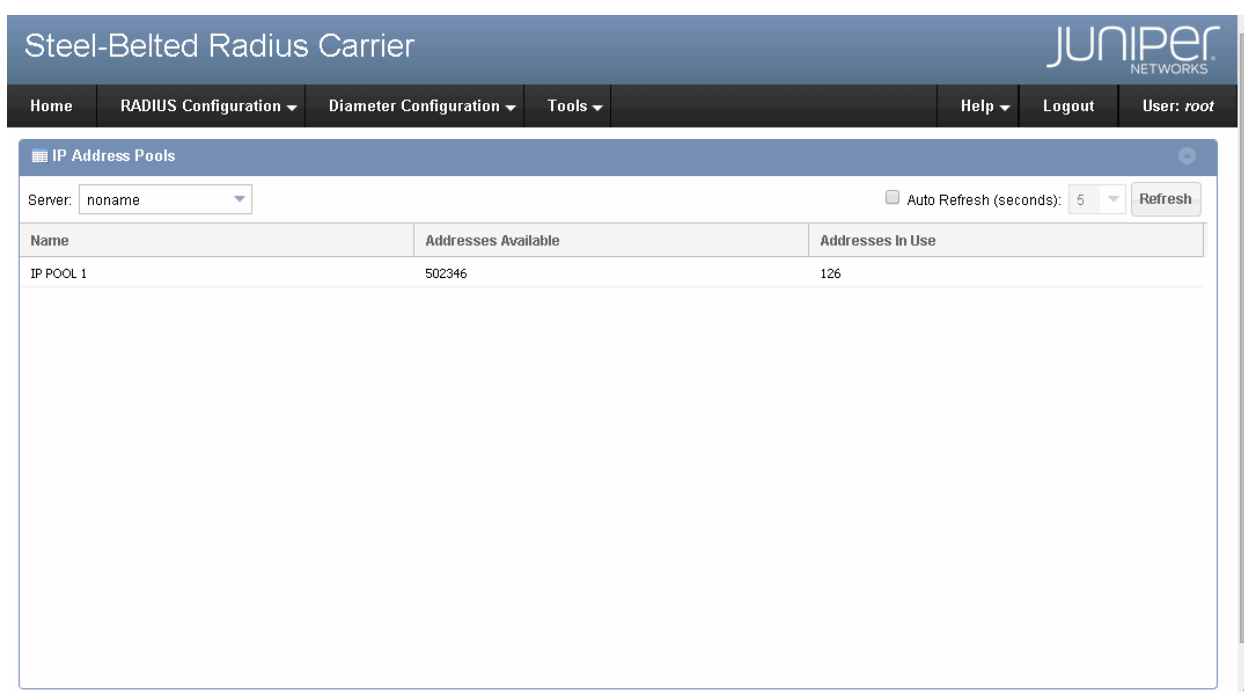
IP address pool statistics provide a summary of the number of addresses allocated from each IPv4 address pool and how many addresses remain available.

To display IP address pool statistics using the Web GUI:

1. Select **RADIUS Configuration > Statistics > IP Address Pools**.

The **IP Address Pools** page ([Figure 270 on page 833](#)) appears.

Figure 270: IP Address Pools Page



2. Select the server entry for which you want to view the IP address pool statistics from the **Server** list.
3. Optionally, select the **Auto Refresh (seconds)** check box and select a time period for refreshing the display from the drop-down list. Default value is 5 seconds.

By default, the auto-refresh feature is disabled.

You can click the **Refresh** button to refresh the display.

Logging and Reporting

IN THIS CHAPTER

- [Logging Files | 834](#)
- [Displaying Authentication Log Files | 835](#)
- [Using the Locked Accounts List | 846](#)
- [Configuring the Log Retention Period | 848](#)
- [Using the Server Log File | 850](#)
- [Using the Authentication Log File | 852](#)
- [Using the Accounting Log File | 854](#)

This chapter describes how to set up and use logging and reporting functions in SBR Carrier. All logging and reporting are local only, not global. This chapter contains these topics:

Logging Files

The files listed in [Table 119 on page 834](#) establish settings for logging and reporting. For more information about these files, refer to the *SBR Carrier Reference Guide*.

Table 119: Logging and Reporting Files

File Name	Function
account.ini	Controls how RADIUS accounting attributes are logged.
authlog.ini	Controls how RADIUS authentication requests are logged by SBR Carrier.
authReport.ini	Controls what authentication logs SBR Carrier generates.
authReportAccept.ini	Controls options for the acceptance authentication log file.

Table 119: Logging and Reporting Files *(continued)*

File Name	Function
authReportBadSharedSecret.ini	Controls options for the invalid shared secret authentication log file.
authReportReject.ini	Controls options for the rejection authentication log file.
authReportUnknownClient.ini	Controls options for the unknown client authentication log file.
events.ini	Configures dilution, suppression, and threshold settings for SBR Carrier traps (except for Diameter-related traps) used to communicate failures, warnings, and other information.
events.xml	Configures dilution and suppression settings for Diameter-related traps used to communicate failures, warnings, and other information.
radius.ini	Controls (among other things) the types of messages SBR Carrier records in the server log file and the location of the log directory.

NOTE: Startup and system log files capture the version information of the plug-ins used in SBR Carrier. If the plug-ins do not contain version information, the log files record all information, without the version information, about these plug-ins.

Displaying Authentication Log Files

You use the Web GUI to enable and display these authentication log files:

- **Successful request authentication log file**—The successful request authentication log file identifies the authentication requests that were approved by SBR Carrier.
- **Invalid shared secrets authentication log file**—The invalid shared secrets authentication log file identifies the authentication requests that failed because a known RADIUS client supplied an incorrect shared secret. This condition is detectable only if the authentication request contains a Message-Authenticator attribute, which is required if credentials are of an EAP type but optional if credentials are PAP, CHAP, or MS-CHAP.

- Failed request authentication log file—The failed request authentication log file identifies the authentication requests that were rejected because the user supplied incorrect credentials.
- Unknown client request authentication log file—The unknown client request authentication log file identifies authentication requests received from unknown RADIUS clients.

File Permissions for Log Files

When you run SBR Carrier, you can specify which users are authorized to read or edit important files, such as authentication and accounting log files. For example, you can specify that system administrators who install and configure SBR Carrier have read/write access for system log files and that network operators who monitor SBR Carrier have read-only (or no) access for system log files.

Security Groups and Permissions

Each file and directory has three security groups associated with it:

- The Owner identifies the person who created or owns the file.
- The Group security group identifies the set of users who are members of the group or groups to which the file Owner belongs. Group members can exercise special privileges with respect to that file. A user can belong to more than one group.
- The Other security group consists of the set of all users who do not belong to Owner or Group.

Each security group has three flags that control what privileges that group can exercise with respect to the file or directory:

- The Read flag (r) determines whether the file can be read. The Read flag has an octal value of 4.
- The Write flag (w) determines whether the security group can create, modify, or delete the file. The Write flag has an octal value of 2.
- The Execute flag (x) determines whether the security group can run a script or executable file. The Execute flag has an octal value of 1.

For example, a file owner might have **rw**x permission for a file, which indicates the file owner has read/write/execute access to the file. Similarly, Other might have **r--** permission (where - indicates no permission), which means that the user can read but not edit or execute the file.

You can add the octal values for permission flags to generate a numeric representation of the file permissions for Owner, Group, and Other:

- 1 = execute only
- 2 = write only
- 3 = write and execute (1+2)
- 4 = read only

- 5 = read and execute (4+1)
- 6 = read and write (4+2)
- 7 = read,write and execute (4+2+1)

The security permissions exercised by Owner/Group/Other are typically expressed as string or a three-digit number. [Table 120 on page 837](#) provides examples of different file permissions.

Table 120: File Permissions

Permission	Octal value	What It Means
-rwxrwxrwx	777	Read, write, and executable for Owner/Group/Other
-rw-rw-r--	664	Read and write for Owner/Group; read access for Other
-rw-rw----	660	Read and write for Owner/Group; no access for Other
-rwx-----	700	Read, write, and executable for owner only
-rw-rw-rw	666	Read and write for owner, group, and all others

The UNIX **chown** command lets you change the owner or group (or both) associated with a file or directory. The UNIX **chmod** command lets you change the permissions of files and directories.

Using the User File Creation Mode Mask

The user file mode creation mode mask (often abbreviated as **umask**) determines the default file system mode for newly created files of the current process. Solaris hosts typically have a hierarchy of **umask** values: a server-level **umask** value, which can be overridden by a user-, shell-, or application-level **umask** value. The result is an *ambient umask value*, which determines what file permissions are used when files are created by any given process.

The **umask** value is a three-digit octal number. The first digit sets the mask for Owner, the second for Group, and the third for Other. The **umask** value identifies the permissions that are *withheld* when a file is created: the **umask** value is subtracted from the full access mode value (777) to determine the access permissions for a new file. For example, if the **umask** value for a process is set to 022, the Write permission for Group and Other are withheld from the full access mode value (777), resulting in a file permission of 755 (**rw-r-xr-x**). Similarly, if the **umask** value of 177 is configured for a process (explicitly or by virtue of the ambient **umask**), files created by the process have a file permission of 600 (**rw-----**).

[Table 121 on page 837](#) summarizes the result of using different octal numbers in an **umask** value.

Table 121: Summary of Umask Permissions

Octal Number	Access	Permission Resulting From umask Value
0	rwx	Read, Write, Execute

Table 121: Summary of Umask Permissions (*continued*)

Octal Number	Access	Permission Resulting From umask Value
1	rw-	Read, Write
2	r-x	Read, Execute only
3	r--	Read only
4	-wx	Write, Execute only
5	-w-	Write only
6	--x	Execute only
7	---	No permissions

The **umask** value affects a file's access permissions only when the file is created. If you change the **umask** value, access permissions for existing files are not affected. Similarly, you can use the **chown** and **chmod** commands to change a file's access permissions after the file has been created.

Implementing Default File Permissions in SBR Carrier

The **RADIUSMASK** parameter in the **sbrd.conf** file specifies the application-level **umask** value used to establish access permissions for all files created by SBR Carrier. Refer to the *SBR Carrier Reference Guide* for information about configuring the **sbrd.conf** file.

If you do not specify a value for the **RADIUSMASK** parameter, SBR Carrier uses the ambient **umask** value established by the server-, user- or shell-level **umask** value to determine the access permissions for files it creates.

Some log files have explicit controls that let you override the **umask** value established by the **RADIUSMASK** parameter or the ambient **umask** value. See [“Implementing Override File Permissions in SBR Carrier” on page 839](#) for more information about overriding the application-level default **umask** value.

As previously noted, the **umask** value affects a file's access permissions only when the file is created. If you change the **RADIUSMASK** setting, new files created by SBR Carrier are assigned the access permission specified by the new setting. This includes files that roll over periodically; the existing file would retain the access file permission it received when it was created, and the new file would be assigned the access permission specified by the new **RADIUSMASK** value.

NOTE: The Execute file permission value for files created by SBR Carrier is always set to None for Owner, Group, and Other. Thus, an **umask** value of 0 (no restrictions) is equivalent to an umask value of 1 (read/write permission) for files created by SBR Carrier.

Implementing Override File Permissions in SBR Carrier

To override file permissions established by the SBR Carrier RADIUSMASK or the ambient **umask** for specific log files, you must modify the **LogFilePermissions** parameter in the applicable initialization (.ini) file.

Table 122 on page 839 identifies the configuration file you must modify to configure non-default file permissions for SBR Carrier log files.

Table 122: Configuration Files for Setting Log File Permissions

Controlled Files	Configuration File
Server Diagnostics log (server log)	radius.ini
Authentication Reporting Library accepts log	authReportAccept.ini
Authentication Reporting Library bad shared secret log	authReportBadSharedSecret.ini
Authentication Reporting Library rejects log	authReportReject.ini
Authentication Reporting Library unknown client log	authReportUnknownClient.ini
Authentication Logging Library logs and header check-point logs	authlog.ini
Accounting Library logs and header check-point logs	account.ini
Server Statistics logs and header check-point logs	statlog.ini

The syntax for the **LogFilePermissions** parameter is:

LogfilePermissions = **owner:group mode**

- Specify the **owner** and **group** settings by entering character strings or decimal integers, as used for arguments to the UNIX **chown(1)** command. For example, **ralphw:proj**, **ralphw:120**, or **1007:120**.
- Specify the **mode** setting as a character string or an octal integer. When permissions are specified as a character string, they follow the format that is used by the UNIX **ls(1)** command; for example, **rw-rw-rw-**.

When permissions are specified as an octal integer, they follow the format used for arguments to the UNIX **chmod(1)** command; for example, **666**.

NOTE: You can specify only read/write permissions for a SBR Carrier file. You cannot specify execute permissions for SBR Carrier files.

The value of each **LogFilePermissions** parameter is read when the SBR Carrier server is started or restarted. The value of the **LogFilePermissions** parameter in the **radius.ini** file is also read when you issue a HUP command to the SBR Carrier server.

- If you enter a valid value for a **LogfilePermissions** parameter, the ownership and permissions of the controlled log file are set as specified whenever the file is opened or created.
- If you do not enter a value for a **LogfilePermissions** parameter, the ownership and permissions of the controlled file are not changed. The controlled file is created using the ownership of the account that is executing the server and the permissions that are derived from the default **RADIUSMASK** value or from the ambient **umask** setting. If the file already exists, new information is appended without changing the existing ownership and permissions of the controlled file.
- If you enter an invalid value for a **LogfilePermissions** setting, then the ownership of the controlled log file defaults to the effective user/group ID of the server process, normally **root:other**, and the permissions for the controlled file default to **0600 (-rw-----)**. This ensures that the affected log file can always be opened without any escalation of file access privileges. Messages similar to the following are logged whenever an explicit file access control is misconfigured:

```
Invalid LogfilePermissions specified in radius.ini [Configuration]: -rwx-----
Server log file permissions defaulted to 0:0 0600
```

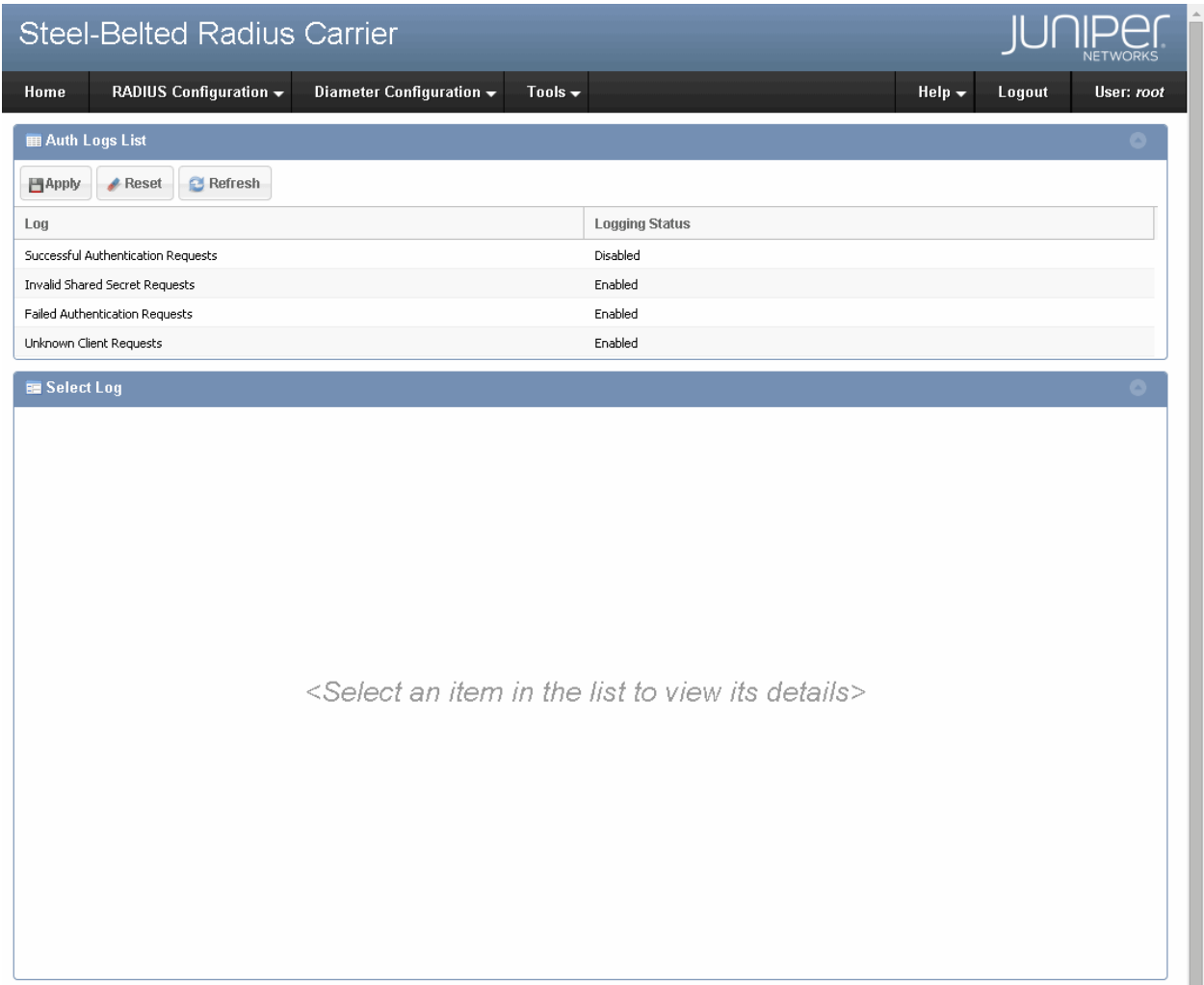
Enabling or Disabling the Authentication Log Files

To enable or disable an authentication log file using the Web GUI:

- 1. Select **RADIUS Configuration > Reports > Auth Logs**.

The **Auth Logs List** page (Figure 271 on page 841) appears.

Figure 271: Auth Logs List Page



- 2. Select the logging status column of the authentication log file you want to enable or disable.

A check box (Figure 272 on page 842) appears in the logging status column of the selected log file entry.

Figure 272: Enabling or Disabling the Authentication Log File

The screenshot shows the Steel-Belted Radius Carrier Web GUI. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and the user root. The main content area is divided into two sections. The first section, titled 'Auth Logs List', contains buttons for Apply, Reset, and Refresh, followed by a table with columns 'Log' and 'Logging Status'. The second section, titled 'Selected Log: Successful Authentication Requests', contains a 'View' tab and a 'Search' input field. Below the search field is a list of log files, with '20150306' selected. A 'Page Size' dropdown is set to '20'. At the bottom of the 'View' tab are 'View' and 'Save' buttons. The right panel of the 'Selected Log' section is empty, with a header that says 'Select a Log File from the Left Panel to View'.

Log	Logging Status
Successful Authentication Requests	<input type="checkbox"/>
Invalid Shared Secret Requests	Enabled
Failed Authentication Requests	Enabled
Unknown Client Requests	Enabled

Select a Log File from the Left Panel to View

Select a Log to View

20150306

Page Size: 20

View Save

3. Select the check box to enable the selected authentication log file.
Clear the check box to disable the selected authentication log file.
4. Click **Apply** to save the configuration.

Viewing the Authentication Log Files

To display an authentication log file using the Web GUI:

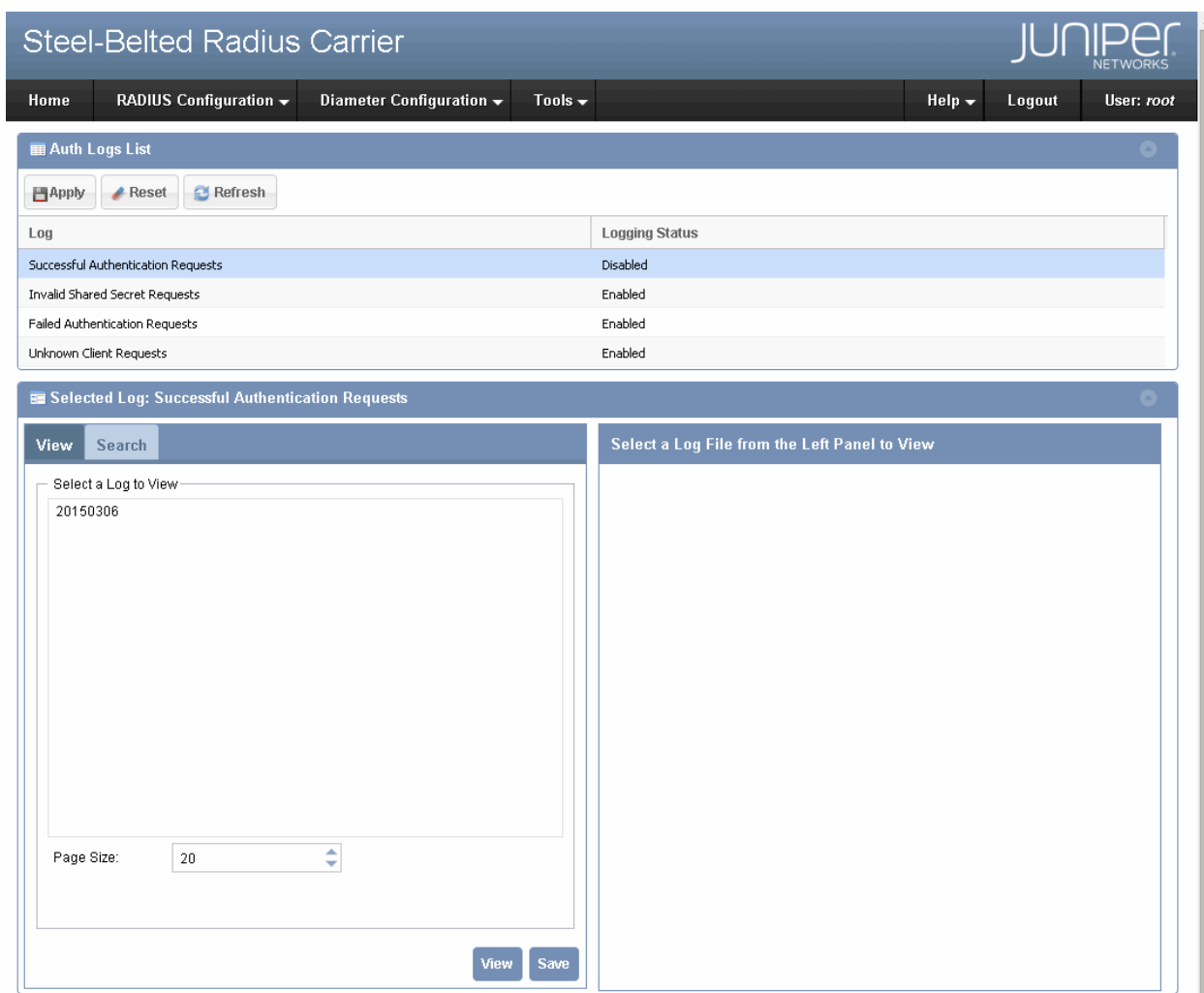
1. Select **RADIUS Configuration > Reports > Auth Logs**.

The **Auth Logs List** page (Figure 271 on page 841) appears.

2. Select the type of authentication log file you want to display.

The **Selected Log** pane (Figure 273 on page 843) appears with the **View** tab selected.

Figure 273: Selected Log Pane



3. Select the log you want to display and click **View**.

By default, Web GUI displays the authentication log file 20 lines at a time. To change the number of lines displayed, enter a different number in the **Page Size** field before you click **View**.

4. Click the **Prev Page** and **Next Page** buttons to page through the log file.

To sort the authentication log file, click the appropriate column header.

To refresh the authentication log file display, click the **Refresh** button.

Saving the Log Files

To save an authentication log file to a text file using the Web GUI:

1. Select **RADIUS Configuration > Reports > Auth Logs**.

The **Auth Logs List** page (Figure 271 on page 841) appears.

2. Select the type of authentication log file you want to display.

The **Selected Log** pane (Figure 273 on page 843) appears with the **View** tab selected.

3. Select the log you want to save and click **Save** to save the log file.

Searching the Log Files

You can search the SBR Carrier authentication log files to display messages within a specified time range, messages relating to a specific client, or messages relating to a specific user.

To search the authentication log files using the Web GUI:

1. Select **RADIUS Configuration > Reports > Auth Logs**.

The **Auth Logs List** page (Figure 271 on page 841) appears.

2. Select the type of authentication log file you want to display.

The **Selected Log** pane (Figure 273 on page 843) appears with the **View** tab selected.

3. Click the **Search** tab (Figure 274 on page 845).

Figure 274: Selected Log Pane—Search

Steel-Belted Radius Carrier

Home RADIUS Configuration Tools Help Logout User: root

Auth Logs List

Apply Reset Refresh

Log	Logging Status
Successful Authentication Requests	Disabled
Invalid Shared Secret Requests	Disabled
Failed Authentication Requests	Disabled
Unknown Client Requests	Disabled

Selected Log: Successful Authentication Requests

View Search

Date and Time

Time Zone: ☐ UTC ☒ Local

From

☒ Now

☐ Specific Date: HH:MM:SS

To

☒ No Limit

☐ Specific Date: HH:MM:SS

Filter By

☐ RADIUS Client:

☐ User Name:

Max Returns: 200

Search

Search Logs from the Left Panel

4. If you want to search the authentication log file for messages within a specified time range:

a. Select the time zone to be used for the search. The available options are:

- UTC—Uses the UTC time zone to search the log file for messages.
- Local—Uses the local time zone to search the log file for messages.

NOTE: The **Time Zone** setting corresponds to the **UTC** parameter in **authReportAccept.ini**, **authReportBadSharedSecret.ini**, **authReportReject.ini**, and **authReportUnknownClient.ini** files.

b. Select the **Now** option button in the **From** area to search the log file for messages until now.

Or

Select the **Specific Date** option button in the **From** area and select the specific date and time to specify the ending date and time for the search.

c. Select the **No Limit** option button in the **To** area to search the log file for messages in an infinite starting time.

Or

Select the **Specific Date** option button in the **To** area and select the specific date and time to specify the starting date and time for the search.

5. If you want to filter messages relating to a specified RADIUS client, select the **RADIUS Client** check box and enter the name of the RADIUS client in the **RADIUS Client** field.
6. If you want to filter messages relating to a specified user, select the **User Name** check box and enter the name of the user in the **User Name** field.
7. If you want to limit the number of messages to be displayed, enter a number in the **Max Returns** field.
8. Click **Search**.

The search results are displayed in the right-hand side of the **Selected Log** pane ([Figure 273 on page 843](#)).

Using the Locked Accounts List

Account lockout lets you disable an account after a configurable number of failed login attempts within a configurable period. For example, if a user enters an incorrect password three times within two minutes, SBR Carrier can lock out the user's account temporarily. During the lockout period, the user cannot log in, even with the correct password.

When a user account is locked out, the user must wait until the lockout period expires, or a network administrator manually clears the lockout status for the account.

NOTE: Do not enable account lockout and account redirection at the same time. If account lockout and account redirection are both enabled, account lockout is used and account redirection settings are ignored. For information about account redirection, see ["Account Redirection" on page 55](#).

NOTE: Account lockout works locally only, not globally. Account lockout state is not maintained if SBR Carrier is restarted.

Configuring Locked Account Settings

To configure account lockout, edit the **lockout.ini** file. For information about the **lockout.ini** file, see the *SBR Carrier Reference Guide*.

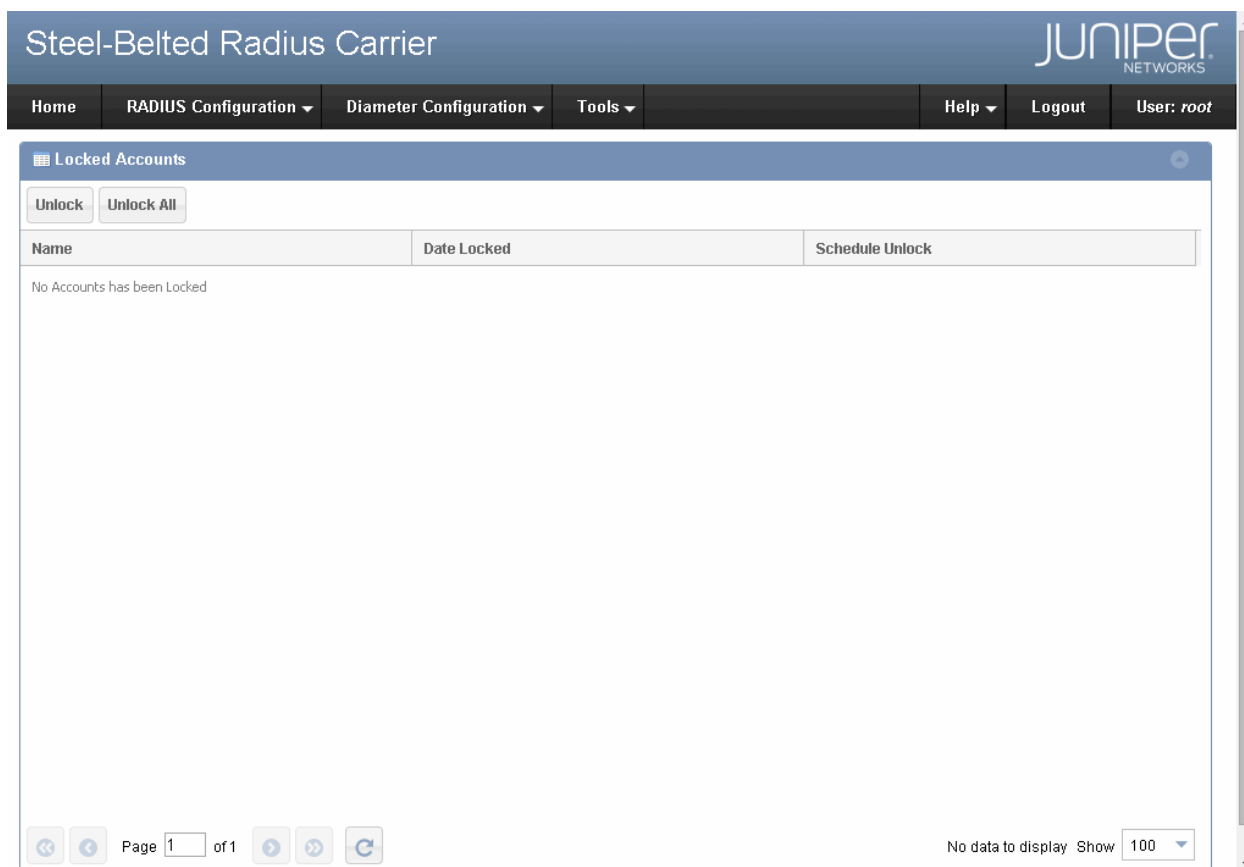
Unlocking a Locked Account

To unlock a locked account using the Web GUI:

1. Select **RADIUS Configuration > Reports > Locked Accounts**.

The **Locked Accounts** page (Figure 275 on page 847) appears. This page displays a list of user accounts that have been locked.

Figure 275: Locked Accounts Page



2. Select the account you want to unlock from the list.
3. Click **Unlock** to unlock the selected account.

To unlock all currently locked accounts, click **Unlock All**.

NOTE: You can use the LDAP configuration interface to clear a locked-out account by creating and executing an LDIF file with the following commands:

```
dn: user=user_name, radiusstatus=lockout, o=radius
changetype: delete
```

Where *user_name* is the name of the locked-out user.

Unlocking a locked-out user through LCI is supported only for standalone servers.

For information about using the LDAP configuration interface, see

[“Using the LDAP Configuration Interface” on page 497.](#)

Configuring the Log Retention Period

Each day at midnight, the previous day's log files are completed, and new log files are created for the new day's transactions. To prevent the log files from filling up available disk space, you can configure SBR Carrier to discard the log files after a specified number of days.

To configure the log retention period using the Web GUI:

1. Select **RADIUS Configuration > Reports > Settings**.

The **Settings** page ([Figure 276 on page 848](#)) appears.

