

# Juniper Networks® Steel-Belted Radius® Carrier

---

## Performance, Planning, and Tuning Guide

Published  
2020-11-19

Release  
8.6.0

Juniper Networks, Inc.  
 1133 Innovation Way  
 Sunnyvale, California 94089  
 USA  
 408-745-2000  
[www.juniper.net](http://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](mailto: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](mailto: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](mailto: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](mailto: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](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 "inif" 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.]

*Juniper Networks® Steel-Belted Radius® Carrier Performance, Planning, and Tuning Guide*

Release 8.6.0

Copyright © 2019 Juniper Networks, Inc. All rights reserved.

#### Revision History

August 2019—Revision 1

November 2020—Revision 2

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

## 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.



# Table of Contents

## About This Guide | xxxviii

Objective | xxxviii

Audience | xxxviii

Documentation Conventions | xxxix

Related Documentation | xl

Obtaining Documentation | xlvii

Documentation Feedback | xlvii

Requesting Technical Support | xlviii

1

## Performance, Planning, and Tuning

### Overview of Performance, Planning, and Tuning of Steel-Belted Radius Carrier | 2

Steel-Belted Radius Carrier Overview | 2

Document Organization | 3

Performance Theory | 4

Performance Theory: Threads | 5

Thread Models | 5

Thread Priority | 6

Locking Threads to a Processor | 7

Performance Theory on Contention | 7

Managing Contention | 8

CPU Utilization | 9

Case 1: More CPU versus More GHz per CPU | 10

Case 2: Virtual CPU versus Physical Cores | 10

Performance Theory on Memory Data Structures | 11

### Understanding Performance Metrics | 13

Metrics and Definitions | 13

Components of a Sample Use Case | 15

Average Peak Rates | 19

Spike Rates at Peak | 20

**Utilities to Measure the Performance of an Existing System | 20**

- prstat | 21
- sar | 24
- iostat | 25
- vmstat | 26
- prtdiag and sosreport | 26
- psrinfo | 28
- psradm | 29
- cpuinfo on Linux | 29
- Taskset on Linux | 31
- Disabling a CPU on Linux | 31
- System Statistics | 32

**Performance Planning | 36****Understanding SBRC Call Models | 36****Understanding the Memory Utilization Levels | 38**

- Memory Utilization on Standalone | 38
- Memory Utilization on SSR Cluster | 39

**Understanding the CPU Requirements | 42**

- Standalone or Front-End Applications | 46
- Back Ends | 46

**Retries | 47****Load Balancing on RADIUS Front-End Applications and Downstream Proxy Devices | 48**

- Stateful and RADIUS-Aware | 48
- Network Configuration for SSR | 49
- Why Latency Kills Performance: D-D and S-D | 49
- Subscriber Scaling | 49
- Why Are Interims Hard, and When Can They Be Thrown Away? | 50

**Handling Downstream Latency and Traffic Spikes | 50**

- Thread Control | 51
- Flood Control | 51
- Settings | 51

## **Understanding the SBRC Network Architecture | 53**

SBRC Architecture Overview | 54

Understanding “Geodiversity” Between the SSR D Nodes | 55

NDB Cluster Timeout Dependencies | 56

High Throughput Configurations | 58

Risks to Uptime within the Architecture | 58

## **Debugging Performance and Functionality Problems | 60**

Determining the Effective Performance and Peak Capacity | 60

Calculating Performance Scaling on an Existing System | 62

Debugging Performance Issues | 64

CPU Bound | 64

I/O Bound | 66

Contended Systems | 68

Notes on Debugging Performance Issues | 70

## **Special Topics in Performance | 71**

Multifactor Authentication over EAP | 71

EAP-TLS, TTLS, and WiMAX | 72

Reauthentication Cache | 72

Cipher Suites and Key Sizing | 73

Cipher Suite Scaling | 74

Reducing the Number of Round Trips | 74

Certificate Management | 74

Multiple TLS/TTLS-Only Front-End Applications | 75

EAP-SIM/AKA and authGateway | 75

External Authentication and Accounting | 76

JavaScript | 77

Proxy Radius | 77

Smart Static Accounting | 78

Spoiled Accounting | 78

IP Pools through the SSR | 79

Accounting-Off or Accounting-On with AutoStop Enabled | 79

## **SBRC System-of-Systems Performance Reference | 81**

Hardware Used | 81

CPU Utilization | 82

Test Results Summary | 82

Pap Authentication, Accounting Start, Accounting Stop | 83

Pap Authentication, Accounting Start, Accounting Stop—Sessions Preloaded | 84

Accounting Only—Sessions Preloaded, One Start and One Stop, Four D Nodes | 84

Accounting Only—Sessions Preloaded, One Start and One Stop, Two D Nodes | 84

Standalone: Auth/start/stop CPS | 85

Standalone: Accounting for 200 Threads | 85

Standalone: Authentication Only | 86

LDAP Authentication Only | 86

Oracle 11g | 87

LCI Queries against the SSR through SBRC | 87

IP Address Allocation | 88

IP Address Allocation with Eight Threads, Two D Nodes | 88

IP Address Allocation with Eight Threads, Four D Nodes | 89

TTLS | 89

TTLS Plus Storing Resumption Context | 90

WiMAX | 91

4D node system: M5000 | 91

Standalone: M9000 | 91

Standalone: T3 | 92

SSR and Standalone Performance: E4870 and X5687 CPUs | 93

## **Performance Metrics for 64-Bit Standalone SBR Carrier | 95**

Components of a Sample Use Case | 95

Test Results Summary of Standalone SBR Carrier | 96

Native User Authentication and Logging | 97

LDAP Authentication and Thread Configuration | 98

SQL Authentication Only | 99

Native Accounting Start Only | 99

Native User Authentication, Accounting Start, Accounting Stop | 100

LDAP Authentication, Accounting Start, Accounting Stop | 100

SQL Authentication, Accounting Start, Accounting Stop | 101

Test Result Summary of Standalone Running on KVM Hypervisor | 101

KVM Hypervisor Running with One VM and Varying CPU Counts | 102

KVM Hypervisor Running with Six VMs | 103

Accounting Start Only for Proxy | 105

# About This Guide

## IN THIS SECTION

- Objective | xxxviii
- Audience | xxxviii
- Documentation Conventions | xxxix
- Related Documentation | xl
- Obtaining Documentation | xlvii
- Documentation Feedback | xlvii
- Requesting Technical Support | xlviii

This preface provides the following guidelines for using the *Steel-Belted Radius Carrier Performance, Planning, and Tuning Guide*:

## Objective

This guide describes how to plan and tune the performance of Steel-Belted Radius Carrier running on the Solaris and Red Hat Enterprise Linux operating systems.

It provides tips, use cases, and tools you need to:

- 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

## Audience

This guide is intended for users who are network or system architects and managers planning a new, or upgrading an existing, SBR Carrier Installation. It also provides guidance to system engineers to better troubleshoot performance issues.