Figure 276: Settings Page

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

Settings

Days to Keep: 365

Server Log File:

Save Reset Refresh

2. Enter the number of days you want SBR Carrier to retain the log file in the **Days to Keep Server Log File** field.
3. Click **Save** to save the setting.

Cluster Management Nodes

A cluster management node is a machine that hosts the SSR management process. It controls itself and all data nodes in the cluster. It provides configuration data, starts and stops nodes, backs up the database, and performs other database operations. It also manages a database process that supports the SSR storage engine. Cluster configuration data is located in an identical **config.ini** file on each node of the management cluster.

Cluster management logs are available in the following directory:

/opt/JNPRndb_mgmd/logs

To configure log retention for cluster management nodes:

1. Log in to the SBR system.
2. Open the **config.ini** file from the **/opt/JNPRhadm** path.
3. Update the **LogDestination** option under the **[ndb_mgmd]** section to specify the maximum size of the log file in bytes and the number of logs to be preserved.

Default value for LogDestination=ndb_node_id_cluster.log

Default value for maxsize=1000000

Default value for maxfiles=6

4. When ndb_node_id_cluster.log reaches the configured maxsize, the existing log file name is incremented by the next highest available number and a new ndb_node_id_cluster.log is created.

For example, when the ndb_node_id_cluster.log file reaches its maxsize, a new ndb_node_id_cluster.log file is created to record the logs and the previously recorded log file is incremented to ndb_node_id_cluster.log.1.

5. Alternatively, specify LogDestination=SYSLOG:facility=local0 and configure the log file destination within **syslog.conf**. Log rotation at fixed intervals can be controlled by **logrotate.d**.

SSR Data Nodes

An SSR data node is machine that hosts the SSR data process. It runs the **ndbd** data process. The **ndbd** process cooperatively manages, replicates, and stores data in the SSR storage engine with other data nodes. Each data node has its own memory and permanent storage. Each one maintains both a portion of the working copy of the SSR database and a portion of one or more replicas of the database.

SSR data node logs are available in the following directory:

/opt/JNPRndbd/logs

To configure log retention for SSR data nodes:

1. Log in to the SBR system.
2. Open the **config.ini** file from the **/opt/JNPRhadm** path.
3. Update the **MaxNoOfSavedMessages** option under the **[ndbd]** section to specify the number of trace files to be stored on the SSR data nodes.

Default value for MaxNoOfSavedMessages=25

4. When the configured limit is reached, the existing logs are overwritten.

Using the Server Log File

The server log file records RADIUS events, such as server startup or shutdown or user authentication or rejection, as a series of messages in an ASCII text file. Each line of the server log file identifies the date and time of the RADIUS event, followed by event details. You can open the current log file while SBR Carrier is running.

Server log files are kept for the number of days specified in the **Settings** page (described in [“Configuring the Log Retention Period” on page 848](#)) and then deleted to conserve disk space.

You can roll the server log file over based on time (using the Rollover parameter), file size (using either the **LogFileMaxMBytes** or **MaxSize** parameter, or a combination of size and time.

When rolling over the file based on size, you can specify a maximum size for a server log file by entering a non-zero value for either the **LogFileMaxMBytes (megabytes)** or **MaxSize (bytes)** setting in the [Logging] section of the radius.ini file. If both **LogFileMaxMBytes** and **MaxSize** are present in this section, **MaxSize** is ignored and the log file size is based on **LogFileMaxMBytes (MBytes)**. If you want to configure the maximum file size in bytes, do not include the **LogFileMaxMBytes** parameter in this section.

- If a maximum file size is set, the server log filename identifies the date and time it was opened. Log files are named as follows: **yyyymmdd_xxxxx.log**, where xxxxx is the sequence beginning with 00000. When the current server log file approaches the specified number, either megabytes (1024 x 1024 bytes) if using **LogFileMaxMBytes**, or bytes if using **MaxSize**, the current log file is closed and a new one is opened. The closed file is slightly smaller than the specified maximum file size.
- If the maximum file size is set to 0 (or if the **LogFileMaxMBytes** setting is absent), the server log file size is ignored and log file names are date stamped to identify when they were opened (**YYYYMMDD.log**).

NOTE: The size of the log file is checked once per minute. The log file may exceed the specified maximum file size temporarily (for less than a minute) after it passes the LogfileMaxMBytes threshold between size checks.

By default, server log files are located in the RADIUS database directory. You can specify an alternate destination directory in the [Configuration] section of the **radius.ini** file.

NOTE: If you specify an alternate destination directory other than the default, ensure that the directory exists before starting the SBR. Otherwise, SBR may fail to function correctly.

Level of Logging Detail

You can control the level of detail recorded in server log files by use of the **LogLevel**, **LogGroup**, **LogAccept**, and **LogReject** settings.

The **LogGroup** parameter specifies the type of server functionality for which you want to log details in the server log file. **LogGroup** can be the numbers from 0 through 4, where 0—All, 1—Administration, 2—SessionControlSuccess and SessionControlFailure, 3—Diameter Peer State, and 4—Others. You can specify more than one number in this parameter; the numbers must be comma separated. For more detailed information about the **LogGroup** parameter, see *SBR Carrier Reference Guide*.

The **LogLevel** setting determines the level of detail to be captured in the server log file for the configured log groups. You configure a log level setting for all enabled log groups. The **LogLevel** can be the number 0, 1, or 2, where 0 is the least amount of information, 1 is intermediate, and 2 is the most verbose. The **LogLevel** setting is specified in the [Logging] section of **radius.ini** and in the [Settings] sections of **.aut** files.

For example, if you set the **LogLevel** parameter to 2 and the **LogGroup** parameter to 2:

radius.ini

[Logging]

...

LogLevel = 2

LogGroup = 2

...

Then, the entries to the server log file contain only debugging messages for session success and failure scenarios.

NOTE: Configuration logs are always included in the server log file.

The **LogAccept** and **LogReject** flags allow you to turn on or off the logging of Access-Accept and Access-Reject messages in the server log file. These flags are set in the [Logging] section of **radius.ini**: a value of 1 causes these messages to be logged, and a value of 0 causes the messages to be omitted. An Accept or Reject is logged only if **LogAccept** or **LogReject**, respectively, is enabled and the **LogLevel** is verbose enough for the message to be recorded.

The **TraceLevel** setting specifies whether packets should be logged when they are received and being processed, and what level of detail should be recorded in the log. The **TraceLevel** can be the number 0, 1, or 2.

If you alter the **LogLevel** or **TraceLevel** settings, you can have them take effect without restarting the server by issuing the **./sbrd hup** command.

Using the Authentication Log File

The authentication log file records each RADIUS authentication request received by SBR Carrier. Authentication log files are Comma Separated Value (CSV) ASCII text files that can be imported into a spreadsheet or database program.

Authentication log files are located in the RADIUS database directory area by default, although you can specify an alternate destination directory in the [Configuration] section of **authlog.ini**. Authentication log files are named **yyyymmdd.authlog**, where **yyyy** is the 4-digit year, **mm** is the month, and **dd** is the day on which the log file was created.

Authentication log files are kept for the number of days specified in the **Settings** page (described in [“Configuring the Log Retention Period” on page 848](#)), and are deleted after that.

The current log file can be opened while SBR Carrier is running.

Authentication Log File Format

The first five fields in every authentication log entry are required by SBR Carrier:

- **Date**—The date when the event occurred
- **Time**—The time when the event occurred

- **RAS-Client**—The name or IP address of the RADIUS client sending the authentication request
- **Full-Name**—The fully distinguished name of the user, based on the authentication performed by the RADIUS server
- **ACC/REJ**—The result of the authentication request (ACCEPT or REJECT)

The RADIUS attributes specified in the **authlog.ini** file appear next. Attributes in the **authlog.ini** file beginning with a semicolon (;), are commented out, and their values are not recorded in the authentication log file.

```
User-Name
NAS-IP-Address
NAS-Port
Service-Type
Framed-Protocol
Framed-IP-Address
Framed-IP-Netmask
Framed-Compression
Login-IP-Host
Callback-Number
State
Called-Station-Id=
Calling-Station-Id=
NAS-Identifier=
Proxy-State=
Login-LAT-Service
Login-LAT-Node
Login-LAT-Group
Event-Timestamp
NAS-Port-Type
Port-Limit
Login-LAT-Port
```

NOTE: If the User-Password attribute is included in the **authlog.ini** file, it is ignored during processing to prevent exposing users' clear-text passwords in the log file.

You can include vendor-specific attributes if the device sending the authentication packet supports them. For more information, see [“Vendor-Specific Attributes” on page 42](#).

You can edit the **authlog.ini** file to add, remove, or reorder the standard RADIUS or vendor-specific attributes that are logged. For information about **authlog.ini**, refer to the *SBR Carrier Reference Guide*.

First Line Headings

The first line of the authentication log file lists the names of all the attributes that have been enabled for logging, in the order in which they are logged. This first line serves as a complete set of column headings for the remaining entries in the file. The content of the first line depends on the attributes specified in the **authlog.ini** file.

The following example shows the heading line and an authentication log file entry consisting of the required attributes.

```
"Date", "Time", "RASClient", "FullName", "ACC/REJ"
"7/3/2003", "12:11:55", "RRAS", "EdisonCarter", "ACCEPT",
```

Comma Placeholders

Log entries may not include every attribute listed in the first line of the authentication log file. When SBR Carrier records the event in the authentication log file, it uses a comma *placeholder* to mark empty entries, so that all entries remain aligned with their headings.

For example, the following log entries indicate that Bob's authentication request was rejected but Alice's authentication request was accepted. The reported fields include Called-Station-Id, Calling-Station-Id, and Port-Limit. The attributes listed in the log heading that were not returned for the authentication events are separated with commas.

```
"Date", "Time", "RAS-Client", "Full-Name", "Acc/Rej",
"User-Name", "NAS-IP-Address", "NAS-Port", "Service-Type",
"Framed-Protocol", "Framed-IP-Address", "Framed-IP-Netmask", "Framed-Compression",
>Login-IP-Host", "Callback-Number", "State", "Called-Station-Id", "Calling-Station-Id",
>NAS-Identifier", "Proxy-State", "Event-Timestamp", "NAS-Port-Type", "Port-Limit",
>Login-LAT-Port""07/14/2003", "13:39:10", "192.168.2.42",
>"BOB", "REJECT" ,,,,,,,,,, "Alice's Office", "Bob's Office" ,,,,, "5",
"07/14/2003", "13:43:26", "192.168.2.42", "ALICE", "ACCEPT" ,,,,,,,,,, "Bob's Office",
"Alice's Office" ,,,,, "5",
```

Using the Accounting Log File

RADIUS accounting events are recorded in the accounting log file. Accounting events include START messages, which indicate the beginning of a connection; STOP messages, which indicate the termination of a connection; and INTERIM messages, which indicate a connection is ongoing.

Accounting log files use comma-delimited, ASCII format, and are intended for import into a spreadsheet or database program. Accounting log files are located in the RADIUS database directory area by default, although you can specify an alternate destination directory in the [Configuration] section of the **account.ini** file. Accounting log files are named **yyyymmdd.ACT**, where **yyyy** is the four-digit year, **mm** is the month, and **dd** is the day on which the log file was created.

Accounting log files are kept for the number of days specified in the **Settings** page (described in [“Configuring the Log Retention Period” on page 848](#)), and are deleted after that to conserve disk space.

NOTE: An accounting log file is not created when there is no accounting request or activity from the client.

The current log file can be opened while SBR Carrier is running.

By default, SBR Carrier truncates a line in the accounting log when a non-printing character is encountered. You can set the **ReplaceUnprintables** parameter in the [Logging] section of **radius.ini** with a printable character which is used instead of non-printing characters when SBR Carrier writes messages to the accounting log file.

NOTE: Characters of ASCII decimal code 0 through 31 (ASCII hex code 0 through 1F) and 127 through 255 (ASCII hex code 7F through FF) are considered as non-printing characters. For more information about the **ReplaceUnprintables** parameter, see the *SBR Carrier Reference Guide*.

Accounting Log File Format

The first six fields in every accounting log entry are provided by SBR Carrier for your convenience in reading and sorting the file:

- **Date**—The date when the event occurred
- **Time**—The time when the event occurred
- **RAS-Client**—The name or IP address of the RADIUS client sending the accounting record
- **Record-Type**—START, STOP, INTERIM, ON, or OFF, the standard RADIUS accounting packet types
- **Full-Name**—The fully distinguished name of the user, based on the authentication performed by the RADIUS server
- **Auth-Type**—A number that indicates the class of authentication performed:

0—Native

13—Solaris User
 14—Solaris Group
 100—Tunnel User
 200—External Database
 (other)—Proxy

By default, the standard RADIUS attributes follow the **Auth-Type** identifier. See [“Standard RADIUS Accounting Attributes” on page 857](#).

You can include vendor-specific attributes if the device sending the accounting packet supports them. For more information, see [“Vendor-Specific Attributes” on page 42](#).

You can edit the **account.ini** initialization file to add, remove or reorder the standard RADIUS or vendor-specific attributes that are logged. For information about **account.ini**, refer to the *SBR Carrier Reference Guide*.

First Line Headings

The first line of the accounting log file is a file header that lists the attributes that have been enabled for logging in the order in which they are logged. The following example of a first line shows standard RADIUS headings in bold, and vendor-specific headings in regular text:

“Date”, “Time”, “RAS Client”, “Record Type”, “Full Name”, “Auth Type”,
“User Name”, “NAS Port”, “Acct Status Type”, “Acct Delay Time”,
“Acct Input Octets”, “Acct Output Octets”, “Acct Session Id”,
“Acct Authentic”, “Acct Session Time”, “Acct Input Packets”,
“Acct Output Packets”, “Acct Termination Cause”, “Acct Multi Session Id”,
“Acct Link Count”, “Acc Err Message”,
 “NauticaAcctSessionId”, “NauticaAcctDirection”,
 “NauticaAcctCauseProtocol”, “NauticaAcctCauseSource”,
 “TelebitAccountingInfo”, “LastNumberDialedOut”,
 “LastNumberDialedInDNIS”, “LastCallersNumberANI”,
 “Channel”, “EventId”, “EventDateTime”, “CallStartDateTime”, “CallEndDateTime”,
 “DefaultDTEDataRate”, “InitialRxLinkDataRate”,
 “FinalRxLinkDataRate”, “InitialTxLinkDataRate”,
 “FinalTxLinkDataRate”, “SyncAsyncMode”,
 “OriginateAnswerMode”, “ModulationType”,
 “EqualizationType”, “FallbackEnabled”, “CharactersSent”,
 “CharactersReceived”, “BlocksSent”, “BlocksReceived”,
 “BlocksResent”, “RetrainsRequested”, “RetrainsGranted”,
 “LineReversals”, “NumberOfCharactersLost”,
 “NumberofBfers”, “NumberofLinkTimeouts”,
 “NumberofFallbacks”, “NumberofUpshifts”,

```
"NumberofLinkNAKs", "BackChannelDataRate",
"SimplifiedMNPLLevels", "SimplifiedV42bisUsage", "PW_VPN_ID"
```

Comma Placeholders

SBR Carrier writes accounting events to the accounting log file, If an event recorded in the accounting log file does not have data for every attribute, a comma *placeholder* marks the empty entry, so that all entries remain correctly aligned with their headings. For example, based on the first line of headings described above, the following is a valid accounting log entry, in which the value of the Acct-Status-Type attribute is 7:

```
"12/23/1997", "12:11:55", "RRAS", "AccountingOn",
,7,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
```

Standard RADIUS Accounting Attributes

[Table 123 on page 857](#) lists the standard RADIUS accounting attributes defined in RFC 2866, *RADIUS Accounting*.

Table 123: Standard RADIUS Accounting Attributes

Attribute	Description
User-Name	The name of the user as received by the client.
NAS-Port	The port number on the client device.
Acct-Status-Type	A number that indicates the beginning or ending of the user service: <ul style="list-style-type: none"> 1—Start 2—Stop 3—InterimAcct 7—Accounting-On 8—Accounting-Off
Acct-Delay-Time	Indicates how many seconds the client has been trying to send this record, which can be subtracted from the time of arrival on the server to find the approximate time of the event generating this request.

Table 123: Standard RADIUS Accounting Attributes *(continued)*

Attribute	Description
Acct-Input-Octets	Number of octets (bytes) received by the port over the connection; present only in STOP records.
Acct-Output-Octets	Number of octets (bytes) sent by the port over the connection; present only in STOP records.
Acct-Session-Id	Identifier used to match START and STOP records in a log file.
Acct-Authentic	Indicates how the user was authenticated by RADIUS, the network access device (local), or another remote authentication protocol: 1—RADIUS 2—Local 3—Remote
Acct-Session-Time	Elapsed time of connection in seconds; present only in STOP records.
Acct-Input-Packets	Number of packets received by the port over the connection; present only in STOP records.
Acct-Output-Packets	Number of packets sent by the port over the connection; present only in STOP records.

Table 123: Standard RADIUS Accounting Attributes *(continued)*

Attribute	Description
Acct-Termination-Cause	<p>Number that indicates how the session was terminated; present only in STOP records:</p> <ul style="list-style-type: none"> 1—User Request 2—Lost Carrier 3—Lost Service 4—Idle Timeout 5—Session Timeout 6—Admin Reset 7—Admin Reboot 8—Port Error 9—NAD Error 10—NAD Request 11—NAD Reboot 12—Port Unneeded 13—Port Preempted 14—Port Suspended 15—Service Unavailable 16—Callback 17—User Error 18—Host Request
Acct-Multi-Session-Id	<p>Unique accounting identifier to make it easy to link together multiple related sessions in a log file.</p>
Acct-Link-Count	<p>The count of links that are known to have been in a given multi-link session at the time the accounting record is generated.</p>

12

PART

Optional Scripting Module

[Introduction to Scripting](#) | **861**

[Creating Scripts](#) | **866**

[Debugging Scripts](#) | **876**

[Creating LDAP Scripts](#) | **881**

[Creating Realm Selection Scripts](#) | **895**

[Creating Attribute Filter Scripts](#) | **908**

[Working with Data Accessors](#) | **919**

[Script Reference](#) | **947**

Introduction to Scripting

IN THIS CHAPTER

- [Scripting Overview | 861](#)
- [Script Types | 862](#)
- [About JavaScript | 865](#)

This chapter introduces the key concepts of Steel-Belted Radius Carrier scripting. It provides examples of how you can use scripting to extend the capabilities of the Steel-Belted Radius Carrier server. This chapter contains these topics:

Incorporating scripts into your Steel-Belted Radius Carrier configuration enables you to fine-tune the behavior of the Steel-Belted Radius Carrier server and implement custom request processing logic. You can use scripts to configure Steel-Belted Radius Carrier to evaluate complex decision logic and manipulate RADIUS request data objects in ways that cannot be expressed through settings in the standard Steel-Belted Radius Carrier initialization files.

Steel-Belted Radius Carrier scripts are written in JavaScript, an easy-to-use, industry standard scripting language with a powerful, object-based syntax.

Scripting Overview

The Steel-Belted Radius Carrier server invokes many built-in functional modules while processing RADIUS requests. These modules are configured by initialization files in the Steel-Belted Radius Carrier home directory. For example, you configure the realm selection module with settings in the **proxy.ini** file.

With scripting, you can supplement or override specific functional modules within the Steel-Belted Radius Carrier server by implementing custom processing logic written in JavaScript. JavaScript APIs allow scripts to perform tasks like these:

- Manipulate RADIUS request attributes.
- Select the processing realm for a request.

- Query external SQL and LDAP servers.
- Print information and debug messages to the server log.

NOTE: Changing passwords through scripting or filters is not supported.

Scripts are part of special initialization files that contain both the script text and settings required by Steel-Belted Radius Carrier to execute (and optionally debug) the script. JavaScript initialization (.jsi) files use a parameter syntax similar to that of other Steel-Belted Radius Carrier configuration files.

To configure a script to load and run, you refer to it by name using the **Script** keyword at the appropriate place in a Steel-Belted Radius Carrier initialization file. The context in which a script executes, which determines the data objects and JavaScript APIs available to it, depends on where the script reference appears in the Steel-Belted Radius Carrier configuration.

NOTE: For the LDAP authentication plug-in, script settings are embedded directly in the `ldapauth.aut` file.

Scripts are loaded, then compiled, when the server boots or when the script is first invoked. If a script fails to load or compile, a diagnostic error message is added to the log and the associated function is either disabled or reverts to its default behavior.

Your script executes each time the flow of control within Steel-Belted Radius Carrier enters a functional module that is configured to run that script. The scripting infrastructure automatically sets up the correct environment for the script, depending on its type. The script executes until it returns normally or it encounters a runtime exception. To prevent the script from being caught in an infinite loop, you can configure an optional watchdog counter to terminate the script after it has executed a preset number of operations.

Logging and tracing functions are provided as an aid to script debugging. Scripts can send messages directly to the Steel-Belted Radius Carrier server log. Additionally, you can use the script trace feature to write line-by-line debug information to the log. Each script trace frame contains the text, filename, and line number of the next JavaScript statement for the script to execute, and the names and values of user-specified script variables.

Script Types

Steel-Belted Radius Carrier supports several types of scripts, each linked to a functional module.

- [“LDAP Authentication” on page 863](#)—Scripts that control the execution of searches and the processing of attributes by the LDAP authentication plug-in. The LDAP authentication scripts are executed only by the LDAP authentication plug-in.
- [“Realm Selection” on page 863](#)—Scripts used to determine the name of a proxy or directed realm to which a RADIUS request is directed for processing. Realm selection scripts are executed during normal request processing by the Steel-Belted Radius Carrier server core and during inner authentication by the tunneled authentication plug-ins (PEAP and TTLS).
- [“Attribute Filter” on page 864](#)—Scripts used to manipulate the values of attributes in the RADIUS request or response packets. Attribute filter scripts are executed any time a server core component or plug-in module invokes an attribute filter that is configured for scripting.

LDAP Authentication

Steel-Belted Radius Carrier uses the LDAP authentication plug-in to authenticate users and retrieve attributes from external LDAP repositories. LDAP connection parameters and search specifications are defined in the **ldapauth.aut** file.

You can configure the LDAP authentication plug-in to perform scripted or unscripted searches. With unscripted searches, selected attributes can be transferred directly from the RADIUS request into the LDAP search string, and from the LDAP search result into the RADIUS response. You can create a simple search tree to execute a sequence of LDAP searches each time Steel-Belted Radius Carrier processes an authentication request.

With LDAP authentication scripts, you have even greater control over the execution of LDAP searches and the processing of attribute values and search results. You can combine, manipulate, and test attribute values, and define conditional logic to select which searches to execute.

Uses for LDAP authentication scripts include:

- Modifying the username and retrying the LDAP search in the case that the initial search returns no result from the repository.
- Selecting a RADIUS response profile for the user based on attributes returned from the LDAP server.
- Reformatting the LDAP result data before assigning values to the RADIUS response.
- Using the results from prior LDAP searches to select subsequent LDAP searches to execute.

For more details, see [“Creating LDAP Scripts” on page 881](#).

Realm Selection

A realm is a collection of authentication methods that Steel-Belted Radius Carrier invokes to process a RADIUS request. When an authentication request is received, Steel-Belted Radius Carrier uses the username, selected RADIUS attributes, or other properties of the request to determine which realm handles the

request. The selected realm can be a proxy realm, a directed realm, or the default realm (if no explicit realm is selected).

Realm selection is performed both by the Steel-Belted Radius Carrier server core and during inner authentication by tunneled authentication plug-ins. Five built-in realm selection methods, plus the scripted method, are supported. Using realm selection scripts, you can define programmed logic to select the realm for processing each RADIUS request. Realm selection scripts may retrieve RADIUS request attributes, query external SQL or LDAP servers, or invoke any of the built-in realm selection methods.

Uses for realm selection scripts include:

- Querying multiple LDAP servers to look up the realm name for a specific user.
- Combining multiple RADIUS request attributes to form a SQL database key for retrieving the realm name.
- Changing the authentication username.
- Setting a profile to be applied to the RADIUS response once the user is authenticated.

For more details, see [“Creating Realm Selection Scripts” on page 895](#).

Attribute Filter

Steel-Belted Radius Carrier uses attribute filters to allow, exclude, add, or modify attribute values in the RADIUS response and request packets. Attribute filters are also used to transfer attribute values in and out of the inner methods of tunneled authentication plug-ins. Attribute filters are defined by name using the Web GUI and are referred to throughout the server configuration.

Unscripted or static attribute filters use simple, fixed rules for manipulating RADIUS attributes. In contrast, scripted attribute filters enable you to specify detailed algorithms to read, write, modify, and delete request and response attribute values. You can query external SQL or LDAP servers and execute static attribute filters by name from your attribute filter scripts.

Uses for attribute filter scripts include:

- Using an LDAP query to select a static attribute filter to execute.
- Adding or removing selected values from a multi-valued attribute.
- Editing the values of string attributes.
- Accepting or rejecting requests based on mathematical calculations on numeric attribute values.

For more details, see [“Creating Attribute Filter Scripts” on page 908](#).

About JavaScript

Steel-Belted Radius Carrier uses the open-source SpiderMonkey JavaScript engine from the Mozilla Foundation to compile and execute scripts. SpiderMonkey is an implementation of JavaScript 1.7, which adheres to the international ECMAScript (ECMA-262) standard.

For more information about SpiderMonkey and links to JavaScript references, see <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

NOTE: The JavaScript compiler and interpreter are Steel-Belted Radius Carrier components. They operate independently from the browser and JVM installed on the machine.

Creating Scripts

IN THIS CHAPTER

- [Script Development Steps | 866](#)
- [JavaScript Initialization Files | 867](#)
- [Writing Steel-Belted Radius Carrier Scripts in JavaScript | 870](#)
- [Saving the Script File | 873](#)
- [Sample Script | 873](#)

This chapter describes the common elements of writing scripts for Steel-Belted Radius Carrier, including the JavaScript initialization file configuration settings. A simple example shows the concepts explained in this chapter.

Script Development Steps

To create and deploy a script on the Steel-Belted Radius Carrier server, you typically follow these steps:

1. Create a JavaScript initialization file containing the script statements and runtime settings and save it in the appropriate directory under the Steel-Belted Radius Carrier installation root directory.
2. Define SQL or LDAP data accessor **.gen** files as required by your script.
3. Run the **scriptcheck** utility to verify your script syntax.
4. Declare your script using the **Script** keyword in the appropriate Steel-Belted Radius Carrier initialization file(s), unless it is an LDAP authentication script.
5. Start the Steel-Belted Radius Carrier server and check the server log to verify successful loading and compilation of the script.
6. Send a RADIUS request to the server and check the server log for debug messages and trace output from your script.

JavaScript Initialization Files

Steel-Belted Radius Carrier scripts are contained in JavaScript initialization (**.jsi**) files, which are similar in format to other Steel-Belted Radius Carrier configuration files. Each **.jsi** file consists of a number of section headers and associated configuration settings. The JavaScript text itself appears in a separate [Script] section within the file. With minor exceptions, the **.jsi** file headers and settings are the same for all script types.

NOTE: Script settings for the LDAP authentication plug-in are embedded directly in the **ldapauth.aut** file. Except where noted, this guide applies equally to both **ldapauth.aut** and **.jsi** file types. For information about configuring other LDAP authentication plug-in settings, see *LDAP Authentication File* in the *SBR Carrier Reference Guide*.

Each **.jsi** file can contain these sections:

- [Settings]
- [Script]
- [ScriptTrace] (*optional*)
- [Failure] (*optional*)

[Settings] Section

The [Settings] section ([Table 124 on page 868](#)) contains parameters that control logging and debugging of your script.

- The **LogLevel** parameter sets the default level assigned to log messages produced by calls to the **SbrWriteToLog()** and **SbrTrace()** API functions. To determine if the message appears in the log, Steel-Belted Radius Carrier compares the message log level to the server log level (configured by the **LogLevel** parameter in **radius.ini**). If the server log level is greater than or equal to the message log level, the message is written to the Steel-Belted Radius Carrier log. If server log level is less than the message log level, the message is not written to the Steel-Belted Radius Carrier log.

NOTE: You can override the script file **LogLevel** parameter when calling **SbrWriteToLog()** and **SbrTrace()** using the optional **LogLevel** function argument. For more details, see [“SbrWriteToLog\(\)” on page 876](#) and [“SbrTrace and ScriptTraceLevel” on page 877](#).

- The **ScriptTraceLevel** parameter controls the amount of line-by-line debugging information that is produced automatically or under program control by the script. At the lowest level, no tracing is performed,

and at the highest level, a trace is written to the log for every JavaScript statement that is executed. See [“Debugging Scripts” on page 876](#) for more information about script debugging.

- The **MaxScriptSteps** parameter limits the number of branch callbacks that a script can perform in a single invocation. A branch callback is a backwards branch in the script code (for example, what occurs in a **for** loop), or a return from a function call. If the limit is reached, the script is automatically terminated with a runtime exception.

Table 124: [Settings] Section Parameters

Parameter	Function
LogLevel	<p>Specifies the default log level at which messages are produced by calls to SbrWriteToLog() and SbrTrace(). The value must be less than or equal to the LogLevel value in the radius.ini file for messages to appear. The parameter can be overridden by supplying a LogLevel argument in the function calls.</p> <p>Default value is 0.</p>
ScriptTraceLevel	<p>Controls the generation of line-by-line script trace information in the log.</p> <ul style="list-style-type: none"> • At Level 0, no traces are logged. • At Level 1, traces are logged only when the SbrTrace() function is executed by the script. • At Level 2, a trace is generated for every line executed by the script. <p>Default value is 0.</p>
MaxScriptSteps	<p>Limits the number of branch callbacks that can be executed during a single script invocation. If the limit is reached, the script automatically terminates with a runtime exception.</p> <p>Default value is 10000.</p>

[Script] Section

The [Script] section contains the body of your script. Unlike other configuration file sections, where parameters appear on individual lines, the script is entered as multi-line blocks of text. The script is processed until a line is encountered that begins with a left bracket (“[”) or the end of the file is reached.

This example shows a simple [Script] section containing code that writes a message to the server log.

```
[Script]
```

```
// Define a function that writes its arguments to the log.
function writeLog(message) {
    SbrWriteToLog("The message is:" + message);
}

// Call the log function.
var msg = "Hello, world" ;
writeLog(msg);

// Return successfully.
return SCRIPT_RET_SUCCESS;
```

[ScriptTrace] Section

The [ScriptTrace] section ([Table 125 on page 869](#)) is optional. You can use the [ScriptTrace] section to select specific data values to print in the script trace logs. If you enable script tracing but do not specify any parameters in the [ScriptTrace] section, the trace frames contain statement and line number information but no script data values.

Each line in the [ScriptTrace] section specifies a type string and an argument. The type string selects the type of data value to be traced and the argument specifies its name.

These types are supported:

- **var**—The argument is the name of a local or global script variable.
- **attr**—The argument is the name of an LDAP variable table entry (**ldapauth.aut** only).

Table 125: [ScriptTrace] Section Parameters

Parameter	Function
var	Declares the name of a local or global JavaScript variable that appears in script trace logs.
attr	Declares the name of an LDAP variable table entry that appears in script trace logs (ldapauth.aut only).

```
[ScriptTrace]
var = count
var = userid
attr = User-Name (ldapauth.aut only)
attr = Service-Type (ldapauth.aut only)
```

In this example, the identifiers **count** and **userid** refer to the JavaScript variables in the script execution context. The identifiers **User-Name** and **Service-Type** refer to entries in the LDAP variable tables and takes effect only when declared in an **ldapauth.aut** script file.

[Failure] Section

The [Failure] section is optional. It specifies a string value that is ultimately returned by a script if the script first returns **SCRIPT_RET_FAILURE**.

- For LDAP authentication scripts, the value of the [Failure] section has a more complex interpretation. For details about the [Failure] section of the **ldapauth.aut** file, see *LDAP Authentication File* in the *SBR Carrier Reference Guide*.
- For realm selection scripts, the value of the [Failure] section specifies the name of the realm to be returned if the script execution fails.
- For attribute filter scripts, the value of the [Failure] section specifies the name of a static attribute filter to execute if the script execution fails.

Writing Steel-Belted Radius Carrier Scripts in JavaScript

This section provides general information and guidelines for writing Steel-Belted Radius Carrier scripts in JavaScript. For descriptions of the Steel-Belted Radius Carrier API functions available to LDAP, realm selection, and attribute filter scripts, see “[Script Reference](#)” on [page 947](#).

Programming in JavaScript

The Steel-Belted Radius Carrier script engine supports JavaScript 1.7 (ECMA-262). You can use any legal JavaScript syntax in your scripts and invoke the standard global object function and attributes, such as **Math**, **String**, and **Date**.

The JavaScript engine runs entirely within the Steel-Belted Radius Carrier server and is not associated with any Web browser. Browser-specific data objects and JavaScript language extensions are not supported by Steel-Belted Radius Carrier.

For in-depth information about JavaScript programming, see the official ECMAScript standard documentation at this URL:

<http://www.ecma-international.org/publications/standards/Ecma-262.htm>

Hidden Wrapper Function

Before Steel-Belted Radius Carrier compiles your script, it wraps the **[Script]** section statements in a JavaScript function named `_sbrScriptMain_`. When the server executes the script, the script invokes this function first. There are no arguments to the function call.

The hidden wrapper function enables your scripts to return result values to the server. This is required because the JavaScript language does not support **return** statements from the global execution context. For convenience, the hidden function call does not appear in script traces, and line numbers in script traces and error messages are adjusted to refer to the exact lines of your JavaScript initialization files.

With few exceptions involving advanced JavaScript programming, the existence of the hidden wrapper function is transparent to your scripts.

Script Return Values

Steel-Belted Radius Carrier uses the script return value to determine what action to take on the pending RADIUS request once the script finishes executing. The script type determines how the return value is interpreted by the server. A legal return value must be:

- A pre-defined return code such as **SCRIPT_RET_SUCCESS**
- A text string
- The JavaScript **null** object

The **SCRIPT_RET_SUCCESS** and the **SCRIPT_RET_FAILURE** codes are defined in the global object for all script types. Returning **SCRIPT_RET_SUCCESS** indicates to Steel-Belted Radius Carrier that script execution completed normally.

Returning **SCRIPT_RET_FAILURE** indicates that an unexpected error occurred during script execution (for example, a database search returned invalid results). If a script returns **SCRIPT_RET_FAILURE** and a [Failure] string is defined for that script, that string is returned as the script result. Otherwise, an error message appears in the server log and the pending RADIUS request is rejected.

Result strings are used by realm selection scripts to return the name of the selected realm. Attribute filter scripts use result strings to return the name of a static filter to execute.

Returning JavaScript **null** is equivalent to returning **SCRIPT_RET_SUCCESS**.

The processing of return codes by LDAP authentication scripts is more complicated than for other script types. For information about choosing the LDAP authentication script return code, see [“Choosing the Return Code” on page 885](#).

Initializing Reusable Data Objects

In JavaScript, when you declare a variable using the **var** keyword, that variable is allocated temporarily on the program stack. When your script returns, the variable goes out of scope and is marked for garbage collection by the script engine. However, variables declared without a **var** keyword become properties of the script engine's Global object and persist across invocations of the script.

To avoid allocating and deallocating data objects each time a script runs, you can create initialization blocks to allocate reusable global objects the first time a script runs. These objects remain available to use in subsequent executions of the script. This code example shows this technique.

```
// Initialization block
if (!this.initialized) {
    // Create persistent data as global object properties.
    filter = new AttributeFilter();
    accessor = new DataAccessor();
    someString = "This is a persistent string"

    // Set initialized flag for next time.
    initialized = true;
}
else {
    // /Clear left-over data from prior request.
    accessor.Clear();
}
```

In this example, the variable **initialized** is a global flag that is tested each time the script runs. If **initialized** is not set, the persistent data objects are allocated and the flag is set. On subsequent calls to the script, the initialization block is skipped but a call is made to clear the prior contents of the Data Accessor.

General Recommendations

Follow these recommendations when developing Steel-Belted Radius Carrier scripts.

- JavaScript comments begin with `//` or `/*`. Within the [Script] section, lines starting with a semicolon (`;`) are not treated as comments. In these example, compilation fails with a syntax error because the commented-out [Failure] section is passed to the JavaScript compiler.

```
[Script]
SbrWriteToLog("hello, world");
return SCRIPT_RET_SUCCESS;

;[Failure]
;SomeString
```

- Thoroughly test all scripts for speed and monitor the performance of Steel-Belted Radius Carrier after you deploy a new script. In general, the performance impact of a script on the server is directly related to its complexity.
- After you modify a realm selection or attribute filter script, you can reload it by executing a platform specific **HUP** command. It is not necessary to restart the server. You cannot use the **HUP** command to reload LDAP scripts.

Saving the Script File

You must save realm selection and attribute filter script (.jsi) files in the **scripts** subdirectory of the Steel-Belted Radius Carrier **/radiusdir** directory. When you refer to a .jsi file using the **Script** keyword in a Steel-Belted Radius Carrier configuration file, the script engine automatically searches the **scripts** subdirectory for the specified file. If the file is not found, an error message appears in the log and the associated script functionality is disabled.

You must save the **ldapauth.aut** file in the RADIUS **/radiusdir** directory, whether or not LDAP scripting is enabled.

Sample Script

This is an example of a simple realm selection script and its configuration settings.

1. To test this script, copy these lines to a new text file named **SampleScript.jsi** in the server's **\radiusdir\scripts** directory.

```
[Settings]
LogLevel = 2
ScriptTraceLevel = 1

[Script]
// Print a message in the SBR log.
SbrWriteToLog("Executing SampleScript.jsi" );

// Allocate a new Realm Selector object.
var selector = new RealmSelector();

// Invoke the built-in Suffix realm selection method to obtain
// the realm for the request.
var realm = selector.Execute("suffix" );
```

```

SbrWriteToLog("Suffix method returned '" + realm + "'");

// Print a trace frame to the log.
SbrTrace();

//Return the realm name as the script result.
return realm;

[ScriptTrace]
var = realm

[Failure]
DefaultRealm

```

2. Next, edit the `\radiusdir\proxy.ini` file.

In the [Processing] section at the top of the file, you see this comment line:

```
;Script <RealmScript>
```

Remove the “;” and replace <RealmScript> with the actual file name (omit the file extension):

```

[Processing]
Suffix
Prefix
DNIS
Attribute-Mapping
Script SampleScript

```

NOTE: Use only the script file base name only when configuring the **Script** setting. If you specify the .jsi extension, Steel-Belted Radius Carrier fails to load the file.

3. If you wish to check the script syntax and environment, set up and run the [“scriptcheck” on page 879](#) utility, discussed in [“scriptcheck” on page 879](#).
4. Restart the Steel-Belted Radius Carrier process.

When the server starts, log messages should indicate that the script loaded and is ready to run:

```

Loading script from file 'C:\radius\Service\scripts\SampleScript'
Extended Proxy: Enabled precedence 4 processing for Script SampleScript

```


5. When a RADIUS request is received, the script executes and messages similar to these appear in the log:

```
Executing 'SampleScript'  
Suffix method returned 'realm1'  
*** Script Trace (C:\radius\Service\scripts\SampleScript)  
    (line 41) SbrTrace();  
    realm=realm1  
CreateRequestEx: using virtual realm realm1 for authentication.
```

NOTE: The actual realm name returned by the script depends on the Steel-Belted Radius Carrier configuration and the suffix decoration of the username specified in the RADIUS request.

For detailed information about the **proxy.ini** file, see *Realm Configuration Files* of the *SBR Carrier Reference Guide*.

Debugging Scripts

IN THIS CHAPTER

- [SbrWriteToLog\(\) | 876](#)
- [SbrTrace and ScriptTraceLevel | 877](#)
- [scriptcheck | 879](#)

This chapter describes the tools and techniques you can use to debug your scripts and monitor their execution. Each debugging tool is discussed in its own topic. This chapter contains these topics:

SbrWriteToLog()

To write text messages to the Steel-Belted Radius Carrier log, call the **SbrWriteToLog()** function from the script.

The log message appears if the server log level is greater than or equal to the message log level. The server log level is determined by the **LogLevel** parameter of the [Configuration] section in **radius.ini**. The message log level is determined by the **LogLevel** parameter in the [Settings] section of the script file, as shown in the fragment from the **FilterScript.jsi** below. For more information, see “[Settings] Section” on page 867.

```
[Settings]
LogLevel = 2
ScriptTraceLevel = 1

[Script]
// Print a message in the SBR log.
SbrWriteToLog("Executing attribute filter script file 'FilterScript.jsi' ");
```

You can also use an optional **LogLevel** argument to specify the log level explicitly in the call to **SbrWriteToLog()**:

```
SbrWriteToLog("This is an INFORMATIONAL level message", 1);
```

In this example, the message log level is set to the value of **1**. The message appears in the log if the server log level is **1** or greater.

For more information about the **SbrWriteToLog()** function, see [“Logging and Diagnostic Methods” on page 950](#). For more information about the server LogLevel setting, see the **radius.ini** file in the *SBR Carrier Reference Guide*.

SbrTrace and ScriptTraceLevel

You can use the script trace feature to set line-by-line debugging of your script as it executes. A script trace is a block of program status information written to the Steel-Belted Radius Carrier server log file before the execution of a JavaScript statement. Information in each script trace frame includes:

- The name of the **.jsi** or **.aut** file in which the script is defined
- The current line number
- The text of the JavaScript statement at that line number
- Listings of selected program variable values
- Listings of selected RADIUS attribute values (for LDAP scripts only)

You define the names of program variables and RADIUS attributes to be displayed in script traces by entering them in the [ScriptTrace] section of the JavaScript initialization file. For more details, see [“\[ScriptTrace\] Section” on page 869](#).

You have two options to enable tracing of your scripts.

- **Manual tracing**—You can set the **ScriptTraceLevel** parameter in the [Settings] section of the script file to 1 and call the **SbrTrace()** function from within your script. This causes a single script trace frame to appear in the log from the point in your script where the **SbrTrace()** function was called.
- **Automatic tracing**—You can set the **ScriptTraceLevel** parameter in the [Settings] section of the script file to 2 to enable automatic tracing. In this mode, a script trace is performed every time that a JavaScript statement is executed by your script.

NOTE: Enabling script tracing for a single script file has a performance impact on all scripts running on Steel-Belted Radius Carrier, whether or not script tracing is enabled for those files. For this reason and because of large volume of log information produced, the use of script tracing is not recommended for production environments.

This example lists a small script and a portion of the automatic script trace generated from it.

```
[Script]
var a = 1;
var s = "Hello";
return SBR_RET_SUCCESS;

[ScriptTrace]
attr = User-Name
var = a
var = s
.
.
.

*** Script Trace (c:\radius\service\ldapauth.aut)
(line 1) var a = 1;
User-Name = testuser
a = <not found>
s = <not found>

*** Script Trace (c:\radius\service\ldapauth.aut)
(line 2) var s = "Hello";
User-Name = testuser
a = 1
s = <not found>

*** Script Trace (c:\radius\service\ldapauth.aut)
(line 3) return SBR_RET_SUCCESS;
User-Name = testuser
a = 1
s = Hello
.
.
.
```

Traces are produced just before execution of the JavaScript statement referenced in the trace. For example, the value of variable `a` is not reflected in the trace on line 1, but appears in the trace on line 2, after the assignment statement has executed. If a variable or attribute has not yet been assigned, or if a variable is out of scope at the time of the trace, the value is displayed in the log as **<not found>**.

NOTE: The **attr** keyword is supported only for LDAP authentication scripts. If this example is configured as a realm selection or attribute filter script, the **attr = User-Name** entry in the [ScriptTrace] section is ignored.

Script traces appear in the log if the server log level is greater than or equal to the trace log level. The server log level is determined by the **LogLevel** parameter of the [Configuration] section in **radius.ini**. The trace log level is determined by the **LogLevel** parameter in the [Settings] section of the script file. For more information, see “[Settings] Section” on page 867.

You can also use the optional **LogLevel** argument to specify the log level explicitly in the call to **SbrTrace()**:

```
SbrTrace(0); //Specify production log level
```

In this example, the argument trace appears unconditionally in the server log regardless of the script file **LogLevel** setting.

For more information about the **SbrTrace** function, see “Logging and Diagnostic Methods” on page 950. For more information about the server LogLevel setting, see the **radius.ini** file in the *SBR Carrier Reference Guide*.

scriptcheck

The **scriptcheck** utility is a command-line application that enables you to check the Steel-Belted Radius Carrier JavaScript configuration files for syntax errors.

NOTE: The **scriptcheck** utility verifies that your script is syntactically correct. The **scriptcheck** utility does not guarantee that your script is free of runtime errors or produces correct results. If your script does not appear to be working properly, review the Steel-Belted Radius Carrier log for error messages and enable script tracing to diagnose the problem.

Unpacking the scriptcheck Utility

The **scriptcheck** utility and its required shared libraries are packaged as a gzip compressed tar file for Solaris - **scriptcheck.version.sol.tgz**, in the **Support_Files/scriptcheck** directory on the Steel-Belted Radius Carrier installation CD.

You can copy the appropriate **scriptcheck** executable version to any convenient location and run it there, provided that you also copy the **radius.lic** file to the same location.

Before you can run the **scriptcheck** utility, you must unpack the correct version of the archive for your platform into a destination folder or directory. With the archive in the destination directory, enter this command to extract its contents:

```
% gunzip -c scriptcheck.version.platform.tar.gz | tar xvf -
```

The files to extract are:

- **scriptcheck**
- **libnspr4.so**
- **libjs.so**
- **README.SCRIPTCHECK**

Running the scriptcheck Utility

To run the **scriptcheck** utility and verify scripts, follow these steps:

1. Before running the **scriptcheck** utility for Solaris, set the **LD_LIBRARY_PATH** environment variable to point to the location where the shared object files are installed.
2. Open a command shell (**cmd**) and change to the **scriptcheck** directory.
3. Execute the **scriptcheck** command by specifying the name of the script as the argument. For example, this command validates the script contained in the **myscript.jsi** file:

```
% cd scriptcheck
% setenv LD_LIBRARY_PATH .
% scriptcheck myscript.jsi
Loading script from file 'myscript.jsi'
Scriptcheck: script file 'myscript.jsi' compiled successfully
```

When the **scriptcheck** utility runs, it loads the [Script] section in the specified .jsi file and uses the JavaScript interpreter to compile the script text. Any error messages produced during script compilation are printed on the console. You can then correct the errors and rerun **scriptcheck** to verify that the script compiles correctly.

Creating LDAP Scripts

IN THIS CHAPTER

- [LDAP Basics | 881](#)
- [LDAP Request Life Cycle | 882](#)
- [Unscripted LDAP Searches | 883](#)
- [LDAP Script Basics | 884](#)
- [Choosing the Return Code | 885](#)
- [LDAP Script Return Codes | 887](#)
- [LDAP Script Examples | 888](#)

This chapter describes how to write scripts that implement authentication using the Lightweight Directory Access Protocol (LDAP). This chapter contains these topics:

LDAP Basics

Many companies use Lightweight Directory Access Protocol (LDAP) directory servers to store user authentication and authorization information. Steel-Belted Radius Carrier can process authentication requests against records stored in one or more external LDAP databases.

LDAP scripting is used when more sophisticated decision logic or attribute manipulation is required than can be implemented using unscripted searches. Incorporating JavaScript into the Steel-Belted Radius Carrier **ldapauth.aut** file gives you much greater flexibility in the processing of LDAP authentication queries. Scripted authentication enables a level of control comparable to SQL stored procedures.

For example, LDAP scripts can combine data from several LDAP queries and analyze the results to determine which query to invoke next. LDAP scripts can evaluate loops and complicated if-then-else logic, build up RADIUS attribute value strings from scratch, and write status messages to the Steel-Belted Radius Carrier log.

NOTE: LDAP scripting does not support `DataAccessors()`. There is no way to use a data accessor to query a SQL database from within the LDAP authentication plug-in, so trying to execute `DataAccessor()` from an LDAP script causes a runtime exception.

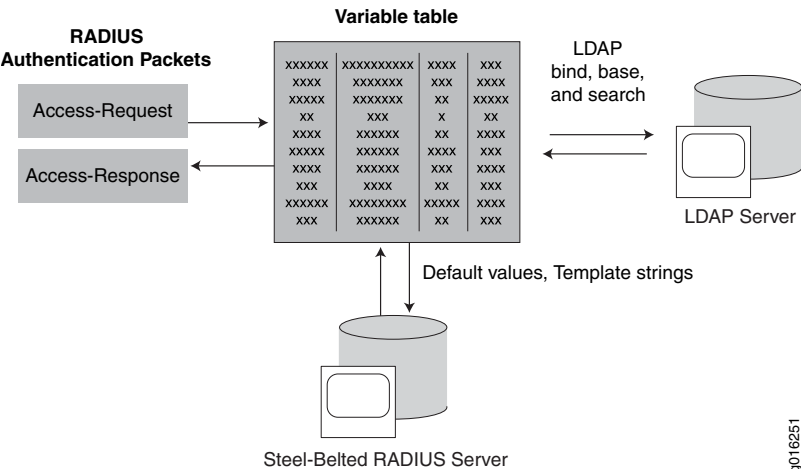
LDAP Request Life Cycle

Steel-Belted Radius Carrier performs these steps in response to an LDAP authentication request for both scripted and non-scripted configurations.

1. At the beginning of each LDAP authentication request, Steel-Belted Radius Carrier creates a variable table to map RADIUS access-request attributes to LDAP attributes for use in LDAP Bind, Base, and Search strings. The [Request] section of the LDAP plug-in configuration file is used to select which attributes are extracted from the incoming request and placed in the variable table.
2. Steel-Belted Radius Carrier performs one or more LDAP searches. Parameters for each search are given in the Search/name] sections of the configuration file. After a search is performed, selected attributes are copied from the LDAP response and placed in the variable table.
3. Steel-Belted Radius Carrier uses the [Response] section to select information from the variable table to be returned to the RADIUS client in the RADIUS response packet.

Figure 277 on page 882 shows how the LDAP variable table is populated with information coming from a RADIUS access-request message, default values, and the results of LDAP Bind, Base, and Search requests. The information in the variable table is then used to format the access-response packet that is returned to the RADIUS client.

Figure 277: Role of the Variable Table in LDAP Authentication

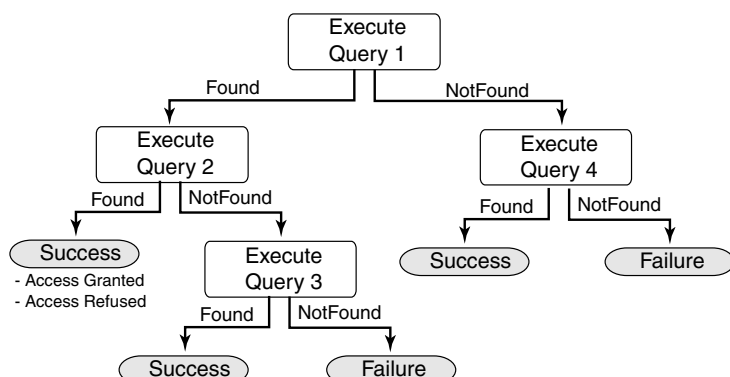


g016251

Unscripted LDAP Searches

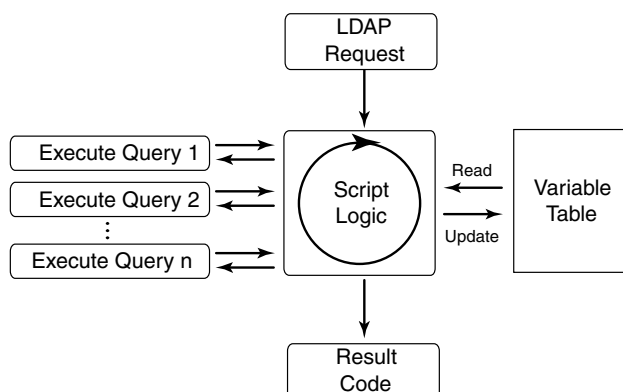
Scripting is not required for basic applications of LDAP authentication. In unscripted configurations, search parameters such as base Distinguished Names (DNs), filter strings, and attribute maps are configured in the `ldapauth.aut` file. Using the **OnFound** and **OnNotFound** settings of the [Search/name] sections, you can configure a decision tree in which the result of one LDAP query (**Found** or **Not Found**) determines whether another query is executed or the final authentication decision is returned to Steel-Belted Radius Carrier. The basic query tree provides sufficient control to meet the needs of many LDAP authentication applications. [Figure 278 on page 883](#) shows a sample query tree using unscripted branching.

Figure 278: Query Tree with Unscripted Branching



[Figure 279 on page 884](#) shows the data flow involved in a scripted query. Instead of following a rigid branch structure, the request is processed according to the logic of the LDAP script, which might be arbitrarily complex. The script executes one or more LDAP queries, computes intermediate results from the return values, updates the LDAP variable table, and possibly executes additional queries against the LDAP server. Once the script has completed processing the request and made an authentication decision, it returns a result code to the plug-in.

Figure 279: Scripted Query Data Flow



LDAP Script Basics

To configure LDAP scripting, you add JavaScript instructions to the [Script] section of the **ldapauth.aut** file. You can perform these operations in your LDAP scripts:

- Get, set, and reset values of variables stored in the LDAP variable table
- Invoke LDAP queries defined in the [Search/name] sections of the **ldapauth.aut** file
- Write diagnostic messages and script traces to the Steel-Belted Radius Carrier log
- Evaluate arbitrary program logic coded in your script
- Exit the script and return a result code string to the LDAP plug-in

When Steel-Belted Radius Carrier starts, it reads the text of the section from **ldapauth.aut** and passes it as a block to the JavaScript interpreter, which compiles it into bytecodes. The bytecodes are stored for execution during subsequent LDAP authentication requests. If syntax errors are detected in the JavaScript text, the script does not compile and the LDAP authentication plug-in is disabled. Any error messages generated during script compilation appear in the Steel-Belted Radius Carrier log file.

You can use the **scriptcheck** utility to check your LDAP scripts for syntax errors without having to start Steel-Belted Radius Carrier. For more information, see [“scriptcheck” on page 879](#).

Working with the Variable Table

You configure the variable table for scripting the same way you do for unscripted configurations. Input RADIUS attributes that the script manipulates must be identified in the [Request] section of the **ldapauth.aut**

file. Output RADIUS attributes that the script manipulates must be identified in the [Response] section of the **ldapauth.aut** file.

The **LdapVariables** object is available to your script for manipulating attributes in the variable table. The **LdapVariables** object exposes three methods that scripts can call:

- **LdapVariables.Get()** retrieves the current value or values for a variable stored in the LDAP variable table.
- **LdapVariables.Add()** creates a new variable or adds a value to an existing variable.
- **LdapVariables.Reset()** deletes all of the values of the specified variable.

Invoking LDAP Queries

Any query defined in a [Search/name] section of **ldapauth.aut** can be invoked programmatically by an LDAP script. Use the **Ldap.Search()** method to invoke the query, giving the name of the query as the argument to the method.

As with unscripted searches, you can identify a set of LDAP attributes to be extracted from the LDAP response and placed in the variable table. You do this by creating an [Attributes/name] section in the **ldapauth.aut** file and specifying this section with the **Attributes** parameter in the query definition.

For more information about LDAP attributes, refer to the section on the *LDAP Authentication File* in the *SBR Carrier Reference Guide*.

Writing to the Steel-Belted Radius Carrier Log

Use the **SbrWriteToLog()** function to insert diagnostic or informational text strings into the Steel-Belted Radius Carrier log file. You can use the optional level argument to control the log level visibility of your message.

Use the **SbrTrace()** function to display trace information about your script in the Steel-Belted Radius Carrier log.

For more details about these functions, see [“Logging and Diagnostic Methods” on page 950](#).

Choosing the Return Code

When a script finishes running, it sends a return value back to the LDAP plug-in. Depending on the return value and the state of the request, the plug-in can do one of several things:

- It can make an authentication decision and send that result directly to Steel-Belted Radius Carrier, ending the processing of that request by the plug-in.

- It can re-execute the script against a different LDAP server and process the new return value when the script is finished.
- It can perform failure processing and return a result to Steel-Belted Radius Carrier based on the [Failure] section in `ldapauth.aut`.

For information about configuring other LDAP authentication plug-in settings, see the section on the *LDAP Authentication File* in the *SBR Carrier Reference Guide*.

An LDAP script may execute several times while handling a single authentication request but eventually the LDAP plug-in must make an authentication decision and send it back to the Steel-Belted Radius Carrier server. It is important for the script programmer to understand exactly how the script return code affects the LDAP plug-in and the authentication decision.

Script Return Codes

You specify the script return code as an argument to the JavaScript **return** statement. The return code must be one of the global constants.

NOTE: The Steel-Belted Radius Carrier pre-6.0 release **SBR_RET_xxx** codes have been deprecated and replaced with the new **SCRIPT_RET_xxx** codes. The **SBR_RET_xxx** codes are supported for backward compatibility.

SCRIPT_RET_SUCCESS

The **SCRIPT_RET_SUCCESS** code indicates to the LDAP plug-in that the user has been authenticated and should be accepted. The plug-in finishes processing the request and sends an accept decision to the Steel-Belted Radius Carrier core.

SCRIPT_RET_DO_NOT_AUTHENTICATE

The **SCRIPT_RET_DO_NOT_AUTHENTICATE** code indicates to the LDAP plug-in that a hard reject should be performed by the server. The plug-in finishes processing the request and sends a reject decision to the Steel-Belted Radius Carrier core.

SCRIPT_RET_TRY_NEXT_AUTH_METHOD

The **SCRIPT_RET_TRY_NEXT_AUTH_METHOD** code indicates that the LDAP plug-in should stop processing the request and ask Steel-Belted Radius Carrier to try the next authentication method without immediately rejecting the user. Last resort processing is not performed.

SCRIPT_RET_NOT_AUTHENTICATED

The **SCRIPT_RET_NOT_AUTHENTICATED** code indicates to the LDAP plug-in that the script could not authenticate the user. If a last resort server is defined, the LDAP plug-in re-executes the script against that server. If there is no last resort server, this return code has the same effect as **SCRIPT_RET_TRY_NEXT_AUTH_METHOD**.

SCRIPT_RET_FAILURE

The **SCRIPT_RET_FAILURE** code indicates to the LDAP plug-in that a communication failure with the LDAP server occurred. The plug-in should re-execute the script against the next LDAP server in the configuration, if defined. If only one server is defined or the last server has already been tried, the LDAP plug-in should process the [Failure] section to determine the final result. If there is no [Failure] section, this return code has the same effect as **SCRIPT_RET_TRY_NEXT_AUTH_METHOD**.

SCRIPT_RET_INVALID_CODE

The **SCRIPT_RET_INVALID_CODE** code indicates to the LDAP plug-in that a script execution failure has occurred due to an invalid operation. The plug-in should re-execute the script against the next LDAP server in the configuration, if defined. If only one server is defined or the last server has already been tried, the LDAP plug-in should process the [Failure] section to determine the final result. If there is no [Failure] section, this return code has the same effect as **SCRIPT_RET_TRY_NEXT_AUTH_METHOD**.

LDAP Script Return Codes

[Table 126 on page 887](#) provides a list of the LDAP script return codes.

Table 126: LDAP Script Return Codes

Script Return Code	Action	Plug-in Return Code
SBR_RET_SUCCESS	Accept the user.	SBR_RET_SUCCESS
SBR_RET_DO_NOT_AUTHENTICATE	Hard reject. Do not invoke another authentication method.	SBR_RET_DO_NOT_AUTHENTICATE
SBR_RET_TRY_NEXT_AUTH_METHOD	Return from the LDAP plug-in and invoke the next authentication method. Do not process [Failure] section or try last resort server.	SBR_RET_NOT_AUTHENTICATED

Table 126: LDAP Script Return Codes *(continued)*

Script Return Code	Action	Plug-in Return Code
SBR_RET_FAILURE	A communication error occurred. Retry script with next server in list, or go to [Failure] section if no server is available.	If another server is available, the plug-in return code depends on the script return code when the script is re-executed. If no server is available, process [Failure] section and return SBR_RET_SUCCESS or SBR_RET_NOT_AUTHENTICATED depending on configuration.
SBR_RET_NOT_AUTHENTICATED	Retry script with last resort server, if defined. Otherwise, go to the next authentication method.	If LastResort is defined, the plug-in return code depends on the script return code when the script is re-executed. If LastResort is not defined, return SBR_RET_NOT_AUTHENTICATED .

LDAP Script Examples

Example 1: Simple Authentication

This script executes the search criteria specified in the [Search/LdapSearch1] section of the **ldapauth.aut** file. If the search is unsuccessful, the script prepends **myco.** to the username and executes the search criteria specified in the [Search/LdapSearch2] section.

```
[Script]
// Try the initial query.
Status = Ldap.Search('LdapSearch1');
if (status == Ldap.NOTFOUND) {
    // Add "myco." to the username and run new query.
    userName = LdapVariables.Get('User-Name');
    LdapVariables.Reset('User-Name');
    LdapVariables.Add('User-Name', 'myco.' + userName);
    status = Ldap.Search('LdapSearch2');
}
```

```
// Return value depends on final search status.
switch (status) {
    case Ldap.FOUND:
        return SBR_RET_SUCCESS;
    case Ldap.NOTFOUND:
        return SBR_RET_NOT_AUTHENTICATED;
    default:
        return SBR_RET_FAILURE;
}
```

Example 2: Profile Assignment

Scripts can use authentication information to determine the profile that should be assigned to a user. In this example, the script executes the query specified in the [Search/Radius] section. This query looks up an object named **ProfileData** that contains multiple instances of the **radiusattrs** attribute. The script iterates through the returned values of **radiusattrs**, looking for the first instance that begins with the prefix **sbr-**. If a matching attribute is found, the prefix is stripped from the attribute and returned as the name of the user profile.

This is the LDIF representation of the **ProfileData** object, showing the values of the **radiusattrs** attributes:

```
dn: name=ProfileData, ou=radius, dc=funk,dc=com
name: ProfileData
objectClass: top
objectClass: radiusobject
radiusattrs: attr1
radiusattrs: attr2
radiusattrs: sbr-defaultprofile
radiusattrs: attr3
```

The relevant sections of the **ldapauth.aut** file are shown below.

```
[Attributes/RadiusAttrs]
radiusattrs

[Response]
%Profile = Return-Profile

[Search/Radius]
Base = ou=radius,dc=funk,dc=com
```

```

Scope = 2
Filter = name=ProfileData
Attributes = RadiusAttrs
Timeout = 20
%DN = dn

[Script]
// Look up "ProfileData" object using the "Radius" query.
if (Ldap.Search("Radius") == Ldap.FOUND) {
    var attr = "";
    var profile = "default";

    // Loop through all "radiusattrs" attributes.
    for(i = 0; attr != null; i++) {
        attr = LdapVariables.Get("radiusattrs", i);
        // If prefix matches "sbr-" extract profile name.
        if ((attr != null) && (attr.substr(0, 4) == "sbr-")) {
            profile = attr.substr(4);
            break;
        }
    }

    // Add profile name to the variable table and return.
    LdapVariables.Add("Return-Profile", profile);
    return SBR_RET_SUCCESS;
}

// Object wasn't found, so signal a failure.
return SBR_RET_FAILURE;

```

Example 3: Received Attribute Normalization

Users frequently need to normalize incoming RADIUS attributes to a common format before performing an LDAP search. This example checks the length of the telephone number string in the Calling-Station-ID attribute, preserving only the final seven digits, if necessary. The truncated telephone number is saved as a new entry (**Stripped-CSID**) in the variable table. The value of **Stripped-CSID** is specified as part of the Filter parameter in the [Search/Query1] query definition. This query is executed by the script, and the resulting status code determines the script return code.

```

[Request]
%UserName = User-Name
Calling-Station-Id = Received-CSID

```



```

[Search/Query1]
Base=ou=people,dc=funk,dc=com
Scope = 2
Filter = (&(uid=<User-Name>)(callingStationId=<Stripped-CSID>))
Timeout = 20
%DN = dn

[Script]
// Get the received Calling-Station-ID attribute.
var csid = LdapVariables.Get("Received-CSID");

// Check length and retain last seven digits of CSID.
var length = csid.length;
if (length > 7) {
    csid = csid.substr(length - 7);
    SbrWriteToLog("Shortened CSID to: " + csid);
}

// Save result to variable table so we can search on it.
LdapVariables.Add("Stripped-CSID", csid);

// Perform the search with normalized CSID.
var status = Ldap.Search("Query1");

// Generate return code based on search result.
if (status == Ldap.FOUND) {
    return SBR_RET_SUCCESS;
}
return SBR_RET_NOT_AUTHENTICATED;

```

Example 4: Conditional Profile Assignment from User Attribute

This example illustrates how you can use LDAP scripts to implement multiple queries and complex decision logic. The script starts by invoking the FindUser query to look up the specified user in the LDAP repository. Depending on the **employeetype** attribute returned from the first query, a second query is selected and invoked to retrieve attributes specific to the user's employee type. Finally, the **Radius-Profile** attribute of the employee type record is returned as the profile name for the authentication response.