## Documentation Conventions

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

Table 1: Notice Icons

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

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

Table 2: Text and Syntax Conventions

Convention	Description	Examples
<b>Text Conventions</b>		
<b>Bold text like this</b>	Represents commands and keywords in text.	<ul style="list-style-type: none"> <li>Issue the <b>clock source</b> command.</li> <li>Specify the keyword <b>exp-msg</b>.</li> </ul>
<b>Bold text like this</b>	Represents text that the user must type.	<b>host1(config)#traffic class low-loss1</b>
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"> <li>Emphasizes words.</li> <li>Identifies variables.</li> <li>Identifies chapter, appendix, and book names.</li> </ul>	<ul style="list-style-type: none"> <li>There are two levels of access, <i>user</i> and <i>privileged</i>.</li> <li><i>clusterId</i>, <i>ipAddress</i>.</li> <li><i>Appendix A, System Specifications</i>.</li> </ul>

Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
Plus sign (+) linking key names	Indicates that you must press two or more keys simultaneously.	Press Ctrl+b.
<b>Syntax Conventions</b>		
Plain text like this	Represents keywords.	terminal length
<i>Italic text like this</i>	Represents variables.	<i>mask, 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 [xli](#) 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 Release Notes</i>	Contains the latest information about features, changes, known problems, and resolved problems.
<i>Steel-Belted Radius Carrier Performance, Planning, and Tuning Guide (This guide)</i>	Provides tips, use cases, and tools you need to: <ul style="list-style-type: none"> <li>• Improve SBRC performance through planning, analysis, and configuration</li> <li>• Increase SBRC throughput and reliability</li> <li>• 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</li> </ul>

**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 xli 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.

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

RFC Number	Title
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.
RFC 2719	<i>Framework Architecture for Signaling Transport</i> . L. Ong et al., October 1999

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

RFC Number	Title
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
RFC 3579	<i>RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP).</i> B. Aboba, P. Calhoun, September 2003.

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

RFC Number	Title
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. Levkowitz. 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.
RFC 4372	<i>Chargeable User Identity.</i> F. Adrangi and others. January 2006.

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

RFC Number	Title
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-TLSv0).</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 and 3GPP2 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 xlvii 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"> <li>• Section 5.1.7: Interworking between PLMN and WLANs</li> </ul>
3GPP TS 23.003 Version 12.6.0	<i>Numbering, addressing, and identification</i>	<ul style="list-style-type: none"> <li>• Section 2.2: Composition of IMSI</li> </ul>
3GPP TS 23.008 Version 12.6.0	<i>Organization of subscriber data</i>	<ul style="list-style-type: none"> <li>• Section 3B: Definition of subscriber data I-WLAN domain</li> </ul>
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"> <li>• Section 6.1: Reference Model</li> <li>• Section 6.2: Network Elements</li> </ul>
3GPP TS 23.402 Version 12.8.0	<i>Architecture enhancements for non-3GPP accesses</i>	<ul style="list-style-type: none"> <li>• Section 4.1: Concepts</li> <li>• Section 4.3: Network Elements</li> </ul>
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"> <li>• Section 6: UE – EPC Network protocols</li> <li>• Section 8: PDUs and parameters specific to the present document</li> </ul>
3GPP TS 29.002 Version 12.7.0	<i>Mobile Application Part (MAP) specification</i>	<ul style="list-style-type: none"> <li>• Section 6: Requirements concerning the use of SCCP and TC</li> <li>• Section 7.1: Terminology and definitions</li> <li>• Section 7.2: Modelling principles</li> <li>• Section 7.3: Common MAP service</li> </ul>
3GPP TS 29.273 Version 12.7.0	<i>Evolved Packet System (EPS); 3GPP EPS AAA interfaces</i>	<ul style="list-style-type: none"> <li>• Section 4: SWa Description</li> <li>• Section 5: STa Description</li> <li>• Section 6: SWd Description</li> <li>• Section 7: SWm Description</li> <li>• Section 8: SWx Description</li> <li>• Section 9: S6b and H2 Description</li> <li>• Section 10: Result-Code and Experimental-Result Values</li> </ul>

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"> <li>• <i>Section 6: Authentication and key agreement procedures</i></li> <li>• <i>Section 7: Establishment of security contexts in the target access system</i></li> <li>• <i>Section 8: Establishment of security between UE and ePDG</i></li> <li>• <i>Section 9: Security for IP based mobility signalling</i></li> <li>• <i>Section 14: Temporary identity management</i></li> </ul>

### 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](mailto: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 CSC Case Manager:  
<https://www.juniper.net/cm/>

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 Case Manager tool in the CSC at <https://www.juniper.net/cm/>
- 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

## Performance, Planning, and Tuning

---

Overview of Performance, Planning, and Tuning of Steel-Belted Radius Carrier | 2

Understanding Performance Metrics | 13

Performance Planning | 36

Understanding the SBRC Network Architecture | 53

Debugging Performance and Functionality Problems | 60

Special Topics in Performance | 71

SBRC System-of-Systems Performance Reference | 81

Performance Metrics for 64-Bit Standalone SBR Carrier | 95

---

# Overview of Performance, Planning, and Tuning of Steel-Belted Radius Carrier

## IN THIS CHAPTER

- [Steel-Belted Radius Carrier Overview | 2](#)
- [Document Organization | 3](#)
- [Performance Theory | 4](#)

This chapter provides an overview of Steel-Belted Radius Carrier, general performance concepts, and key use cases on CPU utilization. The following topics are included in this chapter:

## Steel-Belted Radius Carrier Overview

Steel-Belted Radius Carrier (SBRC) is an authentication, authorization, and accounting (AAA) solution that provides data services for wireline, wireless, and converged carriers. SBRC 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.

SBR Carrier has been qualified and is supported on Oracle Solaris 11.3.36.10.0 and 11.3.36.20.0 (SPARC) and Red Hat Enterprise Linux 7.3, 7.4, 7.5, 7.6, and 7.7 on Intel (Xeon) platforms.

**NOTE:** SBR Carrier 8.6.0 does not support RHEL 6.x, 7.0, 7.1, and 7.2.

The SBR standalone and cluster configurations are supported on the 64-bit version of Linux operating system.

SBR Carrier supports virtualization on Linux, VMware hypervisor, Kernel-based Virtual Machine (KVM) hypervisor, and logical domains on Solaris. SBR Carrier has been tested with VMware ESXi 5.1, 5.5, 6.0, and 6.5 versions and KVM hypervisor on a RHEL 7.3 machine.

**NOTE:** SSR cluster in virtualized environments is not officially supported. Juniper Networks may still provide support for known issues and for those where you can demonstrate the issue exists on the native OS.

The following are the basic operations of SBRC:

- Receives RADIUS requests (UDP packets) from multiple network access devices, or Radius clients.
- Routes RADIUS packets using the proxy functionality.
- Performs a wide array of internal processing.
- Authenticates users against its own or an extended array of supported back-end databases.
- Accounts to files or one or more back-end databases.
- Maintains the current session state either in memory in a standalone configuration, or in the Session State Register (powered by Oracle MySQL NDB) in a clustered configuration.