The LDIF data for a sample user is as follows:

```

dn: uid=SStudent, ou=People, dc=funk,dc=com
employeeType: Student
uid: SStudent

```

```

userPassword:: e1NTSEF9cTZvdFFOYXArcFowaG5rOWJQU3dZYIExbkFIL1doMXBnMIR4
==
objectClass: Sam
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
sn: Student
cn: Sam Student

```

The data objects holding the “Radius-Profile” attributes associated with each employee type are retrieved:

```

dn: ou=radius, dc=funk,dc=com
ou: radius
objectClass: top
objectClass: organizationalunit

dn: name=VendorType, ou=radius, dc=funk,dc=com
Radius-Profile: Vendor-Profile
name: VendorType
objectClass: top
objectClass: radius

dn: name=FacultyType, ou=radius, dc=funk,dc=com
Radius-Profile: Faculty-Profile
name: FacultyType
objectClass: top
objectClass: radius

dn: name=StudentType, ou=radius, dc=funk,dc=com
Radius-Profile: Student-Profile
name: StudentType
objectClass: top
objectClass: radius

```

Finally, here are the configuration settings and the LDAP search script:

```

[Request]
%UserName = User-Name

[Response]

```

```
%Profile = Radius-Profile
>Password = userpassword
```

```
[Attributes/UserAttributes]
employeetype
userpassword
```

```
[Attributes/TypeAttributes]
radius-profile
```

```
[Search/FindUser]
Base=ou=people,dc=funk,dc=com
Scope = 2
Filter = uid=<User-Name>
Attributes = UserAttributes
Timeout = 20
%DN = dn
```

```
[Search/Student]
Base=ou=radius,dc=funk,dc=com
Scope = 2
Filter = name=StudentType
Attributes = TypeAttributes
Timeout = 20
```

```
[Search/Faculty]
Base=ou=radius,dc=funk,dc=com
Scope = 2
Filter = name=FacultyType
Attributes = TypeAttributes
Timeout = 20
```

```
[Search/Vendor]
Base=ou=radius,dc=funk,dc=com
Scope = 2
Filter = name=VendorType
Attributes = TypeAttributes
Timeout = 20
```

```
[Script]
// Look up the specified user in the LDAP repository.
var status = Ldap.Search("FindUser");
if (status != Ldap.FOUND) {
    return SBR_RET_NOT_AUTHENTICATED;
```

```

}

// Get the employeetype attribute from the query result.
var type = LdapVariables.Get("employeetype");

// Execute query to look up employee type object.
switch (type) {
    case "Student":
        status = Ldap.Search("Student");
        break;
    case "Faculty":
        status = Ldap.Search("Faculty");
        break;
    case "Vendor":
        status = Ldap.Search("Vendor");
        break;
    default:
        SbrWriteToLog("Invalid employee type: " + type);
        return SBR_RET_DO_NOT_AUTHENTICATE;
}

// This error should never happen.
if (status != Ldap.FOUND) {
    SbrWriteToLog("No record for employee type: " + type);
    return SBR_RET_DO_NOT_AUTHENTICATE;
}

// Get the profile name for this employee type.
var profile = LdapVariables.Get("profilename");
if (profile == null) {
    profile = "Default-Profile";
}

// Save profile name to variable table and return.
LdapVariables.Add("Radius-Profile", profile);
return SBR_RET_SUCCESS;

```

Creating Realm Selection Scripts

IN THIS CHAPTER

- Realm Selection Script Functions | 896
- Enabling Built-In Realm Selection Methods | 896
- Choosing the Return Code | 897
- Configuring Realm Selection Scripts | 898
- Realm Selection Script Examples | 902

This chapter describes developing realm selection scripts that execute built-in methods or more advanced script logic to authenticate requests. This chapter contains these topics:

Steel-Belted Radius Carrier executes built-in or scripted realm selection methods to determine the authentication realm for processing a request. For built-in methods, you specify the methods and their order of execution in the [Processing] section of the **proxy.ini** configuration file. You specify matching rules in the [Realms] and [Directed] sections. For more information about the **proxy.ini** configuration file, see the section on *Realm Configuration Files* in the *SBR Carrier Reference Guide*.

For scripted realm selection, use the **script** setting in **proxy.ini** to declare the name of a JavaScript initialization (.jsi) file. If the **script** setting appears anywhere in the [Processing] section, Steel-Belted Radius Carrier executes the realm selection script first, before trying any other built-in methods. If the script returns a valid realm name, Steel-Belted Radius Carrier sends the current request to that realm for processing. If the script returns the code **SCRIPT_RET_SUCCESS** instead of a realm name, Steel-Belted Radius Carrier invokes the remaining methods in the [Processing] section to try to determine the realm for the request.

You can also specify a realm selection script for the inner authentication setting of tunneled authentication methods using Web GUI.

Realm selection scripts are useful when your realm selection strategy is too complex to be implemented using basic matching rules. Realm selection scripts can perform any of these actions:

- Retrieve RADIUS request attribute and process their values.
- Execute program logic to determine the realm name.
- Execute built-in Steel-Belted Radius Carrier realm selection methods.

- Invoke SQL queries and LDAP searches, and process the results.
- Specify a profile to be merged with the response.
- Change the authentication username.

Realm Selection Script Functions

Realm selection scripts can execute any standard JavaScript statements and functions. Additionally, attribute filter scripts use the **RealmSelector** API to perform operations specific to realm selection. You must instantiate a **new RealmSelector** object instance and then use it to invoke these methods. For example:

```
var selector = new RealmSelector();
var realm = selector.Execute("suffix");
```

These six **RealmSelector** methods are provided:

- **new RealmSelector()**—Creates a new **RealmSelector()** object instance.
- **Execute()**—Executes a built-in realm selection method and returns the resulting realm name.
- **SetAuthUserName()**—Sets the authentication username for the request.
- **SetAuthProfile()**—Sets the name of a profile to be merged with the result after the user is accepted.
- **CSTAccessor**—Reads field information from the Current Sessions database.
- **SessionControl**—Allows JavaScript to initiate session management and control capabilities.

Realm selection scripts can also execute the **AttributeFilter.Get()** method and any of the methods of the **DataAccessor** class.

For more details about the **RealmSelector** functions and methods, see [“RealmSelector Object” on page 956](#).

The **CSTAccessor** object supports the **Get()** method and **SessionControl** object supports **AddAttribute()** and **Execute()** methods.

Enabling Built-In Realm Selection Methods

You can call the **RealmSelector.Execute()** method from realm selection scripts to execute built-in Steel-Belted Radius Carrier realm selection methods. The argument to the **Execute()** method is one of the standard Steel-Belted Radius Carrier realm selection method names (**suffix**, **prefix**, **dnis**, **attribute-mapping**, or **undecorated**). The return value is a string containing the name of the realm selected by that method, using

the matching rules defined in **proxy.ini** and applied to the username in the current RADIUS request. If no realm is selected, the **Execute()** method returns null.

The matching realm must be declared in **proxy.ini** and the corresponding **.dir** or **.pro** file must exist, or the **Execute()** method returns **null**. This is true even if the current username contains a valid realm name decoration for the specified realm selection method.

If you call the **Execute()** method with the **suffix** or **prefix** argument, you must make sure that the corresponding **suffix** or **prefix** realm selection methods are enabled or the result is always null. The **suffix** and **prefix** methods are enabled by default when you declare a script in the [Processing] section of **proxy.ini**. If you declare a realm selection script in the [Inner_Authentication] section of a tunneled authentication plug-in **.aut** file, the **suffix** and **prefix** methods are not enabled by default. To use the **suffix** and **prefix** methods with the **Execute()** function, you must either declare these methods explicitly, or declare a script in the [Processing] section of **proxy.ini**.

A realm selection script can call built-in realm selection methods at any time during its execution. Depending on its program logic, the script might return the realm name produced by the built-in method as its result or continue processing and return a different result.

Choosing the Return Code

When a realm selection script is finished executing, it returns a result code to the Steel-Belted Radius Carrier core. The return code is a string containing the name of the realm selected to process the current request. If the script is unable to select a realm, it might return a success or failure code instead of a realm string.

Valid return codes are:

- **<RealmName>**—Steel-Belted Radius Carrier should send the current request to the proxy or directed realm given by **RealmName**.
- **SCRIPT_RET_SUCCESS**—The realm selection script executed successfully, but did not select a realm. Steel-Belted Radius Carrier should try the remaining realm selection methods in the [Processing] section of the **proxy.ini** file or process the request in the default realm.
- **SCRIPT_RET_FAILURE**—The realm selection script failed to execute successfully. Steel-Belted Radius Carrier should terminate request processing and reject the user.

NOTE: Do not return the name of an undefined realm from a realm selection script; a runtime error occurs.

Configuring Realm Selection Scripts

You can configure realm selection scripts using either of these two methods:

- For core realm selection—Core realm selection occurs first for all RADIUS requests. Add the **script** keyword to the [Processing] section of the **proxy.ini** file and specify the base filename of the realm selection script file as its argument.
- For tunneled authentication methods (PEAP and TTLS)—Using the Web GUI, specify a realm selection script from the **Inner Authentication** tab of the **Selected EAP Method** pane for the authentication method.

NOTE: For both realm selection script configuration methods, do not include the **.jsi** extension when you enter or specify the name of the script file.

Core Realm Selection Scripts

To configure core realm selection, you configure realm selection scripts in the [Processing] section of **proxy.ini**. All authentication requests go through this phase even if a second realm selection script is run from a tunneled authentication method.

When scripted realm selection is configured in **proxy.ini** from the [Processing] section, it runs before (and possibly replaces) all other realm selection methods.

[Processing] Section

If no [Processing] section is present in the **proxy.ini** file, then the standard methods are applied following this specific default order: **Suffix**, **Prefix**, **DNIS**, **Attribute-mapping**, and **Undecorated**.

If a [Processing] section ([Table 127 on page 899](#)) is present in the **proxy.ini** file, it enables you to specify which realm selection rules are applied and the order in which they are applied.

```
[Processing]
RealmSelector
```

```
·
·
·
```


Table 127: proxy.ini [Processing] Syntax

Parameter	Description
RealmSelector	<p>This can be one of six methods: attribute-mapping, DNIS, prefix, suffix, undecorated, or script <i>scriptname</i>. These are case-insensitive; except for the script file rootname is case-sensitive.</p> <p>If a [Processing] section is present in the proxy.ini file, then these special rules apply:</p> <ul style="list-style-type: none"> • If no scripts are declared in the [Processing] section, then all methods are applied using the order in which they appear in the list. • If a script is declared anywhere within the [Processing] section, then the script <i>scriptname</i> method runs first. • If a script cannot determine the realm, it might return the null keyword, an empty string, or the SCRIPT_RET_SUCCESS return code. In this case, the remaining declared methods are applied using the order in which they appear in the list.

This example shows a [Processing] section with a declared script:

```
[Processing]
Script scriptname
Suffix
Prefix
DNIS
Attribute-mapping
Undecorated
```

Matching rules for the methods are as defined in the [Realms] and [Directed] sections of **proxy.ini**.

Tunneled Authentication Plug-in Realm Selection Scripts

To specify a realm selection script for the inner authentication method of a tunneled authentication method, you must use the Web GUI.

To specify a realm selection script using the Web GUI:

1. Select **RADIUS Configuration > Authentication Policies > EAP Methods**.

The **EAP Methods List** page (Figure 280 on page 900) appears.

Figure 280: EAP Methods List Page



2. Select an EAP authentication method, for example **EAP-TTLS**. The **Selected EAP Method: EAP-TTLS** pane appears with the **Client Certification Validation** tab selected.
3. Click the **Inner Authentication** tab (Figure 281 on page 901) to specify a realm selection script for the authentication method.

Figure 281: EAP-TTLS—Inner Authentication

The screenshot shows the Steel-Belted Radius Carrier web interface. The top navigation bar includes links for Home, RADIUS Configuration, Diameter Configuration, Tools, Help, Logout, and User: root. The main content area is titled "EAP Methods List" and contains a table with the following data:

Name	Status
EAP-TLS	Disabled
EAP-TTLS	Disabled
EAP-PEAP	Disabled
EAP-TLS Helper	Disabled

Below the table, the "Selected EAP Method: EAP-TTLS" configuration panel is shown. It includes a checkbox for "Enable EAP-TTLS Method" and a tabbed interface with the following tabs: Client Certificate Validation, Request Filters, Response Filters, Session Resumption, Inner Authentication (selected), and Advanced Server Settings. The "Inner Authentication" tab contains the following fields:

- Directed Realm:
- Realm Selection Script:

At the bottom of the configuration panel are buttons for Save, Reset, and Cancel.

NOTE: When using JavaScripting, setting the disposition of an inner authentication request (for example, in TTLS) to discard does not suppress the sending of an Access-Reject by the outer request.

4. To specify a realm for the authentication method, enter the name of the realm in the **Directed Realm** field.
5. To specify a realm selection script for the authentication method, enter the name of the script in the **Realm Selection Script** field.
6. Click **Save** to save the changes.

Realm Selection Script Examples

Example 1: Querying Multiple SQL Databases

A common application of realm selection scripts is to query a database for information used to determine the realm name. If the database query fails, you can program the script to query other databases for the required information. In this example, the script defines two **DataAccessor** objects, each pointing to a different SQL database. It also defines **AttributeFilter** and **RealmSelector** objects.

When the script executes, it uses the **AttributeFilter** object to obtain the value of the **Called-Station-ID** attribute from the request. It uses this value as the key to look up a realm selection method name in the first SQL database. If the database record is not found, it retries the query on the second database.

The result of the database query is the name of a built-in Steel-Belted Radius Carrier realm selection method. The script then calls the **RealmSelector.Execute()** method to determine the realm name from the current request, which it returns as the script result.

[Table 128 on page 902](#) and [Table 129 on page 902](#) show sample data for the two databases:

Table 128: Database #1

CallStationId	RealmMethod
1111111	Suffix
2222222	Prefix
3333333	DNIS
4444444	Attribute-Mapping

Table 129: Database #2

CallStationId	RealmMethod
5555555	Suffix
6666666	Prefix
7777777	DNIS
8888888	Attribute-Mapping

Two data accessor **.gen** files are required. They are identical except for the **MethodName** and **Connect** settings.

```

[Bootstrap]
LibraryName=sqlaccessor.so
Enable=1

[Settings]
Connect=DSN=<dsn_name_here>;UID=<username_for_dB>;PWD=<password_for_dB>
ParameterMarker=?
SQL=SELECT RealmMethod FROM RealmExample1 WHERE CallStationId = @Called-Station-Id
MethodName=<RealmExample1-1 | RealmExample1-2>
MaxConcurrent=1
ConcurrentTimeout=30
ConnectTimeout=25
QueryTimeout=25
WaitReconnect=2
MaxWaitReconnect=360
PasswordFormat = 0
DefaultResults = 0

[Results]
RealmMethod= 1/16

[VariableTypes]
Called-Station-ID = string
RealmMethod = string

```

Finally, here is the script configuration file (.jsi) for this example. For script efficiency, you can use an initialization block to define persistent API objects.

```

[Settings]
LogLevel = 2
ScriptTraceLevel = 2

[Script]
// Initialization block
if (!this.initialized) {
    filter = new AttributeFilter();
    accessor1 = new DataAccessor("RealmExample1-1");
    accessor2 = new DataAccessor("RealmExample1-2");
    selector = new RealmSelector();
    initialized = true;
}
else {
    accessor1.Clear();
    accessor2.Clear();
}

```

```

}

// Get the Called-Station-ID attribute from the request.
var csid = filter.Get("Called-Station-ID");
if (csid == null) {
    SbrWriteToLog("RealmExample1: Called-Station-ID attribute is null");
    return SCRIPT_RET_FAILURE;
}
accessor1.SetInputVariable("Called-Station-ID", csid);
accessor2.SetInputVariable("Called-Station-ID", csid);

// Try one database, then the other if the first search fails.
if (accessor1.Execute() == DataAccessor.FOUND) {
    var method = accessor1.GetOutputVariable("RealmMethod");
}
else if (accessor2.Execute() == DataAccessor.FOUND) {
    var method = accessor2.GetOutputVariable("RealmMethod");
}
else {
    SbrWriteToLog("RealmExample1: SQL search failed for Called-Station-ID = " + csid + "");
    return SCRIPT_RET_FAILURE;
}

// Execute the specified realm selection method and return the result.
SbrWriteToLog("RealmExample1: Executing method " + method + "");
var realm = selector.Execute(method);
return realm;

[ScriptTrace]
var = csid
var = method
var = realm

```

Example 2: Using JavaScript to Manipulate Request Attributes

In this example, an **AttributeFilter** object is used to obtain the **User-Name** attribute from the request. JavaScript string functions are used to extract the realm **suffix** decoration from the username. The **suffix** string is concatenated with the value of the **Called-Station-Id** attribute and is used to look up the realm name in a SQL database. The database query also returns the name of a profile to be merged with the result list upon successful authentication. The **SetAuthProfile()** function is called by the script to register the profile name.

This is an example database table for use with this script.

Table 130: Database Information

KeyString	RealmName	Profile
spacely1111111	spacely-realm1	SPACELY-PROFILE
spacely2222222	spacely-realm2	SPACELY-PROFILE
spacely3333333	spacely-realm3	SPACELY-PROFILE
spacely4444444	spacely-realm4	SPACELY-PROFILE
cogswell1111111	cogswell-realm1	COGSWELL-PROFILE
cogswell2222222	cogswell-realm2	COGSWELL-PROFILE
cogswell3333333	cogswell-realm3	COGSWELL-PROFILE
cogswell4444444	cogswell-realm4	COGSWELL-PROFILE

The corresponding data accessor **.gen** file is:

```
[Bootstrap]
LibraryName=sqlaccessor.so
Enable=1
[Settings]
Connect=DSN=<dsn_name_here>;UID=<username_for_dB>;PWD=<password_for_dB>
ParameterMarker=?
SQL=SELECT RealmName, Profile FROM RealmExample2 WHERE KeyString = @Key-String
MethodName=RealmExample2
MaxConcurrent=1
ConcurrentTimeout=30
ConnectTimeout=25
QueryTimeout=25
WaitReconnect=2
MaxWaitReconnect=360
PasswordFormat = 0
DefaultResults = 0

[Results]
RealmName = 1/32
Profile = 2/32

[VariableTypes]
```

```

RealmName = string
Profile = string
Key-String = string

```

The script configuration file is as follows.

```

[Settings]
LogLevel = 2
ScriptTraceLevel = 0

[Script]

// Allocate API objects at first execution.
if (!this.initialized) {
    filter = new AttributeFilter();
    selector = new RealmSelector();
    accessor = new DataAccessor("RealmExample2");
}
else {
    accessor.Clear();
}

// Get the realm suffix from the username attribute.
var suffix = "default";
var username = filter.Get("User-Name");
var index = username.lastIndexOf("@");
if (index >= 0) {
    suffix = username.substring(index + 1);
}

// Get the Called-Station-ID attribute from the request.
var csid = filter.Get("Called-Station-ID");
if (csid == null) {
    SbrWriteToLog("RealmExample2: Called-Station-ID attribute is null");
    return SCRIPT_RET_FAILURE;
}

// Concatenate suffix and Calling-Station-ID string to the SQL search key.
accessor.SetInputVariable("Key-String", suffix + csid);

// Execute the data accessor and get realm name if search succeeds.
var realm;

```



```
if (accessor.Execute() == DataAccessor.FOUND) {
    realm = accessor.GetOutputVariable("RealmName");
}
else {
    SbrWriteToLog("RealmExample2: SQL search failed for key string '" + suffix + csid + "'");
    return SCRIPT_RET_FAILURE;
}

// Set the profile to be merged after the authentication request.
var profile = accessor.GetOutputVariable("Profile");
selector.SetAuthProfile(profile);

// Return the realm name for processing the request.
return realm;
[ScriptTrace]
var = suffix
var = csid
var = realm
var = profile
```

Creating Attribute Filter Scripts

IN THIS CHAPTER

- Using Attribute Filter Scripts | 908
- Attribute Filter Script Functions | 909
- Choosing the Return Code | 910
- Configuring Attribute Filter Scripts | 910
- Attribute Filter Script Examples | 914

This chapter describes developing attribute filter scripts to manipulate the values of attributes in the RADIUS request or response packets. This chapter contains these topics:

Using Attribute Filter Scripts

Attribute filter scripts are executed any time a server core component or plug-in module invokes an attribute filter that is configured for scripting.

Steel-Belted Radius Carrier supports both static and scripted attribute filters.

Each is declared by name using the **Filters List** page in Web GUI.

- Static attribute filters—Specify fixed rules using keywords such as **ALLOW**, **ADD**, and **EXCLUDE**. The filter rules and the action of the filter are the same each time Steel-Belted Radius Carrier executes the filter.
- Scripted attribute filters—Declare the name of a JavaScript initialization (**.jsi**) file containing the script code for the filter. When the **Script** setting is present, Steel-Belted Radius Carrier ignores any other filter rules for that filter.

Because a scripted filter is defined with JavaScript, its behavior can change dynamically from one execution to the next. Scripted filters can perform any of these actions:

- Get, add, delete, or replace attributes in the RADIUS request or response, depending on the filter context.
- Manipulate attribute values under program control.

- Execute static attribute filters by name.
- Invoke SQL queries and LDAP searches, and process the results.

Static and scripted attribute filters are referred to by name throughout the rest of the Steel-Belted Radius Carrier configuration. Externally, the two types of filters are equivalent and interchangeable, making it easy to switch between the two. Initially, you can configure Steel-Belted Radius Carrier with static filters and then change to scripted filters after you test the basic configuration.

Attribute Filter Script Functions

Attribute filter scripts can execute any standard JavaScript statements and functions. Additionally, attribute filter scripts use the **AttributeFilter** API to perform filter-specific operations. You must instantiate a new **AttributeFilter** object instance and then use it to invoke these methods, such as:

```
var filter = new AttributeFilter();  
var csid = filter.Get("Calling-Station-ID");
```

These six **AttributeFilter** methods are provided:

- **new AttributeFilter()**—Creates a new **AttributeFilter** object instance.
- **Get()**—Gets the value of an attribute from the request or response packet.
- **Add()**—Adds a new attribute value to the request or response packet.
- **Reset()**—Deletes one or all values of an attribute.
- **Replace()**—Deletes the value of an attribute and replaces it with a new value.
- **Execute()**—Executes a static attribute filter.

NOTE: Do not use the **Execute()** method to invoke a scripted attribute filter; a runtime error occurs.

Attribute filter scripts can also execute any of the methods of the **DataAccessor** class.

For more details about the **AttributeFilter** functions and methods, see [“AttributeFilter Object” on page 964](#).

Choosing the Return Code

When an attribute filter script is finished executing, it returns a result code to the Steel-Belted Radius Carrier core. The code indicates whether or not the script executed successfully. The script may also return a string containing the name of a static attribute filter for Steel-Belted Radius Carrier to execute after the script is finished.

Valid return codes are:

- **SCRIPT_RET_SUCCESS**—The attribute filter script executed successfully. Steel-Belted Radius Carrier should continue processing the current request.
- **SCRIPT_RET_FAILURE**—The attribute filter script failed to execute successfully. Steel-Belted Radius Carrier should terminate request processing and reject the user.
- **<FilterName>**—Steel-Belted Radius Carrier should execute the static filter given by **FilterName** after the script has finished executing and then continue processing the current request.

NOTE: Do not return the name of a scripted attribute filter from your script; a runtime error occurs.

Configuring Attribute Filter Scripts

Use the Web GUI to configure scripts for attribute filters. The attribute filter data are stored in the **filter.ini** file.

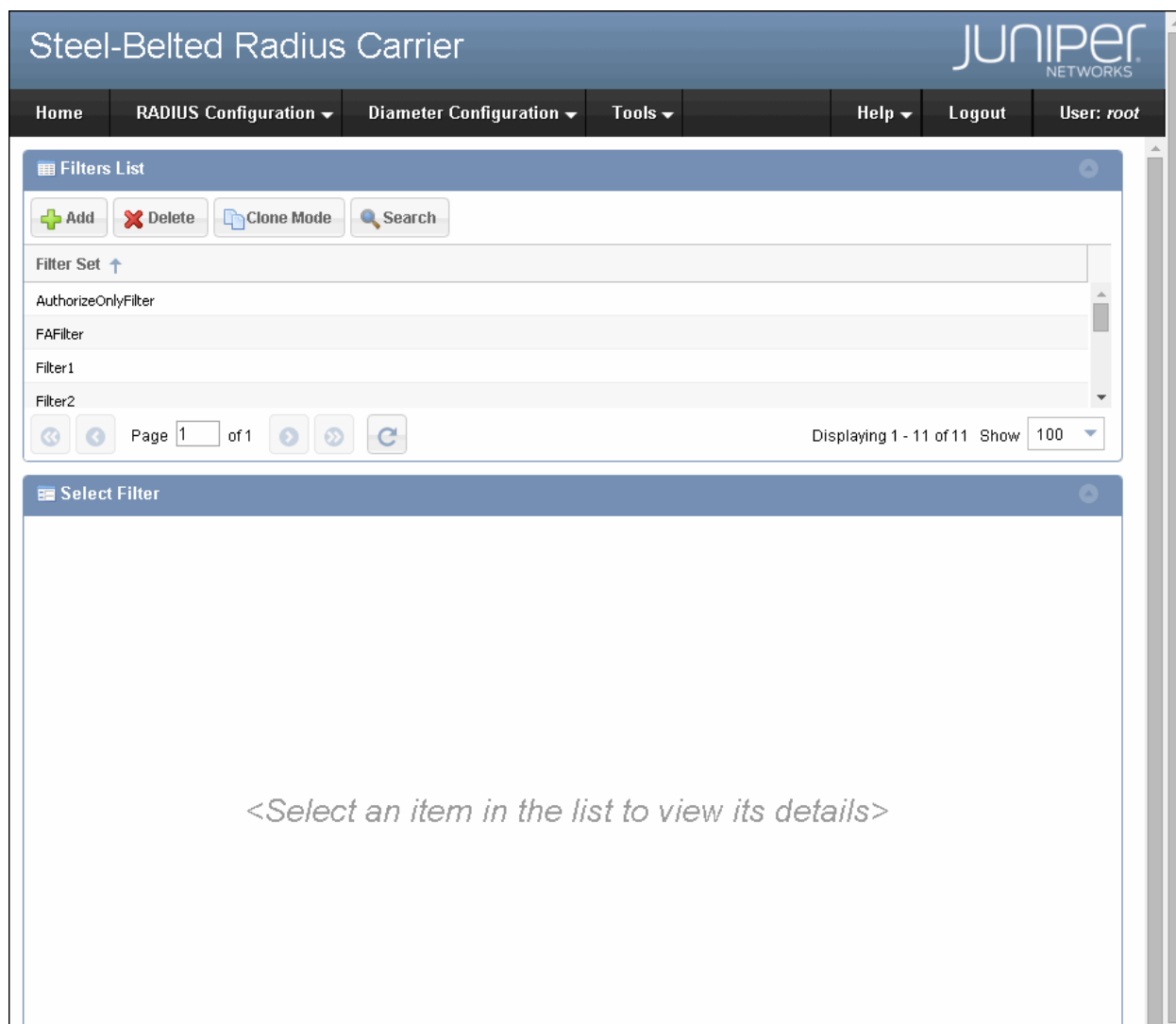
Defining Scripted Filters

To define a scripted filter using the Web GUI:

1. Select **RADIUS Configuration > Filters**.

The **Filters List** page (Figure 282 on page 911) appears.

Figure 282: Filters List Page



2. Click **Add**.

The **Create Filter** pane (Figure 283 on page 912) appears.

Figure 283: Create Filter Pane

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home | RADIUS Configuration ▼ | Diameter Configuration ▼ | Tools ▼ | Help ▼ | Logout | User: root

Filters List

Create Filter

Name:

Description:

Default Rule: ☐ Allow ☒ Exclude

Rules

Rule Type	Attribute	Value	Replacemen...	Replacemen...	Script Name

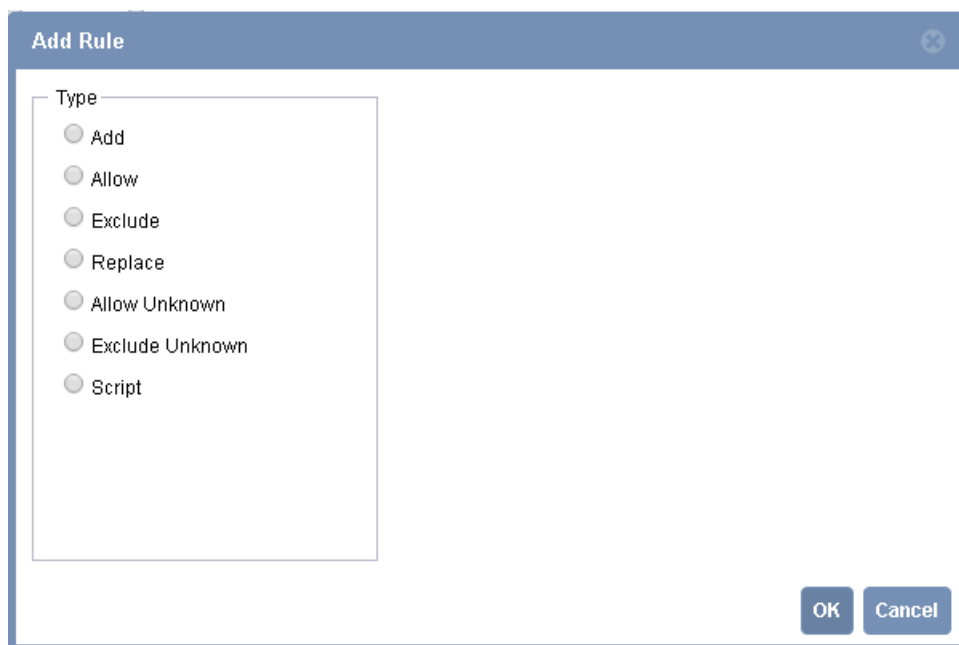
Add Edit Delete

Save Clear Cancel

3. Enter the name and description of the new filter in the **Name** and **Description** fields, respectively. The **Description** field is optional.
4. Click **Add** in the **Rules** area.

The **Add Rule** dialog box ([Figure 284 on page 913](#)) appears.