The performance of SBRC is a function of the hardware resources on which the software executes and the wide array of internal processing required to handle the work. Each additional level of processing done has an impact on the overall resources of a system. As the complexity of processing increases, the effective performance on a given amount of CPU and I/O decreases.

## Document Organization

The *Juniper Networks Steel-Belted Radius Carrier Performance, Planning, and Tuning Guide* is organized as shown in [Table 6 on page 3](#).

**Table 6: Document Organization**

Chapter	Description
Chapter 1, "Overview of Performance, Planning, and Tuning of Steel-Belted Radius Carrier" (This chapter)	Provides an overview for understanding SBRC general performance theory concepts.
Chapter 2, "Understanding Performance Metrics"	Provides information about key utilities that you can use to analyze the performance characteristics of an existing or planned system.
Chapter 3, "Performance Planning"	Provides information that helps you understand the effects of various call models on SBRC memory utilization levels, CPU requirements, load balancing on SBRC front-end applications, and downstream proxy devices.

Table 6: Document Organization (*continued*)

Chapter	Description
Chapter 4, “Understanding the SBRC Network Architecture”	Describes the highly reliable network architecture of SBRC. Explains existing error-handling techniques in SBRC that can improve the performance and reliability of the network.
Chapter 5, “Debugging Performance and Functionality Problems”	Provides information you can use to determine the scalability of an existing SBRC system and debug the system performance issues.
Chapter 6, “Special Topics in Performance”	Provides additional details about the performance of certain subsystems that affect SBRC.
Chapter 7, “SBRC System-of-Systems Performance Reference”	Provides performance results tested on a given system-of-systems to estimate the SBRC load.

## Performance Theory

The performance of a transaction processing system such as SBRC, and each component of a system, can be characterized by various metrics. Using these metrics, performance can be classified under the following areas of analysis:

- **Throughput**—Calculated based on the rate of system input and output. Often referred to as bandwidth, throughput is measured in work items per second.
- **Utilization**—Calculated based on the impact of a given throughput on the internal components of a system, which can be in terms of CPU utilization, I/O utilization, or memory utilization. These metrics are used for scalability analysis to determine which components of the system are the most impacted for a given workload.
- **Latency**—Measured as the difference in time between input and the corresponding output.

Throughput and latency are often linked to the amount of memory utilization (or in some cases, thread utilization) required to act as a buffer to store the state of the transaction waiting to be processed. For example, consider the following use case. A processor is idle for either 1 or 10 seconds before returning a result (such as when the user has entered an invalid password). The system does the same work in either case, but the result occurs either in 1 or 10 seconds after the request. The throughput and CPU utilization remain the same. The latency increases in the 10-second case. While the transaction is outstanding, the system must store the information about the remaining work that needs to be done when it becomes active, which results in an increase in the amount of memory used to buffer the information while the transaction is idle.

Throughput and utilization are the key metrics used for scalability analysis and planning. Latency does not play an important factor in analyzing the performance of SBRC itself, except in managing failure conditions at the RADIUS level. Latency, however, is a critical factor in analyzing the performance of the external systems and directly impacts memory utilization and thread utilization levels. SBRC is impacted by latency on external systems such as the network connecting to back-ends or between SSR nodes, latency of external authentication or accounting databases, or latency to proxy targets.

## Performance Theory: Threads

Before the advent of multitasking operating systems (OSs), a single CPU ran one program to handle one task. If the program required an input (disk I/O or user input), the CPU waited for the input and did not perform any other task.

With the advent of multitasking OSs, the concept of processes arose. Resources of a CPU are time-sliced so that multiple processes can run at approximately the same time. While one process is waiting for an input (such as disk or network I/O), another process performs another task. Each process is encapsulated, so that one process does not see the memory used by other processes without special system calls to share it.

In recent years, processes have been split into smaller units sharing the same memory space. The term used for this is “thread,” also known as lightweight process (LWP). Two threads run the same groups of instructions in their program (although they could be running in different places at the same time on different CPUs) within one process. Multiple threads are executed simultaneously and manipulate the same memory (which may give rise to memory contention. See [“Performance Theory on Contention” on page 7](#) for more information.)

### **Thread Models**

A thread can be thought of as managing a part of the total state of the system. Items managed by a given thread are grouped together with items managed by other threads. This means that passing items from one thread to another is quick and does not require inefficient cross-process calls, which would be required when crossing a process boundary.

For example, a thread with all its state makes an external request for information it needs (perhaps from the network) and goes idle. Another thread reads efficiently from the network and dispatches a task back to the requester thread.

Three thread models are typically used in SBRC.

- **Dispatcher**—One thread manages a resource and many threads wait for it to dispatch a request and return a result. This model is used, for example, in reading and writing to the network or the disk.
- **Buffering**—If task completion is of high priority and you are not concerned about the result of an operation before continuing to process the rest of the transaction, then that part of the task is placed in a queue (buffer) and the rest of the task can continue. This is how logging works: multiple threads write to that log buffer and another thread drains the log buffer in memory and writes it to the disk.

- **Worker thread**—Worker threads are managed by a manager. When a thread needs some other thread to do a particular task (for example, handling a new authentication request), the authentication worker thread manager assigns the task to a thread in an idle state, prompting the thread to execute the authentication code. The thread executes the authentication code while keeping the existing state of the transaction. If the manager cannot identify an idle thread, it creates a new thread and assigns the task. This model is often used in SBRC in a variety of cases.

The problem with too many threads is that only a certain number of tasks can be done on a system in a period of time. Starting new threads takes time and memory. Many threads means that a lot of state is being allocated. Too many threads (since each thread takes some amount of time to make) may lead to high memory and CPU overhead. This leads to memory contention, not enough CPU available to do the actual work, and the task may not be processed in time, which would not occur if the system had simply refused to start more threads and dropped the request, waiting for a retransmit.

The proper number of threads for an SBRC installation varies and has to do with the amount of external latency each thread needs to manage. In a lab environment, you can utilize an entire SBRC machine (to the limits of the CPU or disk speeds) with only 200 threads by doing normal user authentication and regular accounting. However, if you are using an external database that takes 200 ms to return, you might need 1000 threads or more to manage work for the latency of failover behavior.

### **Thread Priority**

Similar to other operating systems, both Linux and Solaris provide a method to set the relative priority. Relative priority is the amount and frequency of a time slice for executing individual threads. In certain cases, setting the priority improves performance. The overall value of the priority and the types of scheduler modes vary.

- On Solaris 10, there are 160 priorities (0 through 159). The two key ranges of thread priority are the SYS range (60 through 99), in which most OS threads run, including network and socket buffer handlers; and the RealTime (RT) range (100 through 159). In the SYS range, the amount of time taken for activation is effectively infinite until interrupted or idle. In the RT range, threads are scheduled more frequently and preempt SYS threads, but operate on a fixed-time quantum, although they typically work until they return explicitly to an idle state. It is better to have only a few RT threads on a multi-CPU system. Starving SYS threads of CPU time may cause your network or disk to stop processing traffic into buffers for use by the rest of the system.
- On Linux, the real-time range is from 0 through 100, with the non-real-time “nice” range from -20 through 19 mapping into the priority range from 100 through 139.