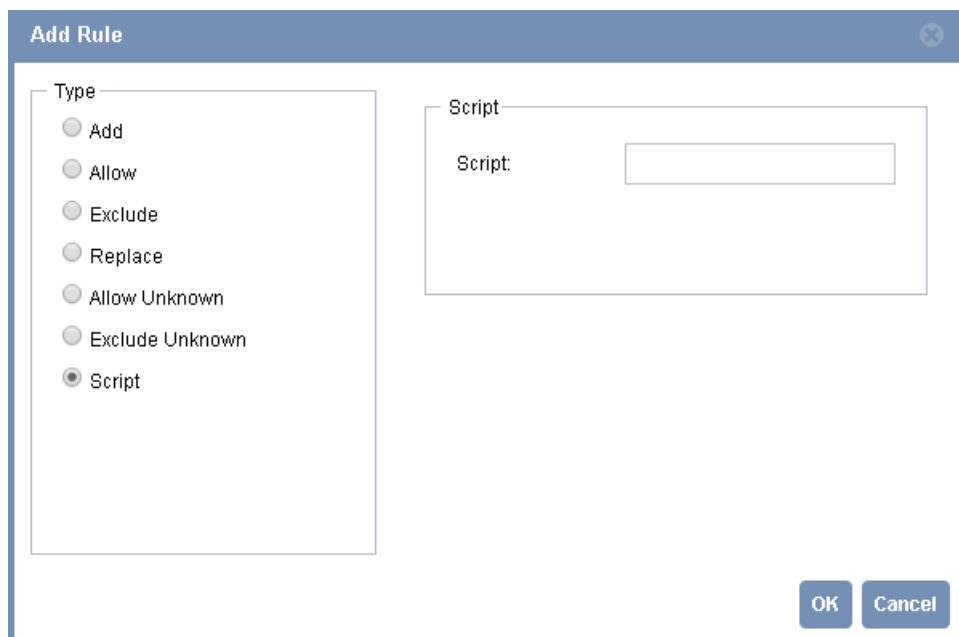
Figure 284: Add Rule Dialog



The "Add Rule" dialog box features a title bar with a close button. On the left, a "Type" section contains a list of radio buttons: Add, Allow, Exclude, Replace, Allow Unknown, Exclude Unknown, and Script. The "Script" option is currently selected. At the bottom right, there are "OK" and "Cancel" buttons.

5. Select the **Script** option button to add a script rule type to the filter. The **Script** pane (Figure 285 on page 913) appears.

Figure 285: Add Rule Dialog—Script



This view of the "Add Rule" dialog shows the "Script" option selected in the "Type" list. A new "Script" pane has appeared on the right, containing a label "Script:" followed by an empty text input field. The "OK" and "Cancel" buttons remain at the bottom right.

6. Enter the name of the script file for the filter in the **Script** field. If you add a script rule type to a filter, all other rule types are ignored.

NOTE: Do not enter the .jsi extension for the script name.

7. Click **OK**.

The **Rules** area in the **Create Filter** pane ([Figure 283 on page 912](#)) displays the updated lists of selected rules.

8. Click **Save** to save the filter configuration.

After you define the attribute filter script, it functions as any other regular attribute filter does. You can define multiple instances of standard and scripted filters. For details about referencing the attribute filter script, see the section on *Attribute Processing Files* in the *SBR Carrier Reference Guide*.

Attribute Filter Script Examples

Example 1: Using an LDAP Query to Select a Static Filter to Execute

This example uses the same LDAP repository as [“Example 4: Conditional Profile Assignment from User Attribute” on page 891](#). In this case, the value of the **Employee-Type** attribute is used to select the name of a statically-defined filter that is returned as the result from the script. If an unknown **Employee-Type** is returned, no static filter is executed, but the script modifies the **Service-Type** attribute programmatically.

Here is the LDAP accessor .gen file:

```
[Bootstrap]
LibraryName=ldapaccessor.dll
Enable=1

[Settings]
MethodName=FilterExample1
Timeout=20
ConnectTimeout=25
QueryTimeout=10
WaitReconnect=2
MaxWaitReconnect=360
UpperCaseName = 0
SSL = 0
Search = FindUser

[Server]
s1=
```



```

[Server/s1]
Host=<server-name>
Port = 389
BindName=uid=admin, ou=Administrators, ou=TopologyManagement, o=NetscapeRoot
BindPassword=<password>

[Request]
User-Name=my-user-name

[Response]
Employee-Type=employeetype

[Attributes/UserAttributes]
employeetype

[Search/FindUser]
Base=ou=people,dc=bigco,dc=com
Scope = 2
Filter = uid=<my-user-name>
Attributes = UserAttributes
Timeout = 20
%DN = dn

[VariableTypes]
User-Name = string
Employee-Type = string

```

Here is the .jsi file for an outbound filter script:

```

[Settings]
LogLevel = 2
ScriptTraceLevel = 0

[Script]
// Initialization block
if (!this.initialized) {
    filter = new AttributeFilter();
    accessor = new DataAccessor("FilterExample1");
    initialized = true;
}
else {
    accessor.Clear();
}

```

```

// Get the employee type from the username and set accessor input variable.
var username = filter.Get("User-Name");
accessor.SetInputVariable("User-Name", username);

// Execute the accessor.
if (accessor.Execute() == DataAccessor.FOUND) {
    // Get the employee type from the query result.
    var type = accessor.GetOutputVariable("Employee-Type");
    if (type == null) {
        SbrWriteToLog("FilterExample1: accessor returned null employee type");
        return SCRIPT_RET_FAILURE;
    }

    // For known employee types, return name of built-in filter to execute.
    switch (type) {
        case "Student":
            return "StudentFilter";
        case "Faculty":
            return "FacultyFilter";
        case "Vendor":
            return "VendorFilter";
        default:
            SbrWriteToLog("FilterExample1: unknown employee type '" + type + "'");
    }
}

// Unknown employee type -- change Service-Type to Authenticate-Only.
filter.Replace("Service-Type", 8);

// Return successfully, but do not execute a built-in filter.
return SCRIPT_RET_SUCCESS;

[ScriptTrace]
var = username
var = type

```

Example 2: Adding Values to Multi-Valued Attributes

This example demonstrates the difference between adding values to orderable and non-orderable multi-valued attributes. The script is configured as an inbound filter. A profile is created to assign three unique **Reply-Message** and three unique **Filter-Id** attribute value strings to the response attribute list. Subsequently, the script adds four new values to each attribute, two of them unique and two of them duplicates. All attribute values are printed to the server log before and after the new values are added.

Because the **Reply-Message** attribute is orderable, the new values are concatenated to the value list and the duplicates are preserved. Because the **Filter-Id** attribute is non-orderable, the new attributes are merged to the list and the duplicates do not appear in the final list.

```
[Settings]
LogLevel = 2
ScriptTraceLevel = 0

[Script]
// Initialization block
if (!this.initialized) {
    filter = new AttributeFilter();
    initialized = true;
}

// Print out all Reply-Message attribute values.
var attr = "";
for(var i = 0; attr != null; i++) {
    attr = filter.Get("Reply-Message", i);
    if (attr != null) {
        SbrWriteToLog("Reply-Message[" + i + "] = " + attr + "");
    }
}

// Add Reply-Message attribute values.
filter.Add("Reply-Message", "Reply-Message 1");
filter.Add("Reply-Message", "Reply-Message 2");
filter.Add("Reply-Message", "Reply-Message 4");
filter.Add("Reply-Message", "Reply-Message 5");

// Print out the new set of Reply-Message attribute values. Since Reply-Message
// is orderable, the values are concatenated, not merged.
attr = "";
for(var i = 0; attr != null; i++) {
    attr = filter.Get("Reply-Message", i);
    if (attr != null) {
        SbrWriteToLog("Reply-Message[" + i + "] = " + attr + "");
    }
}

// Print out all Filter-Id attribute values.
attr = "";
for(var i = 0; attr != null; i++) {
    attr = filter.Get("Filter-Id", i);
    if (attr != null) {
```

```

        SbrWriteToLog("Filter-Id[" + i + "] = " + attr + "");
    }
}

// Add Filter-Id attribute values.
filter.Add("Filter-Id", "Filter-Id 1");
filter.Add("Filter-Id", "Filter-Id 2");
filter.Add("Filter-Id", "Filter-Id 4");
filter.Add("Filter-Id", "Filter-Id 5");

// Print out the new set of Filter-Id attribute values. Since Filter-Id
// is not orderable, the values are merged, not concatenated.
attr = "";
for(var i = 0; attr != null; i++) {
    attr = filter.Get("Filter-Id", i);
    if (attr != null) {
        SbrWriteToLog("Filter-Id[" + i + "] = " + attr + "");
    }
} while (attr != null);

// All done.
return SCRIPT_RET_SUCCESS;

```

Working with Data Accessors

IN THIS CHAPTER

- Data Accessor Overview | 919
- Variable Containers | 920
- Internal Variable Table (LDAP Only) | 921
- Data Accessor Configuration | 921
- Data Conversion Rules | 938
- Data Accessor Configuration File Examples | 944

This chapter explains how to configure Steel-Belted Radius Carrier data accessors and how to use the **DataAccessor** API to connect your realm selection and attribute filter scripts to external SQL databases and LDAP repositories.

You can use data accessors to search for and retrieve credentials and user identity attributes, policy settings, accounting data, and other information from SQL and LDAP data sources. You can insert new information into SQL databases, execute SQL stored procedures, and process the result data values under program control. This chapter contains these topics:

Data Accessor Overview

Data accessors are plug-in modules, similar to the LDAP and SQL authentication plug-ins. Because they do not implement authentication methods, data accessors are called *generic* plug-ins. You configure data accessors with initialization files having the **.gen** filename extension.

When Steel-Belted Radius Carrier starts, it processes all the **.gen** files in its **/radiusdir** directory. Each **.gen** file defines a separate instance of a data accessor plug-in. Settings in the initialization file specify the name of the shared object file (**.so** or **.dll**) to load and a name string to identify the plug-in instance. Many **.gen** files may refer to a single shared object file, but the name string must be unique for each plug-in instance.

The **.gen** file specifies the connection settings and query strings for the SQL database or LDAP repository. These settings are similar or identical to those in the corresponding SQL and LDAP **.aut** files. The **.gen** file also specifies type declarations used when passing data between scripts and the plug-ins.

Scripts use the **DataAccessor** API to communicate with data accessors. The **DataAccessor** API exposes five methods:

- **new DataAccessor()**—Creates a new **DataAccessor** object bound to a single data accessor plug-in instance.
- **SetInputVariable()**—Sets a request data value in the data accessor's input variable container.
- **GetOutputVariable()**—Gets a response data value from the data accessor's output variable container.
- **Clear()**—Deletes all data values in the data accessor's input and output variable containers.
- **Execute()**—Performs a single operation on the external data source.

For more information about the **DataAccessor** API, see [“DataAccessor Object” on page 969](#).

The **.gen** file hides the details of the external data connection from your scripts. The **DataAccessor** API appears the same way to your scripts whether you are connecting to a SQL database or an LDAP repository.

NOTE: LDAP scripting does not support **DataAccessors()**. Trying to execute a **DataAccessor()** from an LDAP script causes a runtime exception.

Variable Containers

You use variable containers to pass data values back and forth between scripts and data accessors. Each data accessor instance has two variable containers: an input variable container and an output variable container. The input container receives data values from the script, and the output container returns data values to the script.

Variable containers are data arrays. Rows in the array have three columns:

- A unique string identifying one variable in the container.
- The variable data type (**string**, **binary**, **integer**, **ipaddress**, **ipv6address**, **ipv6prefix**, **ipv6interface**, or **date**).
- The actual data value.

Settings in the **.gen** file specify the number of rows in each variable container, the variable names, and their data types. These settings are fixed when Steel-Belted Radius Carrier starts up and cannot be changed by your scripts. Only the data values change when you call the **DataAccessor** script API.

Call the **SetInputVariable()** and **GetOutputVariable()** API calls to transfer data values into and out of the variable containers. Each function call transfers a single data value. The name argument selects the container row to read or modify. The name must match one of the variable declarations in the **.gen** file or a runtime error occurs.

To clear all values in both the input and output variable containers, call the data accessor object's **Clear()** method.

When you call the **Execute()** method, values from the input container are bound into the SQL query or LDAP search specified in the **.gen** file. When the operation completes, selected values are copied from the result to the output container where they may be read by your script.

Internal Variable Table (LDAP Only)

Like the LDAP authentication plug-in, the LDAP data accessor has an internal variable table used for temporary data storage during the processing of LDAP search requests. The data accessor variable table works in almost the same way as the authentication plug-in variable table, except that it maps to entries in the input and output variable containers instead of attributes in the RADIUS request and response packets. The data flow in and out of the internal variable table is:

1. At the beginning of each **Execute()** call to an LDAP data accessor, Steel-Belted Radius Carrier uses settings in the [Request] section of the **.gen** file to select which entries in the input variable container are copied into the internal variable table.
2. Steel-Belted Radius Carrier performs one or more LDAP searches. After each search is performed, selected attributes are copied by name from the LDAP result and placed in the internal variable table. Settings in the [Attributes/*name*] sections of the **.gen** file determine which LDAP result attributes are copied.
3. Steel-Belted Radius Carrier uses the [Response] section to select information from the variable table to be copied to the LDAP data accessor output variable container.

For more information about the variable table, see [“Working with the Variable Table” on page 884](#), and for more information about the LDAP request life cycle, see [“LDAP Request Life Cycle” on page 882](#).

Data Accessor Configuration

SQL Data Accessor Configuration

[Bootstrap] Section

The [Bootstrap] section ([Table 131 on page 922](#)) of the SQL data accessor configuration file specifies information that Steel-Belted Radius Carrier uses to load and start the SQL data accessor plug-in.

You can configure more than one SQL data accessor plug-in instance. Each requires its own **.gen** file in the **/radiusdir** directory. The [Bootstrap] section of each **.gen** file must include one of these libraries as the LibraryName entry:

- radsql_accessor_ora.so
- or
- radsql_accessor_jdbc.so

```
[Bootstrap]
LibraryName=radsql_accessor_ora.so
Enable=1
```

Table 131: [Bootstrap] Syntax

Parameter	Function
LibraryName	LibraryName specifies the name of one SQL data accessor plug-in. It may be: <ul style="list-style-type: none">• radsql_accessor_ora.so• radsql_accessor_jdbc.so
Enable	Specifies whether the SQL data accessor instance is enabled. <ul style="list-style-type: none">• If set to 0, the SQL data accessor instance is disabled.• If set to 1, the SQL data accessor instance is enabled. Default value is 0.

[Results] Section

The purpose of the [Results] section is to declare data accessor output container variables and map them to columns in the SQL query result set.

Consider this **SELECT** statement:

```
SELECT user_pwd, attribs, fullname FROM rasusers WHERE user_id =@User-Name
```

Where **user_pwd**, **attribs**, **fullname**, and **user_id** are the names of columns in the SQL table, and **rasusers** is the name of the SQL table itself. The [Results] section maps the output variables to the columns retrieved from the SQL database; for example:

```
[Results]
Password=1
Profile=2
FullName=3
```


Columns in the SQL query are identified in the [Results] section by number; **1** represents the first column in the **SELECT** query (from left to right), and if other columns are also references, **2** represents the second, and **3** the third, and so on.

[Settings] Section

The [Settings] section ([Table 132 on page 923](#)) of the SQL data accessor configuration file defines parameters that control the database connection.

```
[Settings]
Connect=DSN=<dsn_name_here>;UID=<username_for_dB>;PWD=<password_for_dB>
SQL=SELECT password, profile FROM userlist WHERE name = @User-Name
ParameterMarker=?
MaxConcurrent=1
ConcurrentTimeout=30
ConnectTimeout=25
QueryTimeout=25
WaitReconnect=2
MaxWaitReconnect=360
```

Table 132: *.gen [Settings] Syntax

Parameter	Function
ConcurrentTimeout	Specifies the number of seconds a request may wait for execution before it is discarded. Because there may be multiple MaxConcurrent SQL statements executing at one time, new requests must be queued as they arrive until other statements are processed.
Connect	<p>Specifies the string that must be passed to the database client engine to establish a connection to the database. This string has (or refers to) information about the name of the database, its location on the network, the password required to access it, and so forth.</p> <p>The format of the connect string depends on the type of database you use:</p> <p>Oracle:</p> <p>Connect=<dB_username>/<dB_password></p> <p>JDBC:</p> <p>Connect=DSN=<dsn_name_here> ;UID=<username_for_dB>;PWD= <password_for_dB</p>

Table 132: *.gen [Settings] Syntax (continued)

Parameter	Function
ConnectDelimiter	<p>(JDBC only) Specifies the character used to separate fields (DSN, UID, PWD) in the connect string.</p> <p>Default value is a ; (semicolon). If the JDBC connect string requires use of semicolons as part of a field value, you can use this parameter to specify a different delimiter, such as : (colon).</p>
ConnectTimeout	<p>Specifies the number of seconds to wait when attempting to establish the connection to the database before timing out.</p> <p>This value is ignored if the client database engine does not support this feature.</p>
Driver	<p>(JDBC only) Specifies the third-party JDBC driver to load. For example:</p> <p>Driver=com/provider/jdbc/sqlserver/SQLServerDriver</p> <p>NOTE: Third-party JDBC drivers must be installed in the <JRE-path>/lib/ext directory. Where, <JRE-path> indicates the path where the JRE (that is integrated with SBR Carrier) is installed in your system. Refer to the JDBC driver documentation for information about how to install the JDBC driver and supporting files.</p>
LogLevel	<p>Activates logging for the Data Accessor and sets the rate at which it writes entries to the server log file (.LOG). The LogLevel may be the number 0, 1, or 2, where 0 is the lowest logging level, 1 is intermediate, and 2 is the most verbose. If the LogLevel that you set in the *.gen file is different than the LogLevel in radius.ini, the radius.ini setting determines the rate of logging.</p>
MaxConcurrent	<p>Specifies the maximum number of instances of a single SQL statement that may be executing at one time.</p>
MaxWaitReconnect	<p>Specifies the maximum number of seconds to wait after successive failures to reconnect after a failure of the database connection.</p> <p>WaitReconnect specifies the time to wait after failure of the database connection. This value is doubled on each failed attempt to reconnect, up to a maximum of MaxWaitReconnect.</p>

Table 132: *.gen [Settings] Syntax (continued)

Parameter	Function
ParameterMarker	Specifies the character or sequence of characters used as the parameter marker in a parameterized SQL query. Normally, this is the question mark (?), but this can vary among database vendors.
QueryTimeout	Specifies the number of seconds to wait for a response to a query before timing out. This value is passed to the client database engine, which may or may not implement the feature.
SQL	<p>Specifies the SQL statement used to access and insert information in the database and indicates the names of the variables to create in the input variable container when a SQL statement variable is preceded by a @ sign. The SQL statement may be broken over several lines by ending each line with a backslash. The backslash must be preceded by a space character, and followed by a newline character. The subsequent lines may be indented for better readability.</p> <p>Example:</p> <pre>SQL=SELECT password, profile, fullname \FROM usertable \WHERE username = @User-Name</pre>
WaitReconnect	Specifies the number of seconds to wait after a failure of the database connection before trying to connect again.

NOTE: Declare input variables within the SQL query string (**SQL** parameter) by putting a @ sign before the variable name. This causes Steel-Belted Radius Carrier to create an entry for the variable in the data accessor's input variable container. This is different from the LDAP data accessor, which uses a separate [Request] section to declare input container variables.

For an example of a SQL accessor configuration, see ["Example: SQL Data Accessor Configuration File" on page 945](#).

[VariableTypes] Section

The [VariableTypes] section of the SQL data accessor configuration file specifies the storage data type for each entry in the input and output variable containers.

```
[VariableTypes]
< variable >=< type >
< variable >=< type >
.
.
.
```

Where **variable** is the name of an input or output container variable and **type** is the data type specifier, one of **string**, **binary**, **integer**, **ipaddress**, **ipv6address**, **ipv6prefix**, **ipv6interface**, or **date**.

For more information about selecting the variable type specifier, see [“Data Conversion Rules” on page 938](#).

LDAP Data Accessor Configuration

[Bootstrap] Section

The [Bootstrap] section of the LDAP data accessor configuration file specifies information that Steel-Belted Radius Carrier uses to load and start the LDAP data accessor plug-in.

You can configure more than one LDAP data accessor plug-in instance. Each requires its own **.gen** file in the **/radiusdir** directory. The [Bootstrap] section ([Table 133 on page 926](#)) of each **.gen** file must provide a **LibraryName** of **ldapaccessor.so**.

Table 133: *.gen [Bootstrap] Syntax

Parameter	Function
LibraryName	Specifies the name of the LDAP data accessor plug-in. ldapaccessor.so
Enable	Specifies whether the LDAP data accessor instance is enabled. <ul style="list-style-type: none">• If set to 0, the LDAP data accessor instance is disabled.• If set to 1, the LDAP data accessor instance is enabled. Default value is 0.

[Attributes/name] Sections

An LDAP search returns all of the attributes associated with an LDAP entry. Many of these attributes may not be relevant to your script. When specifying an LDAP Search for the data accessor, you can provide a list of specific LDAP result attributes to retain by name in the internal variable table. The other attributes are discarded.

You configure [Attributes/name] sections in the LDAP data accessor **.gen** file to create named lists of LDAP attributes. The syntax is as follows:

```
[Attributes/name]
```

```
attribute
```

```
attribute
```

```
.
```

```
.
```

```
.
```

where ***attribute*** is the name of an LDAP attribute and ***name*** is an arbitrary name for the section. You must type the attribute names exactly as they appear in your LDAP database schema. Use one line per attribute. For example:

```
[Attributes/InterestingAttributes]
```

```
User-Secret
```

```
RADIUS-Profile
```

```
Inactivity-Timeout
```

```
.
```

```
.
```

```
.
```

An [Attributes/*name*] section is associated with a [Search/*name*] section using the **Attributes** parameter. For example:

```
[Search/DoLdapSearch]
```

```
Attributes = InterestingAttributes
```

When the search executes, the selected result attributes are stored by name in the LDAP data accessor internal variable table. If the **Attributes** parameter is omitted from a [Search/*name*] section, all of the attributes returned by the LDAP search are stored. Of these attributes, only those referred to in the [Response] section of the **.gen** file are copied into the output variable container; the rest are discarded after all LDAP searches have completed.

[Response] Section

The [Response] section provides the LDAP data accessor with the names of variables to create in the output variable container. It also indicates to the data accessor which values from the internal variable table to copy to entries in the output variable container after all LDAP repository searches have completed.

The [Response] section syntax is as follows:

```
[Response]
```

```
outvar = tablevar
```

```
outvar = tablevar
```

```
.  
.br/>.
```

Where **outvar** is the name of a variable in the output variable container and **tablevar** is the name of an entry in the internal variable table.

[Search/name] Sections

Each [Search/name] section (Table 134 on page 928) in the LDAP data accessor configuration file specifies the complete details of one LDAP Search request. You can use the same search request on various LDAP repositories because the details of the LDAP server connection are specified separately.

By default, when you execute the search request specified in a [Search/name] section, all the attributes associated with the resulting LDAP record are retained. Use the **Attributes** parameter to specify a list of specific attributes you want to store in the internal variable table.

```
[Search/DoLdapSearch]  
Base = o=bigco.com  
Scope = 2  
Filter = item1=<value1>  
Attributes = LdapAttrs  
Timeout = 20  
%DN = dn  
  
[Attributes/LdapAttrs]  
item2  
  
[Response]  
Output-Var = item2
```

Table 134: *.gen [Search/name] Syntax

Parameter	Function
%DN	Specifies an entry in the internal variable in which to place the distinguished name that results from the Search should be placed.
Attributes	Specifies the LDAP attributes relevant to Steel-Belted Radius Carrier, by referencing an [Attributes/name] section elsewhere in the same .gen file.

Table 134: *.gen [Search/name] Syntax (continued)

Parameter	Function
Base	<p>Specifies the distinguished name (DN) of the entry that serves as the starting point for the search. This filter is a template for an LDAP distinguished name string. The filter follows conventional LDAP syntax and may be as simple or as complex as LDAP syntax permits. It may also include replacement variables from the Variable Table.</p> <p>Each replacement variable consists of the variable name enclosed in angle brackets (<>). Upon execution of the LDAP Search request, the value of the variable replaces the variable name.</p>
OnFound	<p>Specifies the next request section when data is found. The value of this parameter is a string, <i>name</i>. The <i>name</i> specifies an LDAP Search request by referencing a [Search/<i>name</i>] section elsewhere in the same *.gen file. If there is no next request section, the overall operation succeeds.</p>
OnNotFound	<p>Specifies the next request section when data is not found. The value of this parameter is a string, <i>name</i>. The <i>name</i> specifies an LDAP Search request by referencing a [Search/<i>name</i>] section elsewhere in the same *.gen file. If there is no next request section, the overall operation fails.</p>
Filter	<p>Specifies the filter to apply to the search. This filter is a template for an LDAP Search string. The filter follows conventional LDAP syntax and may be as simple or as complex as LDAP syntax permits, with multiple attribute/value assertions in Boolean combination. It may also include replacement variables from the Variable Table.</p> <p>Each replacement variable consists of the variable name enclosed in angle brackets (<>). Upon execution of the LDAP Search request, the value of the variable replaces the variable name.</p> <p>For example, a Search template that uses the User-Name and Service-Type attributes from the RADIUS request might look like this:</p> <p>(&(uid = <User-Name>)(type = <Service-Type>))</p>
Scope	<p>Specifies the scope of the search; 0 (search the base), 1 (search all entries one level beneath the base), or 2 (search the base and all entries beneath the base at any level).</p>

The **OnFound** and **OnNotFound** parameters can be used in [Search/*name*] sections to create serial chains of search requests. The **OnFound** parameter specifies the name of a search to try if the current search returns a non-empty result from the LDAP repository. The **OnNotFound** parameter specifies the name of a search to try if the current search fails to return data. Arbitrarily complex search trees may be created.

This example shows a simple LDAP search tree:

```
[Search/DoSearch2]
Base = o=xyz.com
Scope = 2
Filter = uid=<User-Name>
Attributes = AttrList
Timeout = 20
%DN = dn
OnFound = DoSearch8
OnNotFound = DoSearch9

[Search/DoSearch8]
Base = o=xyz.com
Scope = 2
Filter = uid=<User-Name>
Attributes = AttrList
Timeout = 20
%DN = dn
OnFound = DoSearch9
OnNotFound = DoSearch9

[Search/DoSearch9]
Base = o=xyz.com
Scope = 2
Filter = uid=<User-Name>
Attributes = AttrList
Timeout = 20
%DN = dn
```

[Request] Section

The [Request] section provides the LDAP data accessor with the names of variables to create in the input variable container. It also indicates to the data accessor which input variable container entries to copy to the internal variable table before execution of the LDAP search request tree.

```
[Request]
invar = tablevar
invar = tablevar
.
```



```

.
.

```

Where ***invar*** is the name of a variable in the input variable container and ***tablevar*** is the name of an entry in the internal variable table.

tablevar may be omitted from any [Request] entry. If so, the variable in the input variable container is copied to an internal variable table entry named ***invar***.

```

[Request]
invar =

```

[Defaults] Section

The [Defaults] section of the LDAP data accessor configuration file enables you to add named entries to the internal variable table before the LDAP search tree is executed. You can reference these variables in your queries, even if they are not initialized from the [Request] section.

The format of each [Defaults] entry is:

```

tablevar = value

```

Where ***tablevar*** is the name of a variable in the internal variable table and ***value*** is the value you want to assign to it. For example:

```

[Defaults]
Filter = campus_only
SessionLimit = 600

```

[Server/name] Sections

Several sections of the LDAP data accessor file work together to configure the connection between the Steel-Belted Radius Carrier server and the LDAP database server(s). The sections are: [Server], [Server/name], and [Settings].

Each [Server/name] section of the LDAP data accessor file contains configuration information about a single LDAP server. You must provide a [Server/name] section for each server you named in the [Server] section ([Table 135 on page 932](#)). For example:

```

[Server]
s1=
s2=
[Server/s1]
Host = ldap_1

```

```

Port = 389
.
.
.
[Server/s2]
Host = 130.4.67.1
LastResort = 1
.
.
.

```

Table 135: *.gen [Server/name] Syntax

Parameter	Function
BindName	The BindName parameter specifies the distinguished name (DN) to be used to connect to the LDAP server. In [Server/name] section, specify a unique BindName for a specific server. Use the [Settings] section to specify a default BindName to use for all servers.
BindPassword	The BindPassword specifies the password to be used to connect to the LDAP server. In [Server/name] section, specify a unique BindPassword for a specific server. Use the [Settings] section to specify a default BindPassword to use for all servers.
Certificates	Specifies the path of the certificate database for use with SSL. This path must not end in a filename.
ConnectTimeout	Specifies the number of seconds to wait when attempting to establish the connection to the database before timing out. This value is passed to the client database engine, which may or may not implement the feature.
FlashReconnect	<p>If the server is down when performing a Search, setting this parameter to 1 triggers a reconnection attempt before rejecting the request. Therefore, requests are not rejected due to inactivity timeouts.</p> <p>This setting applies to a particular server. To apply it for all servers, place it in the [Settings] section.</p>
Host	<p>The hostname or IP address of the LDAP server.</p> <p>NOTE: For SSL configurations, the hostname accepts only LDAP-style URIs. For example, ldaps://hostname:port.</p>

Table 135: *.gen [Server/name] Syntax (continued)

Parameter	Function
LastResort	You may identify a <i>last resort</i> LDAP server by providing a LastResort parameter in one of these [Server/name] sections, and setting its value to 1. If the search returns no result, the execution of the search tree completes (unless OnNotFound is configured).
LdapVersion	Specifies the version of LDAP protocol, if needed to override the default given in the [Settings] section.
MaxConcurrent	Specifies the maximum number of instances of a single LDAP request that may be executing at one time.
MaxWaitReconnect	Specifies the maximum number of seconds to wait after successive failures to reconnect after a failure of the database connection. WaitReconnect specifies the time to wait after failure of the database connection. This value is doubled on each failed attempt to reconnect, up to a maximum of MaxWaitReconnect.
Port	<p>The TCP port of the LDAP server, or 0 to use the standard port.</p> <p>Default value is 0.</p> <p>NOTE: For SSL configurations, the default port setting is ignored and the LDAP-style URIs for Host is applied. For example, ldaps://hostname:port.</p>
QueryTimeout	Specifies the number of seconds to wait for the execution of an LDAP request to complete before timing out. This value is passed to the database engine, which may or may not implement the feature.
Search	The value of this parameter is a string, name . The name specifies an LDAP Search request by referencing a [Search/name] section elsewhere in the same .gen file.
SSL	<ul style="list-style-type: none"> • If set to 0, SSL is not used over the LDAP connection. • If set to 1, SSL is used over the LDAP connection. <p>Default value is 0.</p>
WaitReconnect	Specifies the number of seconds to wait after a failure of the database connection before trying to connect again.

[Server] Section

The [Server] section ([Table 136 on page 935](#)) of the LDAP data accessor configuration file lists the LDAP servers. You can specify more than one server in the [Server] section for load-balancing or backup. When more than one server is specified, Steel-Belted Radius Carrier authenticates against these databases in a round-robin fashion.

The syntax is as follows:

```
[Server]
ServerName=TargetNumber
ServerName=TargetNumber
.
.
.
```

Where **ServerName** is the name of a header file section that contains configuration information for that server, and **TargetNumber** is an *activation target number*, a number that controls when this server is activated for backup purposes. TargetNumber is optional and may be left blank. For example:

```
[Server]
s1 =
s2 =
[Server/s1]
.
. ;Connection details for server s1
.
[Server/s2]
.
. ;Connection details for server s2
.
.
.
```

A Steel-Belted Radius Carrier server maintains connectivity with its LDAP servers according to these rules:

- The priority of the server by order. The first entry in the [Server] section has the highest priority.
- By activation target number. The rule for the activation target is that if the number of LDAP servers that Steel-Belted Radius Carrier is connected to is less than the activation target, Steel-Belted Radius Carrier connects to the server and includes it in the round-robin list. While the number of active servers is equal to or greater than the activation target, Steel-Belted Radius Carrier does not use that server in the round-robin list. An activation target of 0 indicates that, in the current configuration, this machine is never used.

[Settings] Section

The [Settings] section ([Table 136 on page 935](#)) of the LDAP data accessor configuration file forms a basis for all Search requests to the LDAP database servers.

The values set in [Settings] for some parameters, such as **ConnectTimeout**, **MaxConcurrent**, or **WaitReconnect**, provide defaults that apply to all servers. These default values can be overridden for a particular server by entering the same parameter with a different value in a [Server/*name*] section.

Table 136: *.gen [Settings] Syntax

Parameter	Function
BindName	In the [Settings] section, BindName and BindPassword specify a default LDAP template to use for all servers. You can also use BindName and BindPassword in [Server/ <i>name</i>] sections to override this default for an individual server.
ConnectTimeout	Specifies the number of seconds to wait when attempting to establish the connection to the database before timing out. This value is passed to the client database engine, which may or may not implement the feature. Default value is 25 seconds.
FilterSpecial CharacterHandling	<ul style="list-style-type: none"> • If set to 1, specifies that non-alphanumeric characters, such as (or), should be converted to an ASCII hex value preceded by a backslash when they are encountered in a username. • If set to 0, non-alphanumeric characters are not converted. Default value is 0.
FlashReconnect	If the server is down when performing a Search, setting this parameter to 1 triggers a reconnection attempt before rejecting the request. Therefore, requests are not rejected due to inactivity timeouts. NOTE: The value specified in this parameter can be overridden in individual [Server/ <i>name</i>] sections of this file.
LdapVersion	Specifies the version of LDAP protocol. Default value is 2.

Table 136: *.gen [Settings] Syntax (continued)

Parameter	Function
LogLevel	<p>Activates logging for the LDAP data accessor and sets the rate at which it writes entries to the Steel-Belted Radius Carrier server log file (.LOG). This value may be the number 0, 1, or 2, where 0 is the lowest logging level, 1 is intermediate, and 2 is the most verbose.</p> <p>If the LogLevel that you set in the .gen file is different than the LogLevel in radius.ini, the radius.ini setting determines the rate of logging.</p> <p>The LogLevel is re-read whenever the server receives a SIGHUP (1) signal.</p>
MaxConcurrent	<p>Specifies the maximum number of instances of a single LDAP request that may be executing at one time.</p> <p>NOTE: The value specified in this parameter can be overridden in individual [Server/<i>name</i>] sections of this file.</p>
MaxWaitReconnect	<p>Specifies the maximum number of seconds to wait after successive failures to reconnect after a failure of the database connection. WaitReconnect specifies the time to wait after failure of the database connection. This value is doubled on each failed attempt to reconnect, up to a maximum of MaxWaitReconnect.</p> <p>NOTE: The value specified in this parameter can be overridden in individual [Server/<i>name</i>] sections of this file.</p>
OnFound	<p>Specifies the next request section when data is found. The value of this parameter is a string, name. The name specifies an LDAP Search request by referencing a [Search/<i>name</i>] section elsewhere in the same .gen file. If there is no next request section, the overall operation succeeds.</p>
OnNotFound	<p>Specifies the next request section when data is not found. The value of this parameter is a string, name. The name specifies an LDAP Search request by referencing a [Search/<i>name</i>] section elsewhere in the same .gen file. If there is no next request section, the overall operation fails.</p>
QueryTimeout	<p>Specifies the timeout value in seconds for an individual search performed against the LDAP server.</p> <p>Default value is 10 seconds.</p>

Table 136: *.gen [Settings] Syntax (continued)

Parameter	Function
Search	The value of this parameter is a string, name . The name specifies an LDAP Search request by referencing a [Search/ <i>name</i>] section elsewhere in the same .gen file.
SSL	<ul style="list-style-type: none"> • If set to 0, SSL is not used over the LDAP connection. • If set to 1, SSL is used over the LDAP connection. <p>Default value is 0.</p> <p>NOTE: The value specified in this parameter can be overridden in individual [Server/<i>name</i>] sections of this file.</p> <p>If SSL=1, the Host parameter in [Server/name] accepts only LDAP-style URIs. For example, ldaps://hostname:port.</p>
WaitReconnect	Specifies the number of seconds to wait after a failure of the database connection before trying to connect again.
Timeout	<p>Specifies the maximum number of seconds for the overall timeout for each request, which includes the delay in acquiring resources, attempts against multiple LDAP servers, and so forth.</p> <p>Default value is 20 seconds.</p>
UTC	<ul style="list-style-type: none"> • If set to 0, time values are displayed using the local time. • If set to 1, time values are displayed using universal time coordinates (UTC).
WaitReconnect	<p>Specifies the number of seconds to wait after a failure of the database connection before trying to connect again.</p> <p>NOTE: The value specified in this parameter can be overridden in individual [Server/<i>name</i>] sections of this file.</p>

[VariableTypes] Section

The [VariableTypes] section of the LDAP data accessor configuration file specifies the storage data type for each entry in the input and output variable containers.

```
[VariableTypes]
< variable >=< type >
< variable >=< type >
.
```

```

.
.

```

where **variable** is the name of an input or output container variable and **type** is the data type specifier, one of **string**, **binary**, **integer**, **ipaddress**, **ipv6address**, **ipv6prefix**, **ipv6interface**, or **date**.

For more information about selecting the variable type specifier, see [“Data Conversion Rules” on page 938](#).

Data Conversion Rules

This section describes the data conversions that occur when you use the SQL and LDAP data accessors. It provides the rules you need to specify variable types in your scripts, external databases, and data conversion files.

A single data variable can be represented differently in a script, a data accessor, and an external database. You must configure all three components correctly so that Steel-Belted Radius Carrier can perform the correct data conversions. Improper configuration can cause data corruption or runtime errors.

The SQL or LDAP schema determines how data are stored in external databases. Use the [VariableTypes] section of the LDAP or SQL data accessor **.gen** file to specify the corresponding data types for input and output container variables. You must also use the appropriate JavaScript syntax in your scripts.

NOTE: The maximum integer value supported during data conversion is 2,147,483,647 (0x7FFFFFFF). To retrieve an integer value greater than the maximum value from the database, the output container variable should be configured as **string** in the [VariableTypes] section of the **.gen** file and the keyword **Number** should be used in your script to convert the string to integer. For an example configuration, see [“Example 4” on page 941](#).

Output Container

SQL and LDAP data accessor plug-ins require that output container variables map to database columns of type **string**. When the **DataAccessor.Execute()** function is called, the result strings returned from the database are parsed according to the respective output container variable types declared in the data accessor configuration (**.gen**) file. The parsed data values are stored in the output container.

Subsequently, when the **DataAccessor.GetOutputVariable()** function is called, a second conversion is performed. Output container variables of type **integer** are returned to the script as JavaScript integer data. All other output container variable types are returned as strings. To avoid unnecessary data conversions,

output container variables that refer to integer data should be declared as **integer** in the [VariableTypes] section of the **.gen** file. All other output container variables should be declared as **string**.

NOTE: It is legal, but inefficient, to declare an output container variable to be a type other than **integer** or **string**. For example, if an output container variable is declared to be type **ipaddress**, when the data accessor executes, the string representation of the address in the database must first be converted to a binary IP address structure, then when the script retrieves the value from the output container, the binary IP address must be converted back to a string representation. This conversion process wastes a significant amount of computing resources.

Attempting to map an output container variable to a non-string database column causes a runtime error when the data accessor is executed regardless of the variable type declaration in the **.gen** file.

Input Container

Data conversions are not performed when data are passed from the script to the input container, and from the input container to the database. Input container variables might refer to string, numeric, or binary column data in the database. The data types must all match across the script, the **.gen** file, and the database schema.

In most cases, the only input container variable types used with the JavaScript API (and the corresponding database column types) are **integer** and **string**. Other data types can be stored in the input container by coding the binary data directly as unformatted JavaScript strings using the standard **String.fromCharCode()** function. You are responsible for programming the binary coding correctly.

Examples

The examples in this section refer to SQL database queries, tables, and columns, but can apply to LDAP searches, records, and tables.

Example 1

When **DataAccessor.SetInputVariable()** is called for an **integer** variable, the JavaScript API can determine how to convert the supplied argument. For example, the following **.gen** file configuration declares an input container variable **Count** of type **integer**:

```
Select FooVar FROM bar WHERE index = @Count
[VariableTypes]
FooVar=string
Count=integer

[Results]
```

```
FooVar=1/64
```

Both of these statements result in the correct integer data type being passed to the **Count** variable in the SQL query:

```
DataAccessor.SetInputVariable("Count", 1234);
DataAccessor.SetInputVariable("Count", "1234" );
```

In this example, the **index** column in the database must also be of integer type to match the **Count** input container variable. If not, a type mismatch error occurs when the plug-in attempts to execute the SQL statement.

Example 2

Represent types other than integer as strings. In this configuration file, the **ipaddress** column in database table **bar** contains IP address values represented as formatted strings. The input container variable **IP-Address** is declared as a string in the **.gen** file; and in the call to **DataAccessor.SetInputVariable()**, the argument value is also passed as a string.

The **.gen** file configuration is as follows:

```
Select FooVar FROM bar WHERE address = @ip-address

[VariableTypes]
FooVar=string
IP-Address=string

[Results]
FooVar=1/64
```

The **.jsi** file configuration is as follows:

```
DataAccessor.SetInputVariable("IP-Address", "128.18.40.1" );
```

You can change the **.gen** file to make **IP-Address** a binary (**ipaddress**) input container variable.

```
[VariableTypes]
FooVar=string
IP-Address=ipaddress
```

Then the script might have to pass the data to **SetInputVariable()** as an integer or unformatted binary string.

```
var intData = 47276480;
DataAccessor.SetInputVariable("IP-Address", intData);
```

or:

```
var binData = String.fromCharCode(0x80, 0x12, 0x28, 0x01);
DataAccessor.SetInputVariable("IP-Address", binData);
```

Because the container variable **ip-address** is declared as type **ipaddress**, the data accessor expects the input container data to be in a binary format appropriate to the type and passes it along unchanged to the database. The definition of the database **address** column must be changed to an appropriate binary type.

Example 3

This example shows how to retrieve an IP address value from the database. Because column **IP-Address** is mapped to an output container variable, the database schema must declare **IP-Address** to be of string type.

The **.gen** file configuration is as follows:

```
Select IP-Address FROM foobar WHERE username = @User-Name

[VariableTypes]
IP-Address=string
User-Name=string

[Results]
IP-Address=1/64
```

The **.jsi** file configuration is as follows:

```
DataAccessor.SetInputVariable("User-Name", "testuser");
DataAccessor.Execute();
var addrStr = DataAccessor.GetOutputVariable("IP-Address");
```

Similar results can be generated by declaring output container variable **IP-Address** to be of type **ipaddress**. As noted, this causes the data value to be converted to a binary form internal to the data accessor, but wastes some computing resources.

Example 4

This example explains how to retrieve an integer value that is greater than the maximum value 2,147,483,647 (0x7FFFFFFF). In this example, the output container variable is mapped to the database column **User-ID** of type integer. The variable **User-ID** is declared as string in the **.gen** file, because the **User-ID** value stored

in the database is greater than the maximum integer value. In the script, the **Number** keyword is used to convert the string back to integer.

NOTE: The following syntax also works in cases where the size of the integer may be unknown.

The **.gen** file configuration is as follows:

```
Select User-ID FROM foobar WHERE username = @User-Name

[VariableTypes]
User-ID=string
User-Name=string

[Results]
User-ID=1/64
```

The **.jsi** file configuration is as follows:

```
DataAccessor.Execute();
var intData=DataAccessor.GetOutputVariable("User-ID" );
IntegerOut=Number(intData);
```

Supported Data Types and Conversions

[Table 137 on page 942](#) lists supported data types and conversions for *input* container variables.

Table 137: Data Types and Conversions for Input Container Variables

Input Container Variable Type	JavaScript Type	Database Type
Integer	Integer, String	Integer
String	String	String
<ul style="list-style-type: none"> • IPv4 Address • IPv6 Address • IPv6 Prefix • IPv6 Interface • Time 	Unformatted Binary String (<i>not recommended</i>)	<Binary Type>

The JavaScript Type column refers to the JavaScript variable types to be passed to the `DataAccessor.SetInputVariable()` function for each input container variable type.

NOTE: We recommend that you not configure the `.gen` file with input container variables of these types:

- IPv4 Address
- IPv6 Address
- IPv6 Prefix
- IPv6 Interface
- Time

Instead, represent these data types as formatted strings within the script, the data accessor, and the database.

Table 138 on page 943 lists supported data types and conversions for *output* container variables.

Table 138: Data Types and Conversions for Output Container Variables

Output Container Variable Type	JavaScript Type	Database Type
Integer	Integer	String
String	String	String
<ul style="list-style-type: none">• IPv4 Address• IPv6 Address• IPv6 Prefix• IPv6 Interface	Type-Specific Formatted String	String
Time	Time-String (<code>yyyy/mm/dd [hh[:mm[:ss]]]</code>)	String

All columns and fields referred to by output container variables must contain string data. Integer output container variables are returned to the script as JavaScript integers. All other types are converted to formatted JavaScript strings.

Data Accessor Configuration File Examples

The following examples show sample LDAP and SQL data accessor configuration files. The script examples shown in [“LDAP Script Examples” on page 888](#), [“Realm Selection Script Examples” on page 902](#), and [“Attribute Filter Script Examples” on page 914](#) also refer to these data accessor configurations.

Example: LDAP Data Accessor Configuration File

```
[Bootstrap]
LibraryName=ldapaccessor.dll
Enable=0

;
;Define data accessor name, search name, and connection settings
;
[Settings]
MethodName=LdapAccessor
Timeout=20
ConnectTimeout=25
QueryTimeout=10
WaitReconnect=2
MaxWaitReconnect=360
UpperCaseName = 0
SSL = 0
Search = DoLdapSearch

;
;Define the server address and bind credentials
;
[Server]
s1=

[Server/s1]
Host=<LDAPServerNameOrIPAddress>
Port = 389
BindName=uid=admin, ou=sales, o=bigco.com
BindPassword=secret

;
;Map the input variables
;
[Request]
Input-Var = value1
```

```

;
;Map the output variables
;
[Response]
Output-Var = item2

;
;Specify attributes to be copied from search response
;
[Attributes/LdapAttrs]
item2

;
; Define the search filter
;
[Search/DoLdapSearch]
Base = o=bigco.com
Scope = 2
Filter = item1=<value1>
Attributes = LdapAttrs
Timeout = 20
%DN = dn

;
;Define the input and output variable types; default is string
;
[VariableTypes]
Input-Var = string
Output-Var = string

```

Example: SQL Data Accessor Configuration File

```

[Bootstrap]
LibraryName=sqlaccessor.so
Enable=0

;
;Define data accessor name, SQL query, and connection settings
;
[Settings]
MethodName=SQLAccessor
SQL=SELECT item1, item2 FROM tablename WHERE item3 = @Input-Var1

```

```
Connect=DSN=<dsn_name_here>;UID=<username_for_dB>;PWD=<password_for_dB>
ParameterMarker=?
MaxConcurrent=1
ConcurrentTimeout=30
ConnectTimeout=25
QueryTimeout=25
WaitReconnect=2
MaxWaitReconnect=360

;
;Map output variables to SQL query result string columns
;
[Results]
Output-Var1 = 1/32
Output-Var2 = 2/64

;
Define the input and output variable types; default is string
[VariableTypes]
Output-Var1 = string
Output-Var2 = string
Input-Var1 = integer
```


Script Reference

IN THIS CHAPTER

- [JavaScript Types | 947](#)
- [API Method Support by Script Type | 948](#)
- [Local and Global Variable Declarations | 949](#)
- [Global Object | 950](#)
- [Ldap Object | 952](#)
- [LdapVariables Object | 954](#)
- [RealmSelector Object | 956](#)
- [SessionControl Object | 962](#)
- [AttributeFilter Object | 964](#)
- [DataAccessor Object | 969](#)

This chapter contains a variety of reference material applicable to all types of RADIUS scripting. This chapter contains these topics:

JavaScript Types

JavaScript types are not defined explicitly. Variable data can be any of these types:

- Integer
- Floating point
- Boolean
- String
- Array of one of the previous types
- Object

Script developers can use the full capabilities of the JavaScript language, such as defining new object types and storing data in arrays. However, the **AttributeFilter**, **RealmSelector**, and **DataAccessor** APIs use simple JavaScript types, and there is no support in the scripting API for structured IP address or time value types.

Steel-Belted Radius Carrier data types are represented in JavaScript variables as follows:

- Signed and unsigned integers are stored as JavaScript integers.
- Binary objects (HEXSTRINGS) are stored as hexadecimal coded JavaScript strings.
- Other data types (IP address types and times) are converted to formatted JavaScript strings.

NOTE: The Steel-Belted Radius Carrier scripting API does not support binary conversion of attributes or data accessor variable values from or to unformatted binary strings.

API Method Support by Script Type

Some API methods are supported for all scripts, while others are only supported for one or two type(s) of scripts. For details about what methods are supported by the scripts, see [Table 139 on page 948](#).

Table 139: API Method Support by Script Type

Object	Method	LDAP Script	Realm Selection Script	Attribute Filter Script
Global Object	SbrWriteToLog ()	✓	✓	✓
	SbrTrace ()	✓	✓	✓
LDAP LDAPVariables	Ldap.Search	✓	–	–
	LdapVariables.Get	✓	–	–
	LdapVariables.Add	✓	–	–
	LdapVariables.Reset	✓	–	–

Table 139: API Method Support by Script Type *(continued)*

Object	Method	LDAP Script	Realm Selection Script	Attribute Filter Script
RealmSelector	Execute ()	-	✓	-
	SetAuthUserName ()	-	✓	-
	SetAuthProfile ()	-	✓	-
AttributeFilter	Get ()	-	✓	✓
	Add ()	-	-	✓
	Reset ()	-	-	✓
	Replace ()	-	-	✓
	Execute ()	-	-	✓
DataAccessor	SetInputVariable ()	-	✓	✓
	GetOutputVariable ()	-	✓	✓
	Execute ()	-	✓	✓
	Clear ()	-	✓	✓

NOTE: Attempting to call an unsupported method from a script causes the script to return **SCRIPT_RET_FAILURE** and an error message to appear in the log.

Local and Global Variable Declarations

Variables declared using the **var** keyword are local variables, which persist only for the lifetime of the scope in which they are declared. Top-level local variables in your script are deleted when the script returns.

However, variables declared without the **var** keyword become properties of the global object. Global variables and their data persist between calls to your script.

NOTE: To avoid unintended consequences, such as memory leaks or security breaches, use local variables by declaring them using the **var** keyword:

```
var count = 0; //Local variable declaration

filter = new AttributeFilter(); //Persistent global property
```

Global Object

The global object is available to all scripts at all times and does not need to be instantiated explicitly.

Logging and Diagnostic Methods

You can use the **SbrWriteToLog()**, **SbrWriteToLogEx()** and **SbrTrace()** methods to insert messages and trace information into the Steel-Belted Radius Carrier log file.

SbrWriteToLog()

Purpose

The **SbrWriteToLog()** method writes text strings to the Steel-Belted Radius Carrier log file.

NOTE: The **SbrWriteToLog()** method must not be used to log variable data.

Syntax

SbrWriteToLog(msg, [logLevel])

Parameters

Table 140: SbrWriteToLog() Parameters

Parameter	Description
<i>logLevel</i>	Specifies the log level of the message. The log level configured in Steel-Belted Radius Carrier must be greater than or equal to this value for the message to appear in the log. Optional log level value in range of 0-2.

Table 140: SbrWriteToLog() Parameters (*continued*)

Parameter	Description
<i>msg</i>	Specifies the message text to be logged.

Returns

Nothing.

Example

```
SbrWriteToLog("Hello, world.", 0);
```

SbrWriteToLogEx()**Purpose**

The **SbrWriteToLogEx()** method supports a format specifier and up to five data arguments. This method must be used when writing variable data.

Syntax

```
SbrWriteToLogEx(loglevel, format specifier, var1,var2,var3,var4,var5)
```

Parameters

Table 141: SbrWriteToLogEx() Parameters

Parameter	Description
<i>loglevel</i>	Specifies the log level of the message. The log level configured in Steel-Belted Radius Carrier must be greater than or equal to this value for the message to appear in the log. Optional log level value in range of 0-2.
<i>format specifier</i>	Applies the format specifiers to the Steel-Belted Radius Carrier log.
<i>var1,var2,var3,var4,var5</i>	Specifies the variables that can be logged.

Returns

Nothing.

Example

```
var MSISDN = filter.Get("Funk-SS7-MSISDN");
SbrWriteToLogEx(2,"MSISDN=%x",MSISDN);
```

SbrTrace()

Purpose

The **SbrTrace()** method writes a script trace to the log from the point in the script where the statement appears.

Syntax

```
SbrTrace([logLevel])
```

Parameters

Table 142: SbrTrace() Parameters

Parameter	Description
<i>logLevel</i>	Specifies the log level of the message. The log level configured in Steel-Belted Radius Carrier must be greater than or equal to this value for the message to appear in the log. Optional log level value in range of 0-2.

Returns

Nothing.

Example

SbrTrace(2);

Ldap Object

The **Ldap** object exposes methods for invoking LDAP queries from scripts.

Ldap Methods

Ldap.Search()

Purpose

The **Ldap.Search()** method is used to invoke an LDAP query defined in a [Search/name] section of the **ldapauth.aut** file.

Syntax

Ldap.Search(SearchSection)

Parameters

Table 143: Ldap.Search() Parameters

Parameter	Description
SearchSection	Specifies the name of a [Search/name] section of the ldapauth.aut file.

Returns

Table 144: Ldap.Search() Return Parameters

Parameter	Description
Ldap.FOUND	The search found a matching LDAP entry.
Ldap.NOTFOUND	The search did not find a matching LDAP entry.
Ldap.FAILED	The search encountered an unexpected failure.
Ldap.NOSUCHSEARCH	The specified section was not found in ldapauth.aut .
Ldap.PASSWORDFAILED	The search found the username, but the password is incorrect.

Example

Given this query definition:

```
[Search/vpn]
Base = ou=vpn dc=jcn dc=com
Scope = 2
Filter = uid=<Vpn-User-Name>
Attributes = VpnAttrList
Timeout = 20
%DN = dn
```

Use this JavaScript command to invoke the query:

```
Ldap.Search("vpn");
```

LdapVariables Object

The **LdapVariables** object exposes methods for manipulating attributes in the LDAP plug-in variable table.

NOTE: If you are trying to access a binary attribute value, you must use the “= bin” syntax for the attribute in the [Attributes] section of **ldapauth.aut** file. The result is returned to JavaScript as a raw binary (not hexadecimal-coded) string.

LdapVariables Methods

LdapVariables.Get()

Purpose

The **LdapVariables.Get()** method retrieves the current value or values for a variable stored in the LDAP variable table. **LdapVariables.Get()** can be used to retrieve binary (raw) data or text strings.

Syntax

```
LdapVariables.Get( variableName [, nItem ])
```

Parameters

Table 145: LdapVariables.Get() Parameters

Parameter	Description
<i>VariableName</i>	Specifies the name of the variable in the variable table.
<i>nItem</i>	Specifies the index of the value for multi-valued attributes. You can specify the value of <i>nItem</i> as part of the command, or you can use a separate variable to set the value of <i>nItem</i> (as shown in this example).

Returns

The value of the specified attribute, or a null value if the attribute does not exist or if the index is out of bounds.

Examples

This example illustrates a single-value attribute lookup:


```
var name = LdapVariables.Get("User-Name" );
```

This example illustrates a multi-value attribute lookup:

```
var index = 2;  
var alias = LdapVariables.Get("User-Alias", index);
```

LdapVariables.Add()

Purpose

The **LdapVariables.Add()** method creates a new variable or adds a value to an existing variable.

Syntax

LdapVariables.Add(variableName, value[, raw])

Parameters

Table 146: LdapVariables.Add() Parameters

Parameter	Description
<i>variableName</i>	Specifies the name of the variable to be created or updated.
<i>value</i>	Specifies the value of the variable, which might be text or binary data.
<i>raw</i>	<ul style="list-style-type: none">• If set to True, specifies that the value is binary data.• If set to False, specifies that the value is text. Default value is False.

Returns

Nothing.

Example

LdapVariables.Add("Vpn-User-Name", "Fred");

LdapVariables.Reset()

Purpose

The **LdapVariables.Reset()** method deletes all of the values of the specified variable from the variable table. If the specified variable does not exist, the method call is ignored.

Syntax

LdapVariables.Reset(variableName)

Parameters

Table 147: LdapVariables.Reset() Parameters

Parameter	Description
<i>VariableName</i>	Specifies the name of the variable to be removed from the table.

Returns

Nothing.

Example

```
LdapVariables.Reset("Vpn-User-Name");
```

RealmSelector Object

Objects of the **RealmSelector** class are used to execute built-in Steel-Belted Radius Carrier realm selection methods from realm selection scripts. To use the **RealmSelector** class, scripts must first create an object instance by calling the **new RealmSelector()** constructor.

Constructor

new RealmSelector()

Purpose

The **new RealmSelector()** constructor is used to create a new object instance of the **RealmSelector** class.

Syntax

new RealmSelector()

Parameters

Nothing.

Returns

The **new RealmSelector()** object reference.

Example

```
filter = new RealmSelector();
```

new CSTAccessor()

Purpose

The **new CSTAccessor()** constructor is used to create a new object instance of the **CSTAccessor** class.

Syntax

new CSTAccessor(fieldNameArray)

Parameters

Table 148: new CSTAccessor() Parameters

Parameter	Description
<i>fieldNameArray</i>	This is an array of strings and each string is the name of a CST field that may be read in a future Get() operation. If a field is not named in the fieldNameArray parameter, an error is returned if it is an argument to a Get() method for this instance.

Returns

The **new CSTAccessor()** object reference. If there is no CST record associated with this JavaScript instance (the current radius transaction), the script terminates and returns SBR_RET_FAILURE.

Example

```
var myFields = new Array ("Sbr_UserName," "Sbr_IpPoolName," "Sbr_NasName");
var CST = new CSTAccessor(myFields);
```

new SessionControl()

Purpose

The **new SessionControl()** constructor is used to create a new object instance of the **SessionControl** class.

Syntax

new SessionControl()

Parameters

Nothing.

Returns

The **new SessionControl()** object reference. If there is no CST record associated with this JavaScript instance, the script terminates and returns SBR_RET_FAILURE.

Example

```
success = (sc.Execute() == SessionControl.SUCCESS);
```

RealmSelector Methods

These methods are provided to execute built-in realm selection methods, set the authentication username passed to the selected directed or proxy realm, and set the user profile upon return from the realm.

Execute()

Purpose

The **Execute()** method is used to execute built-in Steel-Belted Radius Carrier realm selection methods.

Syntax

selector.Execute(method)

Parameters

Table 149: Execute() Parameters—RealmSelector Methods

Parameter	Description
<i>method</i>	Specifies the name of the realm selection method: prefix , suffix , dnis , attribute-mapping , or undecorated .

Returns

The selected realm name or JavaScript null if no realm is matched.

Example

```
selector.Execute("suffix");
```

NOTE: Before you can execute the prefix or suffix built-in methods, you must enable them in the **proxy.ini** file. If a script has been declared in the [Processing] section of the **proxy.ini** file, then the prefix and suffix methods are enabled automatically. However, if a script has been declared in an [Inner_Authentication] section of a tunneled **.aut** file of an authentication method (TTLS, or PEAP), then you must explicitly declare the prefix and suffix methods in the [Processing] section of the **proxy.ini** file.

SetAuthUserName()

Purpose

The **SetAuthUserName()** method is used to set the authentication username for the request.

Syntax

selector.SetAuthUserName(username)

Parameters

Table 150: SetAuthUserName() Parameters—RealmSelector Methods

Parameter	Description
<i>username</i>	Specifies the desired authentication username.

Returns

Nothing.

Example

```
selector.SetAuthUserName("RButler");
```

SetAuthProfile()

Purpose

The **SetAuthProfile()** method is used to set the profile to be returned upon successful authentication.

Syntax

selector.SetAuthProfile(profile)

Parameters

Table 151: SetAuthProfile() Parameters—RealmSelector Methods

Parameter	Description
<i>profile</i>	Specifies the profile name.

Returns

Nothing.

Example

```
selector.SetAuthProfile("Profile1");
```

SetLocationGroupProfile()

Purpose

The **SetLocationGroupProfile()** method is used to set the profile to be returned upon successful authentication.

Syntax

selector.SetLocationGroupProfile(profile)

Parameters

Table 152: SetLocationGroupProfile() Parameters—RealmSelector Methods

Parameter	Description
<i>location group profile</i>	Specifies the location group profile name.

Returns

Nothing.

Example

```
selector.SetLocationGroupProfile("Profile1" );
```

CSTAccessor Methods

These methods are provided to get a field value from the CST for the current session.

Get()

Purpose

The **Get()** method is used to get a field value from the CST for the current session.

Syntax

Get(name)

Parameters

Table 153: Get() Parameters—CSTAccessor Methods

Parameter	Description
<i>name</i>	Specifies the name of the CST field.

Returns

The value of the CST field for this session. For standard fields, the JavaScript type of the return value is similar to the **AttributeFilter.Get()** method for the attribute used to create a particular field, listed in [Table 164 on page 968](#). For user-created fields, the JavaScript type is either a string or integer. If the field name passed as a parameter is not one of the fields passed into the constructor, the script terminates and returns a SBR_RET_FAILURE return code.

Example

```
var username = CST.Get("Sbr_UserName");
```

SetAuthUserName()

Purpose

The **SetAuthUserName()** method is used to set the authentication username for the request.

Syntax

selector.SetAuthUserName(username)

Parameters

Table 154: SetAuthUserName() Parameters—CSTAccessor Methods

Parameter	Description
<i>username</i>	Specifies the desired authentication username.

Returns

Nothing.

Example

```
selector.SetAuthUserName("RButler");
```

SetAuthProfile()**Purpose**

The **SetAuthProfile()** method is used to set the profile to be returned upon successful authentication.

Syntax

selector.SetAuthProfile(profile)

Parameters

Table 155: SetAuthProfile() Parameters—CSTAccessor Methods

Parameter	Description
<i>profile</i>	Specifies the profile name.

Returns

Nothing.

Example

```
selector.SetAuthProfile("Profile1");
```

SetLocationGroupProfile()**Purpose**

The **SetLocationGroupProfile()** method is used to set the profile to be returned upon successful authentication.

Syntax

selector.SetLocationGroupProfile(profile)

Parameters**Table 156: SetLocationGroupProfile() Parameters—CSTAccessor Methods**

Parameter	Description
<i>location group profile</i>	Specifies the location group profile name.

Returns

Nothing.

Example

```
selector.SetLocationGroupProfile("Profile1");
```

SessionControl Object

The SessionControl object allows the JavaScript to initiate session management and control capabilities.

NOTE: The use of SessionControl objects requires a SCM license.

Properties**SUCCESS**

The **SUCCESS** property is returned by the **Execute()** method when the session control action succeeds.

Example

```
success = (sc.Execute() == SessionControl.SUCCESS);
```

FAILURE

The **FAILURE** property is returned by the **Execute()** method when the session control action fails.

Example

```
failure = (sc.Execute() == SessionControl.FAILURE);
```

TIME_OUT

The **TIME_OUT** property is returned by the **Execute()** method when the session control action times out.

Example

```
time_out = (sc.Execute() == SessionControl.TIME_OUT);
```

MISSING_INFO

The **MISSING_INFO** property is returned by the **Execute()** method when the session control action needs more information to execute the action requested.