In SBRC, you can set three threads to RT priority (see [“Thread Control” on page 51](#)): the threads are the ones that can receive network buffers from sockets, populate a packet cache, and dispatch the requests to other threads (authentication thread, the accounting thread, and proxy thread). These can be configured in the [Configuration] section of the radius.ini or the Proxy.ini file, respectively. See the *Juniper Networks Steel-Belted Radius Carrier Reference Guide* for more information.

### ***Locking Threads to a Processor***

SBRC does not require that a thread be locked to a processor because it primarily depends on external sources of data, such as the network and disk I/O, and few threads are critical to the performance of the system. The rare processor drift will not impact the overall performance of SBRC.

However, for performance, ndbmttd does require locking threads to processors and this relates to SBRC with SSR configuration. This mitigates the latency during processing for very high memory bandwidth workloads, as only a few threads are doing the bulk of the work. This is controlled by the `LockExecuteThreadToCPU` directive of the `config.ini` file, stored on each M node.

An increase in performance should only be relevant when the CPU for one thread on the ndbmttd node is within 20 percent of the maximum available CPU usage. On an M3000 with half the virtual CPUs disabled, **prstat -L** will report a thread at 20 percent out of the 25 percent maximum (which represents a fourth of the total cores on the M3000 processor). Other architectures vary radically. For instance, on Linux, “Irix” mode in **top** is similar to **prstat**’s default mode. See [“prstat” on page 21](#) for more information about the **prstat** and **top** utilities.

### **Performance Theory on Contention**

On a uniprocessor system, you can analyze the system performance with only a few variables (CPU speed, network bandwidth, memory speed, and disk I/O bandwidth). You can look at the metrics of a running system (CPU utilization percentage, bits per second on the network, or bytes per second to disk) and have a reliable idea of how much extra capacity will result in an overall performance increase. For example, if you are not constrained by the network bandwidth, memory speed, and disk I/O bandwidth, then doubling the CPU speed will give you approximately twice as much throughput, until you reach other bandwidth limits.

A known mode of contention is seen in what is called the “von Neumann bottleneck” (in the von Neumann architecture that most modern systems utilize). The faster CPU and memory subsystems are separated by a hardware bus of limited data rates. This bottleneck is limited with the addition of CPU caches at different levels of the hardware architecture. This is one source of memory contention.

With the advent of multiprocessing machines (with multiple CPUs and multiple cores per CPU, and even multiple virtual CPUs per core), what were single-use resources (such as a memory location or waiting for the cache to fill from memory) can become contended in real time.

For example, two CPUs might want to operate on the same memory location at the same time. If both CPUs do this operation at the same time when the operation is flushed from the CPU cache back to memory, an update will occur from one CPU but be ignored from the other. This leads to two processors having a different view of the underlying data. The same holds true of two processes attempting to update the same disk location, or two database systems attempting to be in sync, both attempting to update the same row.

There are two ways to handle this at a system level. The first is for operations to go in order. This is usually done by locking or using compare and swap (CAS) instructions and retrying if the result changes between



the start and end of the CAS instruction. The second is to optimize into a given relative operation, such as a transaction that can be rolled back and re-executed, using a single-instruction locked increment or locked decrement that can be taken out of order and has a known result at the end. Most systems (including SBRC) use locking, as you usually want to execute several operations against a reliable amount of data, not just one.

A lock is an in-memory data structure that uses special instructions to tell the CPU to start the negotiation with other CPUs for locking, or to wait for a lock to be released. The lock instruction permits only one processor to execute a set of instructions, or operate against a single, consistent in-memory data structure at a time. Under the lock you read, calculate, and write the result, then release the lock so other processors can perform their tasks.

**NOTE:** A novel way of dealing with contention (widely used in the cloud computing paradigm) is eventual consistency. That is, for certain operations that do not need to be entirely accurate on a read of a single node, optimizations can permit a system to perform multiple inaccurate operations that will eventually produce an accurate result. SBRC does not presently rely on these semantics.

Consider a case of a very large SMP (symmetric multiprocessing) machine with 128 virtual CPUs and 1000 instructions to execute under lock, compared to 2000 instructions that do not need to be executed under the lock, in a loop. In order to finish, processor 1 completes 2000 unlocked instructions, then under lock it completes 1000 instructions and releases the lock. During this time, processor 2 completes the 2000 unlocked instructions, and then under lock completes 1000 instructions. Processors 3 through 128 also complete the 2000 unlocked instructions and are waiting for processor 2 to finish and release the lock so that processor 3 can complete the instructions. However, processors 4 through 128 and now processor 1 are also waiting for the lock. In effect, operations under the lock are executed as if there is only one CPU on the system. This is why faster CPUs are generally more preferable to more CPUs.

In most cases under lock, the processor completes a few instructions then releases the lock and spends most of its time performing unlocked work or locking other minimally contended resources. While one processor holds the lock, other processors can do some other task on other parts of the system, even if they are taking a different lock. However, even for small resources that are heavily used (for example, a global statistic that is incremented non-linearly but relatively often), you could run into contention. For long operations (for example, a locked read or write to a file system, or an update or insert of an SQL row by two SBRCs for managing IP addresses on a cluster), you could run into contention.

### ***Managing Contention***

In SBRC, there are many different ways to manage contention. Locking is limited, in most cases, to short duration operations. For large operations—for example, during a search of an internally indexed table—you can do a certain number of operations, release the lock and respond to the OS with the task completion status so the OS can schedule other tasks, and then reacquire the lock and continue with the longer operation. This permits other threads that have queued up to run with a much shorter delay.

In SBRC, with various use cases, you can run performance testing and attempt to optimize out the greatest sources of contention. One mitigation technique is caching. For profiles, SBRC can be configured to store an in-memory cache of profile information from the database since profiles are used more frequently than user information and there are relatively fewer profiles than users.

SBRC also uses sharding, or arranging large data structures so they can be split apart. The stripes of a RAID disk or NDB D nodes are examples of sharding. In the case of a four-D node system, two D nodes are mirrors of the other two for reliability, but the data is striped between the first two, such that every other row is reliably stored on one or the other D node (and mirrored to its pair). This means that when you are doing an insert, update, or get by the primary key, you are only involving one of the two D nodes. However, when a query is executed, all the D nodes are queried as to whether the row fitting the criteria exists and each D node must do some work to handle the query. Sharding makes some items much faster in a tradeoff that may increase the overall system performance.

**NOTE:** On both Solaris and Linux (Intel), the optimal thread count is defined based on the following cases:

- Case1—When threads are not waiting for an external network event, a thread count of two to three times (real, not virtual) the core count limits the amount of contention each thread experiences. External network event could be the regular Standalone CST and local accounting to file, or a native user authentication, or Max-Acct-Threads in proxy cases with block=0. Enabling the appropriate flood queues permits the SBR Carrier to respond to spikes in a timely manner.
- Case2—When the threads are waiting for an external network event (such as, authentication from or accounting to an external database, or any case that uses database accessors, SSR cases, and proxy cases with block=1), the number of threads can be much higher, taking into account downstream latency.
  - In a case of 10 milliseconds round-trip latency (inclusive of downstream processing), each thread will be able to handle a maximum of or slightly under 100 transactions a second. To handle 5,000 transactions per second, you would need certainly in excess of 50 threads, or probably closer to 100.
  - If the latency is 50 milliseconds in a proxy case, to handle 5,000 transactions a second, you need upwards of 1,000 threads.
  - In cases where there are excessive number of threads, decreasing the WorkerThreadStackSize might be required to avoid reaching the total memory utilization limit.

## CPU Utilization

Several cases are presented in the following sections.

**NOTE:** On Linux (Intel), the optimal CPU utilization is within multiple cores of one CPU (for example, the Intel Xeon E7-4870). The penalty for off-CPU thread execution is at least 12% of the throughput. For basic work such as native user authentication or local accounting to file, on a 10 core-count CPU, enabling the 11th core on a second CPU decreases the throughput by almost 40% owing to off-chip access. Enabling a 12th core returns to a 10% decrease from 10 core performance. Subsequently, enabling more cores decreases throughput mildly until adding more cores has no effect, but never returns to the maximum throughput at the 10 core per one socket rate.

In the TTLS/TLS and proxy cases, enabling two sockets worth of cores does give higher total throughput, but the increase is towards the addition of more sockets.

### ***Case 1: More CPU versus More GHz per CPU***

If you compare a 4 core (Quad Core) machine running at 3 GHz (for a total of 12 GHz) versus a machine with 12 cores each at 1 GHz (for a total of 12 GHz), you will generally see better performance on the higher GHz per CPU.

For example, consider the case for executing 1000 locked instructions. The machine running at 3 GHz executes these 1000 instructions three times faster than the machine running more CPUs at 1 GHz. The amount of time under lock will be one third, so another thread waiting for the resource will wait one-third of the time to do its locked work compared to the 1 GHz CPU.

### ***Case 2: Virtual CPU versus Physical Cores***

The virtual CPU mechanism (known as CoolThreads in the Oracle SPARC Solaris-based system, controlled by `psrinfo` and `psradm`; and Hyper-Threading in an Intel-based system), in the broadest terms, is a way of getting more actual work done while waiting (for example, for memory latency) in much the same way threads wait and store state for disk and I/O latency. The state of a thread, in terms of registers and localized memory, is kept in a Layer 1 cache on the physical CPU while it is waiting for the memory to respond with the values from a particular location. This means that the pipeline inside a physical core that does the actual processing is kept as full as possible, at a minimum (but not zero) overhead. Switching registers in and out of the Layer 1 cache also takes time and sometimes threads migrate to different virtual processors, which takes longer, when the OS decides that a particular processor is underutilized. Threads usually stay with one processor unless there is a good reason for the OS to move them.

**NOTE:** On Linux, the overhead for thread processing (which is accounted for by %SY time) is somewhat higher than for Solaris for a similar workload.

Unless you are constrained by memory lookups and cache misses, you will not be able to utilize the extra time well. SBRC in a highly multithreaded system, where much more time is spent waiting for disk and network resources, will see only slightly increased performance utilizing virtual CPUs for each core. When

handling high CPU utilization authentication types (for example, high-encryption cases such as EAP-T/TLS) with very limited use sets, you might not see a major improvement in performance (unless you get more total GHz in terms of cores).

In real-time transaction processing, especially in the ndbmtd threads, you might want to lock a particular thread to a physical CPU and not permit it to be swapped out for another thread to avoid the associated latency. In both cases, this mechanism provides some configuration to lock certain important threads to a virtual CPU.

In the case of ndbmtd, which is very real-time sensitive and optimized to use the CPU effectively, turning off the virtual CPUs gives a dramatic increase in performance as it decreases potential latency. Further locking the threads to certain CPUs gives an additional minor improvement in performance.

For NDB, you are presently limited to approximately eight cores (with no virtual CPUs), with the execute threads taking up two cores (such as the M3000 with quad core CPUs) or four cores (such as the M4000 with two quad core CPUs); additional cores beyond eight do not provide a benefit for NDB versions 7.1 and earlier. A future NDB version may provide additional threading capabilities.

**In general, more total GHz from your cores (with or without virtual processors enabled) is better, whether they come from many slow cores or fewer fast cores, until you reach a particular contention threshold for your use case, at which time faster is better.**

## Performance Theory on Memory Data Structures

The two most common underlying memory data structures are stack and heap.

- **Stack**—A stack is a continuous chunk of memory that is preallocated and utilized linearly. Every time a procedure call is made, more space is created on the stack to save the data from the registers and automatic storage. Recursive procedures, for example, are stack-intensive. Each time a recursive procedure is called, stack space grows. In general, you must preallocate enough stack space for your workload. You must also limit the amount of memory for the stack of any given thread, so it does not take up too much space, resulting in an out of memory error with many threads.

Stack memory does not leak. When you return from a procedure, the stack is unwound.

- **Heap**—A heap is a mixture of items allocated through operators in code such as malloc or new, or in JavaScript with any object creation. Allocating from heap takes time and can lead to contention, but is a reasonably efficient memory data structure.

On Solaris, libumem and libmtmalloc both do a good job at managing multiple threads, allocating and freeing information within the heap. On Linux, the default malloc offered by glibc is reasonably efficient, especially when the number of threads contended on allocation exceeds the number of CPUs.

But additional efficiencies might be available with specially crafted memory allocation schemes—for instance, Professor Berger’s HOARD memory allocator, tcmalloc, and the commercially available “Lockless, Inc.” memory allocators all show promise for various use cases.

Heap memory can leak. If you have allocated a piece of memory with malloc in C, or new in C++, and you lose the pointer to it by not storing it or never referencing it again, the memory remains blocked and will be unusable. The memory is erased only when the process is deleted.

Memory used by a process is rarely returned to the system until the process exits. After starting SBRC and running load against it, the SBRC process takes more memory for stack as the threads reach deeper into its call trees until a steady state is reached and the amount of memory allocated is sufficient to handle the load.

An increase in memory usage is often normal. For instance, as you add sessions and if you do not delete them with an accounting stop, you will use more memory to store the session information.

However, sometimes the memory continues to increase linearly, even on a steady state test, often by just a few bytes per transaction. Sometimes it is even less, increasing perhaps only on an error condition and these cases are more difficult to detect. This is an indication of a memory leak. In this case, you must contact JTAC and the engineering team for support.

The mechanisms of detecting these sorts of leaks are diverse and often difficult to implement. Enabling libumem “transaction” debug mode or the equivalent for other allocation schemes, followed by obtaining multiple cores over the course of a period of time, may be the only option in certain cases.

**NOTE:** Virtual and physical memory—In all SBRC servers and SSR nodes, the working set of SBRC and the ndbmtd should fit well within main memory and should not be at risk of being swapped out to virtual paging due to other activities on the system. Main memory sizes that are smaller than the working set size of any process are not supported.