Example

```
missing_info = (sc.Execute() == SessionControl.MISSING_INFO);
```

SessionControl Methods

These SessionControl methods allow the JavaScript to initiate session management and control capabilities.

AddAttribute()

Purpose

The **AddAttribute()** method is used to add a named attribute to the Session Control action. If an attribute of the same name exists, the new data value is appended to the end of the attribute list.

Syntax

AddAttribute(name, value)

Parameters

Table 157: AddAttribute() Parameters

Parameter	Description
<i>name</i>	Specifies the name of the attribute dictionary.
value	Specifies a value for the attribute

Returns

Nothing.

Example

```
test.AddAttribute("Unisphere-Med-Ip-Address", "1.1.1.1");
```

Execute()

Purpose

The **Execute()** method is used to execute built-in Steel-Belted Radius Carrier CoA/DM actions.

Syntax

`selector.Execute(name)`

Parameters

Table 158: Execute() Parameters—SessionControl Methods

Parameter	Description
Name	<p>Specifies the name of the CoA/DM action to execute. The deviceModels.xml file defines the CoA/DM actions supported by the client devices in your network.</p> <p>For more information, see “Overview of the Optional Session Control Module” on page 700.</p>

Returns

SUCCESS, FAILURE, TIME_OUT, or MISSING_INFO properties.

Example

```
success = (test.Execute("interceptOn" ) == SessionControl.SUCCESS);
```

AttributeFilter Object

Objects of the **AttributeFilter** class are used to read, write, modify, and reset attributes in the RADIUS request or response packets. To use the **AttributeFilter** class, scripts must first create an object instance by calling the **new AttributeFilter()** constructor.

In javascripting, **new AttributeFilter()** should take either `AttributeFilter.ATTR_LIST_REQUEST` or `AttributeFilter.ATTR_LIST_RESPONSE` as an argument. This ensures that the new object will always point to either the request or response list attributes. Without this argument in the constructor, the new object might not always point to the appropriate list.

Constructor

new AttributeFilter()

Purpose

The **new AttributeFilter()** constructor is used to create a new object instance of the **AttributeFilter** class.

Syntax

`new AttributeFilter()`

Parameters

Nothing.

Returns

The **new AttributeFilter()** object reference.

Example

```
filter = new AttributeFilter();
```

AttributeFilter Methods

Methods of the **AttributeFilter** class operate on either the request or the response attribute list, depending on the context in which they are called. All of the methods of the **AttributeFilter** class are available to attribute filter scripts. The **AttributeFilter.Get()** method is also available to realm selection scripts.

Get()

Purpose

The **Get()** method is used to get a named attribute from the request or response attribute list.

Syntax

Get(name, [index])

Parameters

Table 159: Get() Parameters—AttributeFilter Methods

Parameter	Description
<i>name</i>	Specifies the name of the attribute dictionary.
<i>index</i>	Specifies the optional index for multi-value attributes; default is 0.

Returns

The attribute value or JavaScript null if not found.

Example

```
filter.Get("Calling -Station-ID" );
```

Add()

Purpose

The **Add()** method is used to add a named attribute to the request or response attribute list. If an attribute of the same name already exists, the new data value is appended to the end of the attribute list.

Syntax**Add(name, value)****Parameters****Table 160: Add() Parameters**

Parameter	Description
<i>name</i>	Specifies the name of the attribute dictionary.
<i>value</i>	Specifies the string representation of the attribute value.

Returns

Nothing.

Example**filter.Add("NAS-IP-Address", "1.2.3.4");****Reset()****Purpose**

The **Reset()** method is used to remove one or all instances of a named attribute from the request or response attribute list. If the index is given, the specified value is removed from the attribute list and higher-indexed values drop down to fill in the vacancy.

Syntax**Reset(name, [index])****Parameters****Table 161: Reset() Parameters**

Parameter	Description
<i>name</i>	Specifies the name of the attribute dictionary.
<i>index</i>	Specifies the optional index for multi-value attributes; default is remove all.

Returns

Nothing.

Example**filter.Reset("NAS-IP-Address", 3);**

Replace()**Purpose**

The **Replace()** method is used to delete a named attribute and add a new attribute of the same name and given value. This is equivalent to **Reset()** followed by **Add()**. **When** an index is given, the specified attribute value is removed, the remaining higher-indexed values drop down to fill in the vacancy, and the new value is appended to the end of the attribute list. List order is not preserved.

Syntax

Replace(name, value, [index])

Parameters

Table 162: Replace() Parameters

Parameter	Description
<i>name</i>	Specifies the name of the attribute dictionary.
value	Specifies the string representation of the new attribute value.
<i>index</i>	Specifies the optional index for multi-value attributes; default is remove all.

Returns

Nothing.

Example

```
filter.Replace("Calling-Station-ID", "5551212", 2);
```

Execute()**Purpose**

The **Execute()** method is used to execute a statically-defined attribute filter.

Syntax

Execute(name)

Parameters

Table 163: Execute() Parameters—AttributeFilter Methods

Parameter	Description
<i>name</i>	Specifies the name of the filter defined in filter.ini .

Returns

Nothing.

Example

```
filter.Execute("MyStaticFilter");
```

AttributeFilter API

Table 164 on page 968 provides a list of the RADIUS attribute data types supported by the **AttributeFilter** API.

Table 164: AttributeFilter API Data Types

RADIUS Attribute Type	JavaScript Type
<ul style="list-style-type: none"> 8-bit integer 32-bit integer 32-bit hex (HEX4) (Get only) 	integer
<ul style="list-style-type: none"> Fixed-length string Zero-terminated string 	string
Arbitrary length hex (HEXSTRING)	hexadecimal-coded string
<ul style="list-style-type: none"> IPv4 address IPv4 address pool IPv6 address IPv6 prefix IPv6 interface 	type-specific formatted string
Time	time string (<i>yyyy/mm/dd [hh[:mm[:ss]]]</i>)

Script developers do not control attribute types; a given attribute’s type is determined from its name and dictionary entry. Many bit-length variations of integer and hex data types are possible, but not all are supported or used by Steel-Belted Radius Carrier.

When **AttributeFilter.Get()** is called, the binary data in the attribute are converted into a JavaScript variable. When either **AttributeFilter.Add()** or **AttributeFilter.Replace()** is called, the data in a JavaScript variable are converted into the appropriate RADIUS attribute binary type.

NOTE: Due to an internal limitation in Steel-Belted Radius Carrier, the HEX4 attribute type is supported for **Get()** only. Attempting to **Add()** or **Replace()** a variable of type HEX4 causes a script error.

DataAccessor Object

Objects of the **DataAccessor** class enable scripts to perform SQL queries and LDAP searches using the Steel-Belted Radius Carrier SQL and LDAP data accessor plug-ins.

All **DataAccessor** methods are available to both realm selection and attribute filter scripts.

Properties

FOUND

Purpose

The **FOUND** property is returned by the **Execute()** method when the requested data object is found.

Example

```
found = (da.Execute() == DataAccessor.FOUND);
```

NOTFOUND

Purpose

The **NOTFOUND** property is returned by the **Execute()** method when the requested data object is not found.

Example

```
notfound = (da.Execute() == DataAccessor.NOTFOUND);
```

FAILED

Purpose

The **FAILED** property is returned by the **Execute()** method when the data request fails.

Example

```
failed = (da.Execute() == DataAccessor.FAILED);
```

PASSWORDFAILED

Purpose

The **PASSWORDFAILED** property is returned by the **Execute()** method when the requested data object did not have the correct password.

Example

```
passwordfailed = (da.Execute() == DataAccessor.PASSWORDFAILED);
```

Constructor

The data accessor name is given as an argument to the object constructor.

new DataAccessor()

Purpose

The **new DataAccessor()** object instance is required to use the **DataAccessor** methods.

Syntax

new DataAccessor(name)

Parameters

Table 165: new DataAccessor() Parameters

Parameter	Description
<i>name</i>	Specifies the name of the data accessor, as configured in the .gen file.

Returns

The **new DataAccessor()** object reference.

Example

```
da = new DataAccessor("LDAP-Accessor");
```

Methods

SetInputVariable()

Purpose

The **SetInputVariable()** method is used to set a named variable in the data accessor input container.

Syntax

accessor.SetInputVariable(name, value)

Parameters

Table 166: SetInputVariable() Parameters

Parameter	Description
<i>name</i>	Specifies the name of the input container variable from the data accessor .gen file.
<i>value</i>	Specifies the string representation of the input variable value.

Returns

Nothing.

Example

```
SetInputVariable ("Phone-Number", "5551212" );
```

```
GetOutputVariable()
```

Purpose

The **GetOutputVariable()** method is used to get a named variable from the data accessor output container.

Syntax

```
accessor.GetOuputVariable(name,[index])
```

Parameters

Table 167: GetOutputVariable() Parameters

Parameter	Description
<i>name</i>	Specifies the name of the output container variable from the data accessor .gen file.
<i>index</i>	(Optional) Specifies the index value for multivalued attributes. You can specify the value for <i>index</i> as part of the command, or you can use a separate variable to set the value for <i>index</i> . Default value is 0.

Returns

The string representation of the output variable value.

Example

The following example illustrates a single-value attribute lookup:

```
var name = accessor.GetOutputVariable("User-Name");
```

The following example illustrates a multivalued attribute lookup:

```
var index = 2;
```

```
var address = accessor.GetOutputVariable("Street-Address",index);
```

```
Execute()
```

Purpose

The **Execute()** method is used to execute the query or search using the current input container data values.

Syntax

```
accessor.Execute()
```

Parameters

Nothing.

Returns

FOUND, NOTFOUND, OR FAILED properties.

Example

```
found = (da.Execute() == DataAccessor.FOUND);
```

Clear()**Purpose**

The **Clear()** method is used to clear accessor input and output container values.

Syntax

```
accessor.Clear()
```

Parameters

Nothing.

Returns

Nothing.

Example

```
accessor.Clear();
```

PART 13

Appendixes

This part contains the following appendixes:

APPENDIX A

When and How to Stop and Restart Steel-Belted Radius Carrier

IN THIS SECTION

- [Stopping the Steel-Belted Radius Carrier Server | 977](#)
- [Starting the Steel-Belted Radius Carrier Server | 977](#)

This appendix explains when and how to stop and restart the Steel-Belted Radius Carrier server. This appendix contains these topics:

[Table 168 on page 974](#) indicates when to stop and restart the server. The *least* drastic action that causes this change to take effect is indicated by **Yes**.

Table 168: When to Stop and Restart Steel-Belted Radius Carrier

Item changes:	Save the window or file	Issue a SIGHUP (1) signal	Stop/restart the server
Access window or object	Yes	(Also works)	(Also works)
access.ini file	No	Yes	Yes
*.acc files	No	No	Yes
account.ini file	No	No	Yes

Table 168: When to Stop and Restart Steel-Belted Radius Carrier (continued)

Item changes:	Save the window or file	Issue a SIGHUP (1) signal	Stop/restart the server
admin.ini file	No	Yes	Yes
*.aut files	No	(Sometimes)	Yes
blacklist.ini file	No	No	Yes
Authentication policy	Yes	(Also works)	(Also works)
*.dct files	No	No	Yes
*.dhc files	No	No	Yes
dhcp.ini file	No	No	Yes
*.dic files	No	No	Yes
*.dir files	No	(Sometimes)	Yes
*.eap files	No	(Sometimes)	Yes
eap.ini file	No	No	Yes
enterprises.oid file)	No	No	Yes
events.ini file	No	No	Yes
filter.ini file	No	Yes	(Also works)
*.gen files	No	No	Yes
Import *.rif or users file	Yes	(Also works)	(Also works)
*.ini for directed accounting	No	No	Yes
IP Pools dialog or object	Yes	(Also works)	(Also works)
lockout.ini file	No	No	Yes
Log levels (in radius.ini file)	No	Yes	(Also works)

Table 168: When to Stop and Restart Steel-Belted Radius Carrier (continued)

Item changes:	Save the window or file	Issue a SIGHUP (1) signal	Stop/restart the server
Profiles dialog or object	Yes	(Also works)	(Also works)
Proxy dialog or object	Yes	(Also works)	(Also works)
*.pro files	No	Yes	(Also works)
proxy.ini file	No	(Sometimes)	Yes
radius.dct file	No	No	Yes
radius.ini file	No	(Sometimes)	Yes
RADIUS Clients dialog or object	Yes	(Also works)	(Also works)
Servers dialog or object	Yes	(Also works)	(Also works)
services file	No	No	Yes
snmpdx.acl file	No	No	Yes
TLS and TTLS	No	Yes	(Also works)
Trace levels	No	Yes	(Also works)
Tunnels page or object	Yes	(Also works)	(Also works)
Users dialog or object	Yes	(Also works)	(Also works)
vendor.ini file	No	No	Yes

Stopping the Steel-Belted Radius Carrier Server

After the RADIUS daemon is installed, it stops and starts automatically each time you shut down or restart the server. If you modify the settings in the Steel-Belted Radius Carrier configuration files, you may need to restart the Steel-Belted Radius Carrier server manually before the server recognizes its new settings.



CAUTION: If the SBR Carrier server is part of a Session State Register cluster, do not use this procedure.

You can stop the RADIUS daemon on a Solaris server at any time by issuing the following commands:

```
cd server-directory
./sbrd stop
```

When you execute the `sbrd stop` command, SBR Carrier allows its subsystems to complete outstanding work and release its resources gracefully.

If Steel-Belted Radius Carrier fails to stop after you issue a **sbrd stop** command, you can use the optional force argument to terminate all subsystems immediately.

```
cd server-directory
./sbrd stop force
```

Starting the Steel-Belted Radius Carrier Server

You must restart the RADIUS daemon after you modify the Steel-Belted Radius Carrier configuration files.

Use the following commands to start the RADIUS daemon on a Solaris server:

```
cd server-directory
./sbrd start
```

If you change configuration settings for your Steel-Belted Radius Carrier server, you may need to restart the server to make the changes effective. As an alternative to issuing a **sbrd stop** command immediately

followed by a **sbrd start** command, you can use the **sbrd restart** command when you want to restart Steel-Belted Radius Carrier.

```
cd server-directory  
./sbrd restart
```


APPENDIX B

Authentication Protocols

Table 169 on page 979 provides a matrix of authentication methods and their supported authentication protocols.

Table 169: Authentication Protocols

Method	PAP	CHAP	MS-CHAP	MS-CHAP-V2	EAP-MSCHAP-V2	EAP-MD5
Microsoft PEAP available inner authentication protocols	No	No	No	No	Yes	No
TTLS available inner authentication protocols	Yes	Yes	Yes	Yes	Yes	Yes
Native	Yes	Yes	Yes	Yes	Yes	Yes
Native (password saved as Allow PAP only, {SHA} or {crypt}).	Yes	No	No	No	No	No
Solaris Authentication Methods						
Solaris User	Yes	No	No	No	No	No
Solaris Group	Yes	No	No	No	No	No
LDAP						
BIND	Yes	No	No	No	No	No

Table 169: Authentication Protocols (*continued*)

Method	PAP	CHAP	MS-CHAP	MS-CHAP-V2	EAP-MSCHAP-V2	EAP-MD5
BINDNAME (password stored in clear- text)	Yes	Yes	Yes	Yes	Yes	Yes
BINDNAME (password stored in SHA/Solaris Crypt text)	Yes	No	No	No	No	No
BINDNAME (password Stored as MD4 hash of Unicode value text)	Yes	No	No	Yes	Yes	No
BINDNAME (password Stored as enc-md5)	Yes	Yes	No	No	No	No
SQL						
Password stored in clear-text	Yes	Yes	Yes	Yes	Yes	Yes
Password password stored in SHA/Solaris Crypt text	Yes	No	No	No	No	No
Password stored as {MD4} hash of Unicode value text	Yes	No	No	Yes	Yes	No
Password stored as {enc-md5}	Yes	Yes	No	No	No	No

APPENDIX C

Importing and Exporting Data

IN THIS SECTION

- [Importing Information from an XML File | 981](#)
- [Exporting Information to an XML File | 984](#)

This appendix describes how to export database information from one SBR Carrier server to an XML file, and then selectively import database information into another SBR Carrier server. The ability to export and import information facilitates configuration of multiple SBR Carrier servers of the same type (operating system and release level) and role (standalone, primary, or replica).

SBR Carrier uses XML files with UTF-8 encoding to store exported information. Export files contain only those attributes that have been assigned values, including defaults. For example, if an exported local user item does not have a profile associated with it, the exported information does not identify a profile name, though it includes default values required by SBR Carrier.

NOTE: You cannot import or export Diameter configuration information.

This appendix contains these topics:

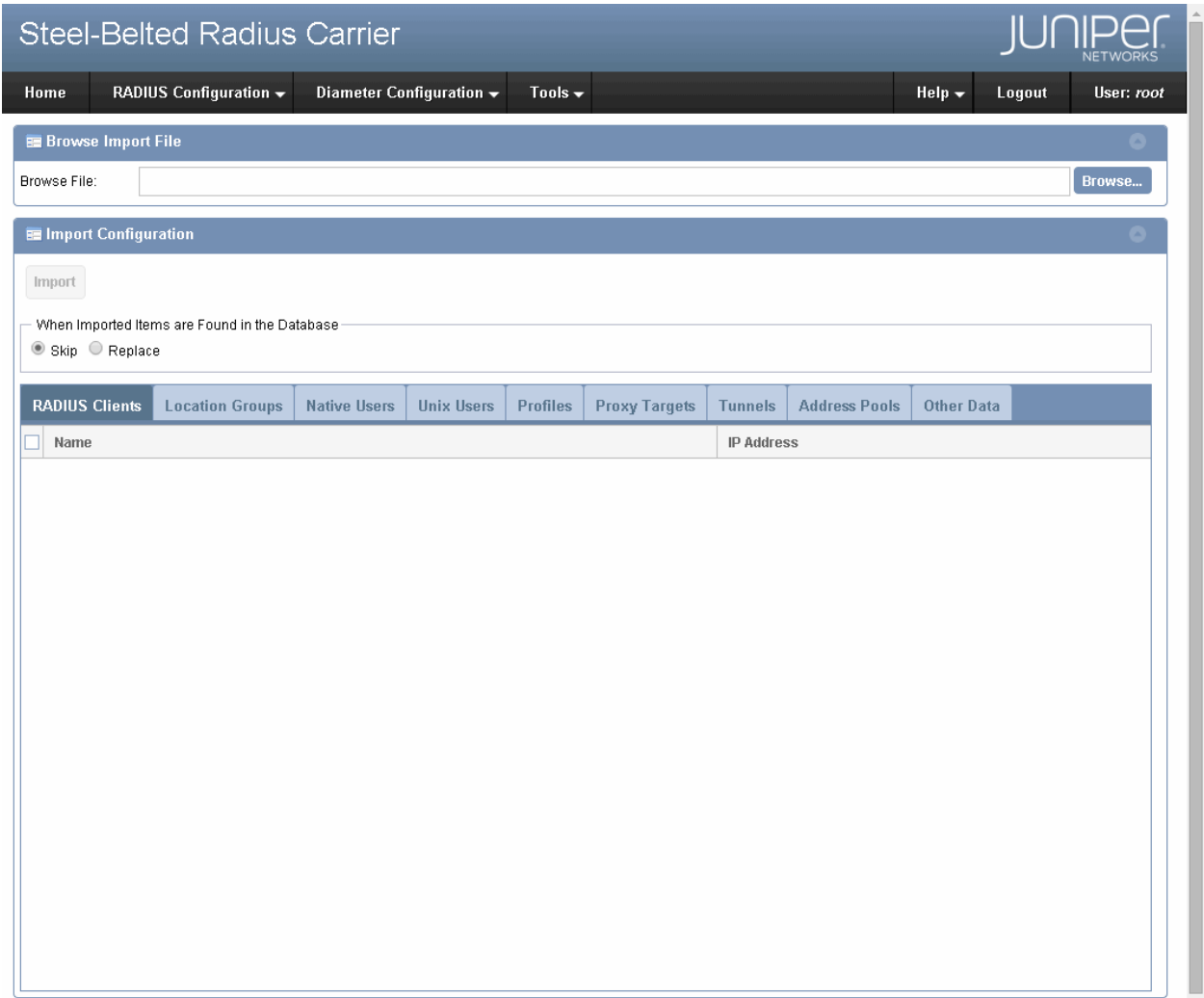
Importing Information from an XML File

To import information from an XML file into the database on your SBR Carrier server using the Web GUI:

- 1. Launch the Web GUI.
- 2. Select **Tools > Import**.

The **Browse Import File** page (Figure 286 on page 982) appears.

Figure 286: Browse Import File Page



- 3. Click **Browse** to browse the XML file containing the information you want to import, select the file, and click **Open**.

Each tab in the **Import Configuration** pane (Figure 287 on page 983) lists the imported items of a particular category.

Figure 287: Import Configuration Pane

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

Browse Import File

Browse File:

Import Configuration

When Imported Items are Found in the Database

☒ Skip ☐ Replace

RADIUS Clients		Location Groups	Native Users	Unix Users	Profiles	Proxy Targets	Tunnels	Address Pools	Other Data
<input checked="" type="checkbox"/>	Name	IP Address							
<input checked="" type="checkbox"/>	<ANY>								
<input checked="" type="checkbox"/>	SHARMI	10.212.98.29							
<input checked="" type="checkbox"/>	SHARMI6	10.212.98.66							
<input checked="" type="checkbox"/>	TEST	10.212.98.36							
<input checked="" type="checkbox"/>	USER	10.212.98.65							

4. Specify what you want the Web GUI to do when it finds an object with the same name in the SBR Carrier database.
 - Select the **Skip** option button if you want the Web GUI to leave the item already in the database intact.
 - Select the **Replace** option button if you want the Web GUI to overwrite the item in the database with the imported information.
5. Select the information you want to import by clicking each tab and selecting items to import.

To select a contiguous range of items, select the first item in the range, hold down the **Shift** key, and click the last item in the range.

To select a non-contiguous set of items, hold down the **Ctrl** key as you click each item you want.

To select all items in a category, select the check box of the corresponding column header.

6. Click **Import** to import the selected items to the SBR Carrier database.

Exporting Information to an XML File

To export information from the SBR Carrier database to an XML file using the Web GUI:

1. Launch the Web GUI.
2. Select **Tools > Export**.

The **Export Configuration** page ([Figure 288 on page 984](#)) appears.

Figure 288: Export Configuration Page

Steel-Belted Radius Carrier

JUNIPER NETWORKS

Home RADIUS Configuration Diameter Configuration Tools Help Logout User: root

Export Configuration

Export

RADIUS Clients	Location Groups	Native Users	Unix Users	Profiles	Proxy Targets	Tunnels	Address Pools	Other Data
<input checked="" type="checkbox"/> Name								IP Address
<input checked="" type="checkbox"/> <ANY>								
<input checked="" type="checkbox"/> SHARMI								10.212.98.29
<input checked="" type="checkbox"/> SHARMI6								10.212.98.66
<input checked="" type="checkbox"/> TEST								10.212.98.36
<input checked="" type="checkbox"/> USER								10.212.98.65

3. Select the information you want to export.

Each tab in the **Export Configuration** page lists exportable items of a particular category. For each category, select the appropriate tab and select items you want to export.

To select a contiguous range of items, select the first item in the range, hold down the **Shift** key, and click the last item in the range.

To select a non-contiguous set of items, hold down the **Ctrl** key as you click each item you want.

To select all items in a category, select the check box of the corresponding column header.

4. After you have selected the items you want to export, click **Export** to save the selected items as an XML file.

APPENDIX D

Technical Bulletins

IN THIS SECTION

- [Service Type Mapping | 986](#)
- [Ascend Filter Translation | 994](#)
- [Changing IP Addresses in an SSR Cluster Without Redefining the Cluster | 997](#)

This appendix contains the following technical bulletins:

Service Type Mapping

Service type mapping enables a single user to have multiple authorization attribute sets based on the service type the user is requesting. The service type is determined based on request attributes using rules that may differ depending on the NAS device.

Using static configuration parameters in the **servtype.ini** file, you can specify, on a NAS-by-NAS basis, a mapping of request attributes or values to service type strings. These strings can be attached to the username, either as a prefix or as a suffix. The elaborated username is used for both authentication and authorization, and for allowing different authorizations based on the service type requested.

Configuration

Service type mapping is configured in the following way:

- Create multiple Local User entries in the database according to specific naming and mapping conventions. For example:

```
ppp:george
vpn:george
ppp:martha
isdn:martha
```

- Define a set of rules in the **servtype.ini** file mapping each incoming Access-Request packet to the appropriate database entry for the user.

Local User Database Entries

The Local User entries you define to support service type mapping must follow a consistent naming convention. However, you are free to use any convention you like.

For example, you can store entries for PPP users using the convention **ppp:username** (for example, **ppp:george** and **ppp:martha**) and entries for VPN users using the convention **vpn:username** (**vpn:george** and **vpn:martha**).

For the mapping to work, however, you must define users who do not have any of these mapped prefixes or suffixes in the local users database. For example, if you want to map **vpn:emil** and **ppp:emil** so that the appropriate profiles are returned, you can enter three user entries in the local users database:

```
vpn:emil
ppp:emil
emil
```

Alternatively, you can omit **emil** from the local users database, authenticate **emil** against a non-local method and then apply the mapping. The mapped names still have to be in the local user database for profiles to be returned.

You can support classes of service by varying the string you use in creating Local User entries. For example, if you offer three classes of VPN service, your VPN entries might use the conventions **vpn1:username**, **vpn2:username**, and **vpn3:username** (for example, **vpn3:george** and **vpn1:martha**).

A delimiting character (such as a colon) in your service type string makes your user record names easier to read—for example, **vpn:amar** instead of **vpnamar**. When you design a service type string, consider whether it is a prefix (**string** + **separator**) or a suffix (**separator** + **string**) to the username.

NOTE: You can define Local User records using the Administration program or the LDAP configuration interface.

servtype.ini File

The **servtype.ini** configuration file controls service type mapping and contains the sections listed in [Table 170 on page 988](#).

Table 170: servtype.ini Syntax

servicetype.ini Section	Meanings
[Settings]	<p>Indicates how to attach the service type string to the username before look up in the Local User database: by prefix, by suffix, or not at all. The two fields Prefix and Suffix may be enabled (set to 1) or disabled (set to 0) independently of each other. If both are set to 0 (the default) the service type feature is completely disabled.</p> <p>Using this example, if user george requests PPP service and the string for that service type is ppp:, the Local User record with the return list for this request has the name ppp:george.</p>

Table 170: servtype.ini Syntax (continued)

servicetype.ini Section	Meanings
[NAS]	<p data-bbox="464 373 1161 441">Enables you to map NAS devices to <i>[mapping]</i> sections. The syntax for [NAS] is as follows:</p> <p data-bbox="464 472 527 499">[NAS]</p> <p data-bbox="464 531 678 558"><i>NASname = mapping</i></p> <p data-bbox="464 590 678 617"><i>NASname = mapping</i></p> <p data-bbox="464 669 472 690">.</p> <p data-bbox="464 722 472 743">.</p> <p data-bbox="464 795 472 816">.</p> <p data-bbox="464 848 678 875"><i>NASname = mapping</i></p> <p data-bbox="464 907 1161 1115">Each <i>NASname</i> in the [NAS] section must match the name of a RADIUS client entry in the database. When an Access-Request is received, its NAS-IP-Address attribute is matched to a RADIUS client entry in the database. If a match is found, and the RADIUS client name matches a <i>NASname</i> in the [NAS] section, a corresponding <i>[mapping]</i> section is found.</p>

Table 170: servtype.ini Syntax (continued)

servicetype.ini Section	Meanings
[mapping]	<p>Each section does the following:</p> <p>Names the strings to be added to the username for performing lookups in the Local User database, to find the correct return list.</p> <p>Provides a set of rules which the incoming Access-Request packet must meet if an Access-Accept is to be returned.</p> <p>The syntax for each [mapping] section is as follows:</p> <p>[mapping]</p> <p>ServiceTypeString</p> <p>RADIUSattribute = value</p> <p>~RADIUSattribute = value</p> <p>.</p> <p>.</p> <p>.</p> <p>ServiceTypeString</p> <p>RADIUSattribute = value</p> <p>RADIUSattribute = value</p> <p>.</p> <p>.</p> <p>.</p> <p>There may be zero or more rules in each ServiceTypeString section.</p> <p>Each rule is a statement about an attribute in the incoming Access-Request packet. Each rule begins with a tab character, followed by a RADIUSattribute=value string, followed by a carriage return. Every component of the rule is optional.</p> <p>If a rule provides a RADIUSattribute field, this field must name a standard or vendor-specific RADIUS attribute that is known to the server. If a rule provides an optional value field, this field must name a valid possible value for that attribute.</p>

Table 170: servtype.ini Syntax (continued)

servicetype.ini Section	Meanings

Table 170: servtype.ini Syntax (continued)

servicetype.ini Section	Meanings
	<p>The following logic is applied to the [mapping] section:</p> <ol style="list-style-type: none"> <p>The <i>next</i> (initially, the <i>first</i>) ServiceTypeString in the [mapping] section is sought. It combines the ServiceTypeString with the username as defined in the [Settings] section, and tries to find a matching Local User entry.</p> <p>If a matching entry is found, each rule in the <i>ServiceTypeString</i> section is tried against the attributes in the incoming Access-Request packet. Otherwise, if there was no next ServiceTypeString or no match could be found, the user is rejected.</p> <p>In the following example, the rule syntax is: RADIUSattribute = value</p> <p>If the RADIUSattribute named is present in the Access-Request packet, and if it has the <i>value</i> shown, this rule is true. Evaluate the next rule. If there is no next rule, select this ServiceTypeString.</p> <p>If the RADIUSattribute named is not present in the Access-Request packet, or if it is present but does not have the value shown, then control returns to step 1.</p> <p>In the following example, the rule syntax is:</p> <p style="padding-left: 40px;">RADIUSattribute</p> <p>NOTE: The absence of a value is important. If the RADIUSattribute is present in the Access-Request packet, this rule is true. Evaluate the next rule. If there is no next rule, select this ServiceTypeString.</p> <p>If the RADIUSattribute is not present in the Access-Request packet, control returns to step 1.</p> <p>In the following example, the rule syntax is:</p> <p style="padding-left: 40px;">~RADIUSattribute =value</p> <p>NOTE: Note the tilde (~) operator. If the RADIUSattribute named is present in the Access-Request packet, and if it does not have the value shown, this rule is true. Evaluate the next rule. If there is no next rule, accept this ServiceTypeString.</p> <p>If the RADIUSattribute named is not present in the Access-Request packet, or if it is present but has the <i>value</i> shown, control returns to step 1.</p>

Table 170: servtype.ini Syntax (continued)

servicetype.ini Section	Meanings
	<p>NOTE: The following is not valid syntax: RADIUSattribute = ~value</p> <p>5. In the following example, the rule syntax is:</p> <p style="padding-left: 40px;">~RADIUSattribute</p> <p>NOTE: Note the tilde (~) operator and the absence of a value. If the RADIUSattribute named is not present in the Access-Request packet, this rule is true. Evaluate the next rule. If there is no next rule, accept this ServiceTypeString.</p> <p>If the RADIUSattribute named is present in the Access-Request packet, control returns to step 1.</p> <p>6. If no RADIUSattribute rules are provided and a ServiceTypeString section exists, but contains no rules, the ServiceTypeString is selected automatically.</p> <p>7. If no ServiceTypeString sections are provided and a [mapping] section exists, but is empty, the user is rejected automatically.</p>

In addition to enabling a prefix or suffix, the [Settings] section of the **servtype.ini** file enables you to specify a default [**mapping**] section to be used when an Access-Request packet arrives from a NAS device that is not listed in the [NAS] section of **servtype.ini**. The syntax for setting this default is as follows:

```
[Settings]
Default = mapping
```

The Default field is optional. If you do not set up a default mapping, and the server cannot determine the mapping in any other way, the server ignores the service type and authenticates the user without it.