# Understanding Performance Metrics

IN THIS CHAPTER

- Metrics and Definitions | 13
- Components of a Sample Use Case | 15
- Average Peak Rates | 19
- Spike Rates at Peak | 20
- Utilities to Measure the Performance of an Existing System | 20

This chapter provides information about key utilities that you can use to analyze the performance of an existing or planned system. It also provides information about some of the terminologies used throughout this guide.

## Metrics and Definitions

Table 7 on page 13 describes the key performance metrics of SBRC.

Table 7: Key Performance Metrics

Acronym/Term	Definition
Accts/s	<p>RADIUS accounting per second.</p> <p>In SBRC, starts, stops, and interims have approximately the same load requirements. In certain use cases, starts can be marginally faster (a new session when a phantom was not generated), interims marginally slower (requiring an update, retrieval of existing values, and re-writing the record), and stops may be marginally more work (such as managing IP addresses and concurrency).</p>
Auths/s	<p>RADIUS authentications per second.</p> <p>In general, auths/s refers to PAP authentication or CHAP authentication, which is the RADIUS Auth-Request.</p> <p>The performance characteristics of these auths are similar; the differences in simple authentication cryptographies are usually minor.</p>

Table 7: Key Performance Metrics (*continued*)

Acronym/Term	Definition
CPS	<p>Calls per second.</p> <p>This can be one RADIUS authentication, one accounting start, and one accounting stop.</p> <p>This metric corresponds to a model where the sessions are transient, rather than of long duration where re-authentication occurs with each connection.</p>
CSPS	<p>Call setups per second.</p> <p>This is one authentication and one accounting-start.</p> <p><b>NOTE:</b> CSPS is rarely used by Juniper Networks, but is commonly used in other sources of reference.</p> <p>Adding 33 percent to CSPS equalizes the CPS metric, since CSPS ignores stops, whether generated by NAS, or autostops generated from the session timeout.</p>
EAP	<p>Extensible Authentication Protocol.</p> <p>Most EAP-based authentication protocols require multiple round trips of RADIUS authentication requests and responses.</p> <p>The number of round trips is strongly influenced by the client and the number of protocols supported. The client can send negative acknowledgements (NAKs) for each authentication protocol until either the last authentication protocol or the server has attempted to serve all the protocols on the available list, after which the server should be configured to reject.</p> <p>If several types of EAP requests are supported, and the most likely ones are the preferred ones; this will under most circumstances decrease the number of transactions done at the negotiation stage.</p>
EAP-SIM and EAP-AKA	<p>Extensible Authentication Protocol method for GSM Subscriber Identity Module (EAP-SIM) and Extensible Authentication Protocol method for UMTS Authentication and Key Agreement (EAP-AKA).</p> <p>EAP-SIM and EAP-AKA require multiple round trips and less cryptography in the SBRC front-end application, some of which is pushed into the authGW and some to the Home Location Register (HLR).</p>
EAP-TLS/TTLS	<p>EAP-Transport Layer Security/Tunneled Transport Layer Security.</p> <p>In addition to multiple round trips, some of the packets that require much heavier cryptography to process can take many CPU cycles on each front-end application.</p>

Table 7: Key Performance Metrics (*continued*)

Acronym/Term	Definition
Proxy	The performance overhead of writing a request downstream and reading a response from downstream is similar to the equivalent auths/s or accts/s. This is in addition to the overhead of proxy processing of greater or lesser complexity.
SPS	<p>Sessions per second.</p> <p>This can be one authentication, one accounting start, three interim-accounting, and an accounting-stop.</p> <p>In certain use cases, the number of transactions can vary significantly, but the three interim-accounting transactions are in place in proportion to the authentications, starts, and stops for calls of average duration. In an average case, interims are sent to extend the session life three times in a varying period of time.</p> <p><b>NOTE:</b> CPS can also be referred to as SPS in other documents or in literature from other organizations.</p>
TPS	<p>Transactions per second.</p> <p>In SBRC, a transaction refers to a RADIUS input packet and its corresponding response packet.</p> <p>For example, a key TTLS transaction may take 100 CPU-milliseconds to execute. An interim accounting transaction might generate an ACK, but it might not perform an update, making it quicker. Thus, the “TPS” of a system is not a great metric for comparative performance.</p> <p>In general, TPS refers to any RADIUS transaction processing such as receiving a packet, turning the attribute-value pairs (AVPs) into an internal representation, doing minimal processing common to all transactions, turning the internal representation of AVPs back into RADIUS and calculating the authenticator, and sending simple responses.</p>
WiMAX	<p>Worldwide Interoperability for Microwave Access.</p> <p>WiMAX is similar to EAP-TLS/TTLS for authentication in specific cases. In WiMAX, the session resumption keys and certain AVPs are stored within the NDB and use the WiMAX accounting flow functionality. This adds to the overhead for processing the accounting starts, interims, and stops.</p>

## Components of a Sample Use Case

Each use case can be considered as a collective work done by the components of SBRC, in terms of CPU, I/O, network, and latency.



For example, take the following medium, complex, and heavy utilization use case of a roaming WiMAX TTLS authentication, against an Oracle back end with foreign agent (FA) keying, IP address assignment, and using WiMAX accounting flows and an Oracle back end for accounting data.

Table 8 on page 16 shows the breakdown of use case tasks done by the SBRC components.

**Table 8: Sample Use Case Tasks**

Task Execution Sequence	Number of Transactions	Subtasks	Description
EAP-TTLS authentication	5	<ol style="list-style-type: none"> <li>1. SQL transaction validates the username and the TTLS password against an Oracle database.</li> <li>2. Profile AVP is assigned for internal SBRC processing.</li> <li>3. NDB phantom session is created with WiMAX attributes set for keying material.</li> <li>4. NDB insert to the TtlsResumptionCache and TtlsResumptionAttrs attributes.</li> <li>5. NDB insert to the WimaxMobilitykeys attribute.</li> <li>6. NDB update to the IP address table to mark an IP address as being used.</li> <li>7. Generated response is returned.</li> </ol>	<p>RADIUS transaction performs EAP-TTLS authentication.</p> <p>This is mostly a function of CPU utilization on the front-end applications, some of which have high CPU utilization.</p>
Foreign agent/gateway RADIUS transaction	1	<ol style="list-style-type: none"> <li>1. Transaction read from the NDB WimaxMobilityKeys.</li> <li>2. Response is calculated.</li> </ol>	Foreign agent or gateway RADIUS transaction queries the phantom session to return the generated keying material.

Table 8: Sample Use Case Tasks (continued)

Task Execution Sequence	Number of Transactions	Subtasks	Description
WiMAX accounting start	1	<ol style="list-style-type: none"> <li>1. NDB update to the CurrentSessions table.</li> <li>2. Insert to an Oracle schema for accounting the flow for the billing identity.</li> <li>3. ACK sent on success.</li> </ol>	<p>WiMAX accounting starts with the attribute WiMAX-Beginning-Of-Session set as true to start the flow accounting session.</p>
Accounting start	1	<ol style="list-style-type: none"> <li>1. NDB update to the CurrentSessions table.</li> <li>2. Insert to the Oracle table for the session.</li> <li>3. ACK sent on success.</li> </ol>	Accounting start updates the CurrentSession to turn a phantom session to a live session.
WiMAX interims	3 (one per hour)	<ol style="list-style-type: none"> <li>1. NDB update to the CurrentSessions table.</li> <li>2. Insert to the Oracle table for the session.</li> <li>3. ACK sent on success.</li> </ol>	<p>WiMAX interims with the attribute WiMAX-Session-Continue set as true.</p> <p>This attribute updates both the CurrentSession, and the flow to which this session belongs.</p>
Accounting request	1	<ol style="list-style-type: none"> <li>1. NDB delete from the CurrentSessions table and WimaxMobilityKeys.</li> <li>2. Insert or update to the Oracle table for the session.</li> <li>3. Simple ACK on success.</li> </ol>	Accounting request to stop the current session.

Table 8: Sample Use Case Tasks (continued)

Task Execution Sequence	Number of Transactions	Subtasks	Description
EAP-TTLS re-authentication request/response	2 to 3	<ol style="list-style-type: none"> <li>1. Profile AVP assigned as internal SBRC processing.</li> <li>2. NDB phantom session created with WiMAX attributes set for keying material.</li> <li>3. NDB insert to the WimaxMobilitykeys for the key.</li> <li>4. NDB update to the IP address table to mark an IP address as being used.</li> <li>5. Generated response is returned.</li> </ol>	EAP-TTLS re-authentication request/response pairs read from the NDB TtlsResumptionCache and TtlsResumptionAttrs attributes. In addition, performs cryptography to validate re-authentication.
Foreign agent/gateway RADIUS transaction	1	<ol style="list-style-type: none"> <li>1. Read from the NDB WimaxMobilityKeys.</li> <li>2. Response is calculated.</li> </ol>	Foreign agent or gateway RADIUS transaction queries the phantom session to return the generated keying material.
Accounting request	1	<ol style="list-style-type: none"> <li>1. NDB update to the CurrentSessions table.</li> <li>2. Insert or update to the Oracle table for the flow.</li> <li>3. Simple ACK on success.</li> </ol>	Accounting request to start a new session within the flow.
WiMAX interims	3	<ol style="list-style-type: none"> <li>1. NDB update to the CurrentSessions table.</li> <li>2. Insert or update to the Oracle table for the session.</li> <li>3. Simple ACK on success.</li> </ol>	WiMAX interims with the attribute WiMAX-Session-Continue set as true update both the CurrentSession and the flow to which the session belongs (two NDB hits each).

Table 8: Sample Use Case Tasks (*continued*)

Task Execution Sequence	Number of Transactions	Subtasks	Description
Accounting request	1	<ol style="list-style-type: none"> <li>1. NDB delete from the CurrentSessions table and WimaxMobilityKeys.</li> <li>2. Insert or update to the Oracle table for the session.</li> <li>3. Simple ACK on success.</li> </ol>	Accounting request to stop the second session and delete the session.
WiMAX stop	1	<ol style="list-style-type: none"> <li>1. NDB delete from the CurrentSessions table and WimaxMobilityKeys.</li> <li>2. Insert or update to the Oracle table for the session.</li> <li>3. Simple ACK on success closing the flow.</li> </ol>	WiMAX stop to end the flow.

From the above use case, you can see the complexity of scaling a single SBRC to support multiple use cases simultaneously, although they are additive when broken down into components. When applying complex use cases such as 3G and fixed TLS with DHCP, you will notice that scaling becomes extremely difficult to plan for due to the latency of external servers. However, by breaking the case down into component operations, you can estimate well. You can turn the above use case into a six-hour session:

- Approximate Oracle transactions = 13
- NDB hits = 23
- RADIUS transactions (of which 8 decompose into 1 TTLS and 1 resumption) = 22

## Average Peak Rates

This section describes the average peak rates.

For example, with one million subscribers, 20 percent of them operate the call model (as described in [Table 8 on page 16](#)) during the daytime peak hours. This is up to 200,000 sessions over 6 hours, or 33,333 sessions per hour, or 556 sessions per minute, or 10 sessions per second. To manage this load on average, you need to do 130 Oracle transactions per second, 230 NDB hits per second, plus 10 TTLS and 10

resumptions per second, plus 140 accountings per second. However, the assumption of a constant task flow at an average rate is not entirely valid; you will generally see that the spikiness in call processing exceeds the low average figure (see [“Spike Rates at Peak” on page 20](#)).

## Spike Rates at Peak

This section describes the spike rates at peak.

For example, in a spike case, 20 percent of users attempt to begin the call model (as described in [Table 8 on page 16](#)) within 10 minutes from the start of day. Their focus is primarily on tasks 1 through 4, assuming that tasks 5 through 12 will even out over time as subscribers perform other operations.

- Oracle transactions = 3
- NDB hits = 7
- RADIUS transactions = 8, TTLS = 1, FA = 1, and accountings = 2
- 200,000 sessions over 10 minutes = 20,000 sessions per minute, or 167 sessions per second

This means you need to do 501 Oracle transactions, 1169 NDB hits, 167 TTLS, 167 FA, plus 334 RADIUS accountings per second.

This example is within the recommended base configuration guidelines for SBRC, but you can easily see that spike rates of shorter duration (for example, a GGSN reconnecting all its users), a larger subscriber base, or a higher than expected percentage of users attempting to connect at the same time (for example, call traffic generally spikes after notable public events) could easily push the rate even higher. Establishing a realistic call model and planning beforehand as to what limits you to consider the worst-case peak load for your use case will give you the flexibility you need when establishing and maintaining a reliable and performing solution.

## Utilities to Measure the Performance of an Existing System

This section describes the following utilities available in the Solaris 10 operating system used to measure and view the performance of an existing system:

- [prstat on page 21](#)
- [sar on page 24](#)
- [iostat on page 25](#)
- [vmstat on page 26](#)
- [prtdiag and sosreport on page 26](#)

- [psrinfo on page 28](#)
- [psradm on page 29](#)
- [System Statistics on page 32](#)

## prstat

The **prstat** or **top** utility shows the current active processes information on the system and reports statistics based on the selected output mode and sort order. The following is a sample output with no arguments for an SBRC system under load for reference:

```
PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
28193 root        248M  142M cpu0   42    0  0:23:26  94% radius_generic/285
27695 hadm       169M  129M sleep   59    0  0:00:17  0.1% mysqld/18
27609 hadm       131M   25M sleep   59    0  0:00:15  0.1% ndb_mgmd/10
28244 root       3888K 3248K cpu2   59    0  0:00:00  0.0% prstat/1
 1916 noaccess   166M   79M sleep   59    0  1:58:27  0.0% java/18
 1737 root       4968K 2160K sleep   59    0  0:28:51  0.0% nmbd/1
27947 root       6688K 4424K sleep   59    0  0:00:00  0.0% sshd/1
...
Total: 82 processes, 559 lwps, load averages: 12.43, 5.73, 2.29
```

The command on Linux is similar:

```
top - 11:40:40 up 5 days, 19:52,  1 user,  load average: 0.09, 0.04, 0.01
Tasks: 101 total,  1 running, 100 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3%us,  0.8%sy,  0.0%ni, 97.8%id,  1.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3923200k total, 1872928k used, 2050272k free,  147100k buffers
Swap:  6160376k total,      0k used,  6160376k free,  481212k cached
```

```
PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
2631 root       -5   0 1871m 899m 8088  S   1.7  23.5 135:24.36 ndbmttd
1858 root       20   0  727m  90m  12m  S   0.7   2.4 23:51.54 radius
1641 hadm       20   0  842m  60m 5652  S   0.3   1.6 24:30.98 mysqld
   1 root       20   0 19272 1456 1176  S   0.0   0.0  0:00.78 init
```

[Table 9 on page 22](#) describes the key fields of the **prstat** utility output.