The following is a sample **servtype.ini** file:

```
[Settings]
Prefix=1
Suffix=0
Default=defaultmap

[NAS]
```

```

nas1=nas1map
nas2=nas2map

[nas1map]
ppp:
    Framed-Protocol=1
    Service-Type=2

vpn:
    Framed-Protocol=6
    ~Service-Type=2

other:
    Framed-Protocol
    Service-Type

[nas2map]
analog:
    NAS-Port-Type=1

isdn:
    NAS-Port-Type=2

[defaultmap]
ppp:

```

Any syntax error in the **servtype.ini** file prevents initialization of the file. If this occurs, service type mapping is disabled. This event is logged in the **date.log** file.

Ascend Filter Translation

Ascend defines two attributes—**Ascend-Data-Filter** (242) and **Ascend-Call-Filter** (243)—that contain structured binary data representing a filter to be applied to the NAS device.

Instead of entering hexadecimal strings to configure these attributes, users can configure these attributes as text strings. Steel-Belted Radius Carrier automatically converts the text strings to the proper binary representation. The original filter attributes are still supported, and these attributes still may be configured as hexadecimal strings.

The following attributes allow configuration as text:

Ascend-Data-Filter-String
Ascend-Call-Filter-String

When Steel-Belted Radius Carrier formats a response packet, it translates the string version of the attribute to the appropriate binary value, and returns the attribute in the Access-Accept message.

Configuration

These attributes may be entered as text strings through the Web GUI. The attributes may also be returned from an LDAP or SQL database during authentication.

No syntax validation is performed when the attribute is configured. The validation of syntax occurs only when the response packet is formatted. If the syntax is invalid, a reject response is issued and an error is logged.

NOTE: We recommend that you test these attributes before using them on a production server.

Two types of filter are supported: *ip* and *generic*.

Syntax

In the syntax descriptions in [Table 171 on page 995](#), brackets [] indicate that the items enclosed are optional.

ip [*direction*] [*action*] [*srcip address[/mask]*] [*dstip address[/mask]*] *protocol* [*srcport operator port*] [*dstport operator port*]

Table 171: Syntax

Parameter	Values
direction	May be <i>in</i> or <i>out</i> . The default is <i>out</i> .
action	May be <i>forward</i> or <i>drop</i> . The default is <i>drop</i> .
address	An IP address in decimal dotted notation.

Table 171: Syntax (continued)

Parameter	Values
mask	The number of bits (decimal) in the network portion, from 0 through 32. The default is based on class of network.
protocol	<p>The protocol number (decimal); for example, 6 for TCP or 17 for UDP. The following protocol names are translated to the proper number:</p> <p>icmp(1)</p> <p>tcp(6)</p> <p>udp(17)</p> <p>ospf(89)</p>
operator	May be = (equal sign), != (exclamation and equal sign), < (less than), or > (greater than).
port	<p>The port number (decimal). In addition, the following service names are translated to the proper port number:</p> <p>ftp-data(20)•www(80)</p> <p>ftp(21)•kerberos(88)</p> <p>telnet(23)•hostname(101)</p> <p>smtp(25)•nntp(119)</p> <p>nameserver(42)•ntp(123)</p> <p>domain(53)•exec(512)</p> <p>tftp(69)•login(513)</p> <p>gopher(70)•cmd(514)</p> <p>finger(79)•talk(517)</p>

Example:

ip out forward srcip 10.1.1.0/24 6 dstport = 80 srcport < 1023

NOTE: See your Ascend documentation for details about the syntax for these attributes.

Changing IP Addresses in an SSR Cluster Without Redefining the Cluster

The Steel-Belted Radius Carrier software generates the required configuration files that contain the cluster definition details and cluster node details. These details are configured in each SM node and D node, which form part of the SSR cluster.

When you change an IP address of an SM or D node in the cluster, you should also change the IP address of the node available under the cluster node details in the configuration files. You can change the IP address of the node by redefining the cluster or by manually changing the IP address in the generated configuration files in all other SM and D nodes. To manually change the IP address:

- In SM nodes, replace the old IP address with the new IP address in the following files:
 - **dbclusterndb.gen** in **/opt/JNPRsbr/radius**
 - **config.ini** in **/opt/JNPRhadm**
 - **my.cnf** in **/opt/JNPRhadm**
- In D nodes, replace the old IP address with the new IP address in the following files:
 - **config.ini** in **/opt/JNPRhadm**
 - **my.cnf** in **/opt/JNPRhadm**



CAUTION: Manually changing the generated configuration files in SM or D nodes might result in loss of data. It is recommended that you completely redefine the SSR cluster by configuring cluster definition and cluster node details in each SM node and D node so that the software generates updated configuration files. For more information about configuring SSR nodes, see the *SBR Carrier Installation Guide*.

APPENDIX E

SIR.sh Script

IN THIS SECTION

- Syntax | 999
- Options | 999
- Example | 1000

You can use the **SIR.sh** script to collect information from your system when the information is requested by JTAC. The collected information is stored in tar files under a directory named **SBR_Information_Report**, in the user-defined path or in the default path (**/opt/JNPRsbr/radius/debug/**). Using this script, you can collect SBR configuration information, process information, system information, package and patch information, log files, and core files. You must configure the corresponding parameters in the **SIR.conf** file to collect these information. For more information about configuring the **SIR.conf** file, see the *SBR Carrier Reference Guide*.

NOTE: Make sure that all the following basic OS commands used by this script are available in your system: **pstack**, **pfiles**, **netstat**, **lsof**, **dmidecode**, **prtdiag**, **ls**, **grep**, **rm**, **find**, **file**, **mkdir**, **ps**, **pgrep**, **prtcnf**, **netstat**, **uname**, **pfiles**, **iostat**, **df**, and **rpm**.

You cannot use the **SIR.sh** script to collect information from remote SBR Carrier servers.

This appendix contains the following sections:

Syntax

The syntax of the **SIR.sh** script is:

```
SIR.sh [ -d DestDir ] [ -r RadiusCore ] [ -s SnmpCore ]
SIR.sh -h
```

Options

Table 172 on page 999 lists the options that you can use with the **SIR.sh** script.

Table 172: SIR.sh Options

Option	Description
-d <i>DestDir</i>	Specifies the absolute path where you want to store the tar files under a directory named SBR_Information_Report .
-r <i>RadiusCore</i>	<p>Specifies the RADIUS core file (with the absolute path) to be collected. For example, ./SIR.sh -r /opt/JNPRsbr/radius/core.1624.</p> <p>NOTE: The SIR.sh script must be executed with this option if you have enabled the radius parameter in the [Core_Files] section of SIR.conf file.</p>
-s <i>SnmpCore</i>	<p>Specifies the SNMP core file (with the absolute path) to be collected. For example, ./SIR.sh -s /opt/JNPRsbr/radius/core.1626.</p> <p>NOTE: The SIR.sh script must be executed with this option if you have enabled the snmp parameter in the [Core_Files] section of SIR.conf file.</p>
-h	Displays help for the SIR.sh script.

NOTE: You can identify or differentiate RADIUS and SNMP core files by using the **file core_filename** command.

Example

You must log in to the system as a root user for executing the **SIR.sh** script. By default, the script is located in the **/opt/JNPRsbr/radius/install/debug** path. The following example displays a sample output of **SIR.sh** script executed with the **-d DestDir** option.

```
$ ./SIR.sh -d /tmp/Storage

[ INFO ]: All requested data will be stored in the directory [
/tmp/Storage/SBR_Information_Report/ ].
[ INFO ]: 8.7G available in the [ /tmp/Storage/SBR_Information_Report/ ] directory.
[ INFO ]: Configured SBR directory: [ /opt/JNPRsbr/radius/ ].
Do you wish to continue ( Y/N ) [N]: Y

[ INFO ]: Processing Level One Information.
[ INFO ]: Processing text-based configuration file section.
[ INFO ]: Processing GUI configuration section.
[ INFO ]: Processing SBR package and patch information.
[ INFO ]: Processing SBR dictionary section.
[ INFO ]: Processing SBR jar and certificate section.
[ INFO ]: Processing SBR XML file section.
[ INFO ]: Processing radius process information.
[ INFO ]: Processing System information section.
[ INFO ]: Levell information packed in Levell.tar.gz file.
```

The following example displays a sample output of **SIR.sh** script executed with the **-d DestDir** and **-r RadiusCore** options.

```
$ ./SIR.sh -d /tmp/Storage/ -r /opt/JNPRsbr/radius/core.162408

[ INFO ]: All requested data will be stored in the directory [
/tmp/Storage//SBR_Information_Report/ ].
[ INFO ]: 8.7G available in the [ /tmp/Storage//SBR_Information_Report/ ] directory.
[ INFO ]: Configured SBR directory: [ /opt/JNPRsbr/radius/ ].
Do you wish to continue ( Y/N ) [N]: Y

[ INFO ]: Processing Level One Information.
[ INFO ]: Processing text-based configuration file section.
[ INFO ]: Processing GUI configuration section.
[ INFO ]: Processing SBR package and patch information.
[ INFO ]: Processing SBR dictionary section.
```

```
[ INFO ]: Processing SBR jar and certificate section.  
[ INFO ]: Processing SBR XML file section.  
[ INFO ]: Processing radius process information.  
[ INFO ]: Processing System information section.  
[ INFO ]: Processing Level Two Information.  
[ INFO ]: Processing Log File Section.  
[ INFO ]: Configured date : 01-10-2014.  
[ INFO ]: Processing core file section.  
[ INFO ]: Collecting radius core file.  
[ INFO ]: Level1 information packed in Level1.tar.gz file.  
[ INFO ]: Level2 information packed in Level2.tar.gz file.
```

APPENDIX F

Thread and Flood Control Mechanism

IN THIS SECTION

- [Thread Control Settings | 1003](#)
- [Flood Control Settings | 1003](#)
- [SNMP Trap | 1004](#)
- [Logging Information | 1004](#)

SBR Carrier uses thread pools to process many concurrent authentication, accounting, and proxy requests. Under certain circumstances, a thread pool may be flooded by incoming requests caused by failovers or clients coming online. SBR Carrier uses the flood control feature to queue the packets (for example, first-in-first-out) until threads are available to process them.

NOTE: The flood control feature does not guarantee that every received packet is processed. Even with the flood control feature properly configured, it is possible that RADIUS packets are still dropped.

This appendix contains the following sections:

Thread Control Settings

This section provides information about the thread control settings in the [Settings] section of the **radius.ini** file.

- Max-Auth-Threads = 1 to 100,000 (default value is 100)
- Max-Acct-Threads = 1 to 100,000 (default value is 200)
- Max-Proxy-Threads = 1 to 100,000 (default value is 100)

Each thread setting controls the maximum number of threads allocated to handle the system load. The authentication and accounting thread settings handle all authentication and accounting requests, while proxy threads are responsible for forwarding accounting requests to proxy realms configured with Block=0. In this case, a separate proxy thread is used to handle the forwarding of the request, while the original accounting thread delivers the response to the NAS.

- Auth-Receive-Realtime-Thread-Priority = <0-2,147,483,647> (default value is 2,147,483,647, not real time)
- Acct-Receive-Realtime-Thread-Priority = <0-2,147,483,647> (default value is 2,147,483,647, not real time)

Proxy thread priority is set by the RealTime configuration setting in the [Configuration] section of the **proxy.ini** file.

For **WorkerThreadStackSize** in the [Settings] section of the **radius.ini** file, the default value is 512K (524,288).

See the *Juniper Networks Steel-Belted Radius Carrier Reference Guide* for more information.

Flood Control Settings

When all the threads for a packet type are in use simultaneously and flood queues are enabled, packets received are placed in a queue to be processed on a best-effort basis, in a defined order.

This section provides information about the following flood control settings in the [Configuration] section of the **radius.ini** file:

- Set the Max-Auth-Floods, Max-Acct-Floods, and Max-Proxy-Floods, which is the maximum number of packets retained by the flood queue. The default is 10,000 times the number of threads.
- The value for Max-Auth-Threads-In-Flood, Max-Acct-Threads-In-Flood, and Max-Proxy-Threads-In-Flood by default is half the total threads for the type. The minimum value is 1 and the maximum value is the

number of threads configured. This is the maximum number of threads that are allowed to process packets in the flood queue, rather than taking new work.

- The value for Auth-Flood-Queue-Shape, Acct-Flood-Queue-Shape, and Proxy-Flood-Queue-Shape is FIFO, LIFO, or RAND. This is the order in which the queue is drained, as well as the order in which packets are dropped if the flood queue is filled.
 - FIFO (First-In-First-Out)–Drops the new packet if the queue is full. This is recommended for most instances.
 - LIFO (Last-In-First-Out)–Drops the oldest packet in the queue when the queue is full, always puts the new packet at the start of the queue, and always gets the last packet in the queue first. This is recommended when you know you have to throw packets away, in certain cases, to keep up with the workload.
 - RAND (Random-in-Random-Out)–If the queue is full, drops either the first packet in the queue or the packet being received in order to get an item of work out of the queue. Drops random packets in the queue while the queue is full. This is rarely recommended, but approximates random delays. In some cases of network flooding, it may be desirable to randomly distribute responses.

See the *Juniper Networks Steel-Belted Radius Carrier Reference Guide* for more information.

SNMP Trap

The FloodQueueOverflow warning trap is sent whenever a request is dropped due to threads not being available and the maximum size of the flood queue being reached. The trap can be diluted (sent only every n times a request is dropped) by adding FloodQueueOverflow = n to the [Dilution] section of **events.ini**, where n represents the number of packets that must be dropped before the trap is sent. The OID for this SNMP trap is defined in **fnkradtr2.mib**.

See the *Juniper Networks Steel-Belted Radius Carrier Reference Guide* for more information.

Logging Information

The thread pool and flood queue information is logged to the SBRC log. To see the flood queue in action, turn on the following flood info parameter in the [Status] setting of the **radius.ini** file:

[Status] section

Auth-Thread-Flood-Info

Acct-Thread-Flood-Info
Proxy-Thread-Flood-Info

See the [“Settings to Support the Proxy CoA/DM Functionality” on page 715](#) for more information.

Glossary

Numerics

3GPP	Third generation Partnership Project (GSM).
3GPP2	Third generation Partnership Project 2 (CDMA).
802.1X	IEEE standard 802.1X. Standard for Local and Metropolitan Area Networks-Port-Based Network Access Control. Defines a mechanism that allows a supplicant (client) to connect to a wireless access point or wired switch so that the supplicant can provide authentication credentials that can be verified by an authentication server.

A

AAA	Authentication, Authorization, and Accounting.
AC	Access Controller.
accounting	The process of recording and aggregating resource use statistics and log files for a user, connection session, or function for billing, system diagnosis, and usage planning.
ACL	Access Control List.
agent	SNMP module on a managed device that responds to requests from a management station and sends traps to one or more recipients (trap sinks) to inform administrators of potential problems.
AKA	Authentication and Key Agreement. An extension to the EAP protocol that enables authentication and session key distribution using a mechanism based on symmetric keys and usually runs on a USIM.
AP	Access Point.
APN	Access Point Name.
attribute	RADIUS attributes that carry specific authentication, authorization, and accounting messages.
AuC	Authentication Center. The network element that provides the triplets for authenticating the subscriber.
authentication	The process of verifying the identity of a device and its user. This process is accomplished through transmission of identifying data at the time of connection.
Authentication and Key Agreement	See AKA.

authentication server	A back-end server that verifies, from the credentials provided by an access client, whether the access client is authorized to use network resources.
authorization	The process of controlling the access settings, such as privileges and time limits, that the user can exercise on a protected network.
autonomous server	A Steel-Belted Radius Carrier server that does not use centralized configuration management.
AVP	Attribute Value Pair. An attribute and its corresponding value; for example, User-Name = admin .

B

balun	Balanced/unbalanced converter. A device used to match impedance between balanced and unbalanced lines, usually twisted-pair and coaxial cable.
BAOC	Barring of All Outgoing Calls.
blocklist	A profile of checklist attributes that cause Steel-Belted Radius Carrier to reject an authentication request. For example, a blocklist profile might specify calling station phone numbers or IP addresses that are blocked by Steel-Belted Radius Carrier.
BS	Base Station.

C

CA	Certificate Authority. A trusted entity that registers the digital identity of a site or individual and issues a digital certificate that guarantees the binding between the identity and the data items in a certificate.
CCB	Customer Care and Billing system.
CCM	Centralized Configuration Management. The process by which configuration information is shared between a primary RADIUS server and one or more replica RADIUS servers so that all machines operate in a similar way.
CDF	Charging Data Function.
CDR	Call Detail Record. Call transaction record created by an MSC to track the network resources used by subscribers in making and receiving calls, so that billing systems can compute charges based on usage.
certificate	A digital file signed by a CA that guarantees the binding between an identity and the contents of the certificate.
CG	Charging Gateway. Device that collects, validates, and consolidates CDRs from other network components for processing by the network billing system.
Change of Authorization	See CoA.

CHAP	Challenge Handshake Authentication Protocol. An authentication protocol where a server sends a challenge to a requestor after a link has been established. The requestor responds with a value obtained by executing a hash function. The server verifies the response by calculating its own hash value. If the two hash values match, the authentication is acknowledged.
checklist	A list of attributes that must accompany a request for connection before the connection request can be authenticated.
CoA	Change of Authorization. Refers to RADIUS Change of Authorization, which is the dynamic change of the state of a previously authorized session by use of a RADIUS request sent towards the access equipment.
community	A group of devices and management stations running SNMP. An SNMP device or agent may belong to more than one SNMP community.
community string	Character string included in SNMP messages to identify valid sources for SNMP requests and to limit access to authorized devices. The read community string allows an SNMP management station to issue Get and GetNext messages. The write community string allows an SNMP management station to issue Set messages.
credentials	Data that is verified when presented to an authenticator, such as a password or a digital certificate.
CRL	Certificate Revocation List. A data structure that identifies the digital certificates that have been invalidated by the certificates' issuing CA before their expiration date.
CSCF	Call Session Control Function.
D	
daemon	A program on a UNIX or Linux host that runs continuously to handle service requests.
DHCP	Dynamic Host Configuration Protocol. Protocol by which a server automatically assigns (leases) a network address and other configuration settings to a client temporarily or permanently.
dictionary	Text file that maps the attribute/value pairs supported by third-party RADIUS vendors.
Disconnect Message	See DM.
DM	Disconnect Message. Refers to RADIUS Disconnect, which is the dynamic termination of a previously authorized session by use of a RADIUS request sent towards the access equipment.
DNIS	Dialed Number Identification Service. A telephone service that identifies what number was dialed by a caller.

DNS Domain Name Service. Internet protocol for mapping hostnames, domain names, and aliases to IP addresses.

E

EAP Extensible Authentication Protocol. An industry-standard authentication protocol for network access that acts as a transport for multiple authentication methods or types. Defined by RFC 2284. The base protocol used for a variety of authentication methods with Radius and 802.1X.

EAP-AKA EAP method that allows authentication with a mobile subscriber USIM card.

EAP-SIM EAP method that allows authentication with a mobile subscriber SIM card.

EAP-TLS Authentication method that uses EAP (Extensible Authentication Protocol) and TLS (Transport Layer Security).

EAP-TTLS Authentication method that uses EAP (Extensible Authentication Protocol) and TTLS (Tunneled Transport Layer Security).

Extensible Authentication Protocol *See* EAP.

F

FMC Fixed/Mobile Convergence.

FQDN Fully Qualified Domain Name.

FTP File Transfer Protocol.

function (Specific to IMS) Any one of the identified (and named) separable components of the IMS, which communicates with other functions exclusively using reference points.

G

General Packet Radio Service *See* GPRS.

GGSN Gateway GPRS Support Node.

Global System for Mobile Communications *See* GSM.

GPRS General Packet Radio Service. Packet-based wireless communication service for wireless phones and mobile computer users.

GSM Global System for Mobile Communications. A mobile telephone system that uses a SIM for subscriber identification.

GUI Graphical User Interface.

H

HA Home Agent. Maintains connection information about the mobile station (MS) and manages a persistent IP connection on the network for the MS. (In the SBR/HA 5.5 release, HA meant “High Availability,” but that term has been deprecated in favor of Session State Register, or SSR.)

HAAA Home Authentication, Authorization and Accounting server. AAA server on the subscribers home network.

HLR Home Location Register. Contains the primary subscriber database in a GSM network using SIM or USIM credentials.

home agent *See* HA.

Home PLMN *See* HPLMN.

Home Subscriber Server *See* HSS.

Home WLAN A WLAN that interworks with the HPLMN without using a VPLMN.

hotspot A WLAN Access Point offering network connectivity to the public.

HPLMN Home Public Land Mobile Network. The mobile network that has a billing relationship with the mobile subscriber, and usually the one that authenticates the user and authorizes access.

HSS Home Subscriber Server. The IMS function that contains the primary subscriber database in IMS networks that satisfy Release 6 of the IMS reference (IMS R6).

I

identity protection Prevention of an eavesdropper from discovering the identity of a user being authenticated.

IMS IP Multimedia Subsystem. An IP multimedia and telephony core network that is defined by 3GPP and 3GPP2 standards and organizations based on IETF Internet protocols. IMS is access independent as it supports IP to IP sessions over wireline IP, 802.11, and 802.15 packet data along with GSM/EDGE/UMTS and other packet data applications. IMS is a standard reference architecture that consists of session control, connection control, and an applications services framework along with subscriber and services data.

IMSI International Mobile Subscriber Identity. A unique subscriber identifier consisting of a three-digit Mobile Country Code (MCC), a two- or three-digit Mobile Network Code (MNC), and 10-digits-or-fewer Mobile Subscriber Identification Number (MSIN).

**International Mobile
Subscriber Identity** *See* IMSI.

IP	Internet Protocol.
IP Multimedia Subsystem	<i>See</i> IMS.
IPv4	Implementation of the TCP/IP suite that uses a 32-bit addressing structure.
IPv6	Implementation of the TCP/IP suite that uses a 128-bit addressing structure.
ISP	Internet Service Provider.
J	
JavaScript	Programming language designed for use in distributed environments such as the Internet.
JDBC	Java Database Connectivity. Application programming interface for accessing a database from programs written in Java.
L	
LCI	LDAP configuration interface.
LDAP	Light-weight directory access protocol. An IETF standard protocol for updating and searching directories over TCP/IP networks.
LDIF	LDAP Data Interchange Format. The format used to represent directory server entries in text form.
M	
managed device	A device that runs an SNMP agent.
management station	Host that monitors and controls managed devices running SNMP agents.
MAP	Mobile Access Part. The SS7 protocol standard that addresses registration of roaming users and the intersystem handoff procedure in wireless mobile telephony.
MCC	Mobile Country Code. The MCC, together with the MNC, uniquely identify an operator and help identify the authentication center from which subscriber information should be retrieved.
MGW	Media Gateway.
MIB	Management Information Base. A database of objects, such as alarm status or statistics counters, that can be monitored or overwritten by an SNMP management station.
MNC	Mobile Network Code. The MNC, together with the MCC, uniquely identify an operator and help identify the authentication center from which to retrieve subscriber information.
Mobile Application Part	<i>See</i> MAP.

Mobile Country Code	<i>See</i> MCC.
Mobile Network Code	<i>See</i> MNC.
Mobile Services Switching Center	<i>See</i> MSSC.
Mobile Station	<i>See</i> MS.
Mobile Subscriber ISDN	<i>See</i> MSISDN.
MPPE	Microsoft Point-to-Point Encryption. A means of representing point-to-point packets in an RC4 encrypted format. Defined in RFC 3078.
MS	Mobile Station. Device used to attach to a mobile network.
MS-CHAP	Microsoft CHAP. Proprietary version of CHAP.
MSC	Mobile Services Switching Center. Responsible for connecting calls together by switching packets from one network path to another. MSCs also provide information to support mobile service subscribers, including user registration, authentication, and location updating.
MSISDN	Mobile Subscriber ISDN. Telephone number of the mobile user, which conforms to the dialed number formats in the subscriber's country.
MTP	Message Transfer Part.
N	
NAD	Network Access Device. Network device that accepts connection requests from remote users, authenticates users via RADIUS, and routes users onto the network.
NAI	Network Access Identifier.
NAT	Network Address Translation.
native user	A user authenticated by Steel-Belted Radius Carrier using its internal authentication database.
network element	An addressable node or cluster of nodes in an IMS network, which may host any number of IMS functions.
NGN	Next Generation Network.
NIC	Network Interface Card.

node	A node is a logical element of a Session State Register cluster, which includes SBR Carrier nodes, management nodes, and data nodes.
nonce	Random value included in data exchanges to guarantee uniqueness and protect against replay attacks.
NSP	Network Service Provider.
numbering plan	Interpretation of the digits of an IMSI.
O	
ODB	Operator-Determined Barring. An HLR authorization of service designation that specifies that a subscriber is barred from service.
offline charging	Mechanism for collecting and forwarding charging information concerning I-WLAN and core network resource usage without affecting the service rendered in real-time.
P	
PAP	Password Authentication Protocol. An authentication protocol where a requestor sends an identifier and password to a server after a link has been established. If the identifier and password match an entry in the server's database, the authentication is acknowledged.
PDA	Personal Digital Assistant.
PDSN	Packet Data Serving Node. The attachment point between the RADIUS network and the IP network. May also be known as the foreign agent (FA) when Mobile IP is used.
PEAP	Protected Extensible Authentication Protocol. A two-phase authentication protocol where (1) an authentication server is authenticated to a supplicant using a digital certificate and a secure channel is established; and (2) the supplicant is authenticated to the authentication server via the secure channel.
permanent identity	The permanent identifier of a peer, including an NAI realm portion in environments where a realm is used. The permanent identity is usually based on the IMSI. Used on full authentication only.
PLMN	Public Land Mobile Network. Refers to a mobile network.
point code	The unique identifier for each node in an SS7 network.
PPP	Point-to-Point Protocol. Network protocol defined in RFC 1661 that provides a standard method for transporting multiprotocol datagrams over point-to-point links.
provisioning	A process, possibly requiring multiple steps, that enables customers to obtain services.

proxy RADIUS Process of authenticating users whose profiles are on other RADIUS servers by forwarding access-request packets received from a RADIUS client to a remote RADIUS server (the proxy target), and then forwarding the response from the remote server back to the RADIUS client.

proxy target The remote RADIUS server that actually performs authentication in a proxy RADIUS sequence.

pseudonym identity A pseudonym identifier of a peer, including a NAI realm portion in environments where a realm is used. Used on full authentication only.

**Public Land Mobile
Network** See PLMN.

Q

quintets The authentication data formed by the UMTS values: RAND (random number), XRES (expected response), CK (cipher key), IK (integrity key), and AUTN (authentication).

R

RADIUS Remote Authentication Dial-In User Service. A client/server security administration standard that functions as an information clearinghouse, storing authentication information about users and administering multiple security systems across complex networks.

reauthentication identity The reauthentication identifier for a peer, including a NAI realm portion in environments where a realm is used. Used on reauthentication only.

**Remote Access Dial-In
User Service** See RADIUS.

return list A list of attributes that Steel-Belted Radius Carrier must return to a RADIUS client after authentication of a user succeeds. The return list usually provides additional parameters that the RADIUS client needs to complete the connection.

roaming The ability to move from one Access Point coverage area to another without interruption of service or loss of connectivity.

S

SBC Session Border Controller.

SBR Steel-Belted Radius, the product family that includes Steel-Belted Radius Carrier.

SCTP Stream Control Transmission Protocol. An Internet Protocol used by the SIGTRAN protocol stack to transport SS7 signaling commands. See IETF RFC 4166.

server In a Session State Register cluster, a computer that hosts one or more nodes.

service authorization Authorization allowing a subscriber to access the requested service based on subscription.

session ID	Session Identifier. A string of characters uniquely identifying the session.
Session State Register	<i>See</i> SSR.
SHA-1	Secure Hash Algorithm-1. A one-way cryptographic function that takes a message of any length and produces a 160-bit message digest.
Signaling System 7	<i>See</i> SS7.
Signalware	The Mavenir SIGTRAN protocol stack provided with Steel-Belted Radius Carrier.
SIGTRAN	Protocol stack supporting SS7 signaling using the SCTP Internet Protocol. <i>See</i> IETF RFC 4166.
silent discard	The process of discarding a packet without further processing and without notification to the sender.
SIM	Subscriber Identity Module.
SIM card	A SIM-based hardware SmartCard that contains the authentication keys for a GSM mobile telephone subscriber.
SIP	Session Initiation Protocol.
SmartCard	A small card containing a computer chip that can store information, including authentication information and algorithms.
SNMP	Simple Network Management Protocol.
SS7	Signaling System 7. The network and protocols used to provide out-of-band signaling (control) for telephone services to support call establishment, billing, routing, and information exchange for the public switched telephone network.
SSID	Service Set Identifier.
SSL	Secure Sockets Layer. Program layer that manages the security of messages on a network.
SSR	Session State Register, an optional module for Steel-Belted Radius Carrier that implements a multi-computer cluster to support shared databases that multiple SBR Carrier servers can access to ensure that a single set of data is used for all transactions and to implement a high-availability environment.
STP	Signaling Transfer Point.
supplicant	The client in an 802.1X-authenticated network.

T

TISPAN	Telecoms & Internet converged Services & Protocols for Advanced Networks (standardization body of ETSI).
TLS	Transport Layer Security.
TLV	Type-Length-Value. A synonym for AVP; named because the raw encoding of such a value is a type field (for example, 1 for User-Name) followed by a length value (for example, 6) followed by the value of the attribute (for example, test).
trap	An SNMP message that reports a significant event, such as a problem, error, or change in state, that occurred within a managed device.
trap sink	The destination for trap messages sent by an SNMP agent on a managed device.
TS	Teleservice. HLR authorization of service designation.
TTLS	Tunneled Transport Layer Security.

U

UE	User Equipment.
UICC	Universal Integrated Circuit Card. The chip card used in mobile terminals in GSM and UMTS networks. The UICC ensures the integrity and security of all kinds of personal data, and typically holds a few hundred kilobytes.
UMA	Unlicensed Mobile Access.
UMTS	Universal Mobile Telecommunications System. Type of mobile network (next generation after GSM) that uses the USIM card for authentication.
Universal Mobile Telecommunications System	<i>See</i> UMTS.
user database	A database where a RADIUS server keeps information about users, such as authentication information and network access permissions.
user identifier	Identifier of a user that may be used, for example, in charging functionality for billing purposes.
user profile	A record in the user database that describes how to configure a particular user or class of users during authentication and authorization.
USIM	UMTS Subscriber Identity Module.

USIM card A SIM-based hardware SmartCard that contains the authentication keys for a 3G mobile telephone subscriber.

V

VAAA Visited Authentication, Authorization and Accounting server. AAA server on the visited access network, responsible for routing authentication and accounting requests to home network.

Visited PLMN See VPLMN.

VLAN Virtual Local Area Network.

VLR Visitors Location Register.

VoIP Voice over IP.

VPLMN Visited Public Land Mobile Network. The mobile network that is providing connectivity to a roaming user.

VPN Virtual Private Network.

VSA Vendor-Specific Attributes. Usually refers to a vendor-specific attribute *and* its associated value. VSA may be used to indicate a vendor-specific attribute or vendor-specific AVP. In RADIUS, VSAs are special attributes that contain an IANA-assigned enterprise code followed by TLVs (Type Length Value) that can be defined by the vendor who owns the enterprise code. As a result, vendors can define their own RADIUS VSAs without fear of colliding with another vendor's VSA assignments.

W

W-CDMA Wideband Code Division Multiple Access.

W-CDR Wireless LAN type of CDR.

WEP Wired Equivalent Privacy. An encryption method designed to encrypt traffic between a WLAN client and an access point.

Wi-Fi Wireless local area network that uses the IEEE 802.11a, b, or g radio protocols.

WiMAX Worldwide Interoperability for Microwave Access.

WISP Wireless Internet Service Provider.

WLAN Wireless Local Area Network.