Table 9: prstat Output Description

Field	Description
CPU	Shows the percentage of in-use CPU across the entire machine. The CPU time is regressed over the number of seconds, so this value does not show spike utilization, but is averaged over time.
SIZE	Shows the total working-set virtual memory size of the process (on some versions of Solaris, this reads as SWAP). In general, this should be less than the system memory size, and for the SBRC front-end application, as a 32-bit application, must be less than 4G.
RSS	The resident set size (RSS) shows the amount of memory that is paged in. Some paging out of SBRC is tolerable, but if this value is less than the working-set size, check for other parts of the system processes using too much memory.
NLWP	Number of light-weight processes (or threads).
TIME	Shows the amount of CPU time that the process has been executing since the start.
STATE	The information is not very useful in a multithreaded application in the thread roll-up display, but in <b>prstat -L</b> , it will show the state of the thread.
PRI	The information is not very useful in a multithreaded application in the thread roll-up display, but in <b>prstat -L</b> , it will show the thread priority. See <a href="#">“Thread Priority” on page 6</a> for more information.

The following is an output of **prstat -L** showing per-thread utilization:

PID	USERNAME	SIZE	RSS	STATE	PRI	NICE	TIME	CPU	PROCESS/LWPID
28193	root	252M	146M	sleep	51	0	0:01:20	2.2%	radius_generic/65
28193	root	252M	146M	sleep	53	0	0:01:21	2.0%	radius_generic/28
28193	root	252M	146M	sleep	53	0	0:01:00	1.3%	radius_generic/66
28193	root	252M	146M	sleep	54	0	0:00:14	0.8%	radius_generic/156
28193	root	252M	146M	sleep	58	0	0:00:19	0.8%	radius_generic/122
28193	root	252M	146M	cpu4	53	0	0:00:17	0.7%	radius_generic/167
28193	root	252M	146M	sleep	55	0	0:00:21	0.7%	radius_generic/93
...									

Size and RSS refer to the process and not to the specific thread. STATE, CPU and Time refer to the specific thread.

**NOTE:** The maximum CPU for a given thread is equal to 100 divided by the number of **virtual** CPUs active on the system. In this case, since SBRC runs on an M3000 with 8 virtual CPUs enabled, the maximum would be 12.5 percent. Since all CPU utilization is less than 12.5 percent and evenly distributed across multiple threads, you will get more than 90 percent of the total CPU on the system. In this case, SBRC works well under the load.

For more general usage information on the **prstat** utility, see the man page at <http://download.oracle.com/docs/cd/E19253-01/816-5166/6mbb1kqdk/index.html>.

On Linux, the **top** command accepts the uppercase “H” subcommand to switch between per-process (the default) and per-thread reporting:

```
top - 11:34:34 up 5 days, 19:46, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 239 total, 1 running, 238 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.5%us, 1.2%sy, 0.0%ni, 97.2%id, 1.0%wa, 0.0%hi, 0.2%si, 0.0%st
Mem: 3923200k total, 1873052k used, 2050148k free, 147100k buffers
Swap: 6160376k total, 0k used, 6160376k free, 481212k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2631	root	-5	0	1871m	899m	8088	S	1.3	23.5	104:18.06	ndbmttd
2636	root	-5	0	1871m	899m	8088	S	1.0	23.5	54:27.83	ndbmttd
1681	hadm	20	0	842m	60m	5652	S	0.7	1.6	18:51.67	mysqld
2637	root	-5	0	1871m	899m	8088	S	0.7	23.5	67:02.36	ndbmttd
1680	hadm	20	0	842m	60m	5652	S	0.3	1.6	18:12.45	mysqld
2624	root	20	0	727m	90m	12m	S	0.3	2.4	15:55.08	radius
2625	root	20	0	727m	90m	12m	S	0.3	2.4	16:52.83	radius
2634	root	-5	0	1871m	899m	8088	S	0.3	23.5	1:48.90	ndbmttd
2638	root	-5	0	1871m	899m	8088	S	0.3	23.5	41:18.35	ndbmttd
2639	root	-5	0	1871m	899m	8088	S	0.3	23.5	35:21.34	ndbmttd
2666	root	-5	0	1871m	899m	8088	S	0.3	23.5	2:37.54	ndbmttd

Unlike Solaris, the default %CPU on Linux totals to 100% times the number of processors. Thus, 100% utilization for a thread corresponds to one virtual CPU. It does not appear to be the case that 100% CPU utilization is realistic on a hyper threaded system. Further, in this calculation, %System time is not accounted for.

It is safe to assume, given that the highest CPU utilization threads for SBR are generally those involved with network I/O (which takes the bulk of the %SY time), that if the highest CPU utilization thread plus the overall %Sys time approximates to 100%, then that thread is at its maximum.



**sar**

This is a system activity reporter utility. This Solaris command shows the overall activity of the system. The following is a sample output:

```
# sar 1 10 [every 1 second, display the results 10 times]
```

```
SunOS sbr-perf2 5.10 Generic_141444-09 sun4u      01/10/2011

15:14:10      %usr      %sys      %wio      %idle
15:14:11          70          25          0          5
15:14:12          71          26          0          3
...
```

For more general usage information on the **sar** utility, see the man page at <http://download.oracle.com/docs/cd/E19253-01/816-5166/6mbb1kqgf/index.html>.

The following is a sample output:

```
# sar -A 1 10 [every 1 second display all the results 10 times]
```

```
15:13:06      %usr      %sys      %wio      %idle
              device          %busy  avque  r+w/s  blks/s  avwait  avserv
runq-sz %runocc swpq-sz %swpocc
bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrnt/s
swpin/s bswin/s swpot/s bswot/s pswch/s
scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
iget/s namei/s dirbk/s
rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
proc-sz  ov  inod-sz  ov  file-sz  ov  lock-sz
msg/s  sema/s
atch/s  pgin/s ppgin/s  pflt/s  vflt/s slock/s
pgout/s ppgout/s pgfree/s pgscan/s %ufs_ipf
freemem freeswap
sml_mem  alloc  fail  lg_mem  alloc  fail  ovsz_alloc  fail
...
15:13:08          72          26          0          2
              nfs1              0      0.0          0          0      0.0      0.0
              nfs2              0      0.0          0          0      0.0      0.0
              nfs3              0      0.0          0          0      0.0      0.0
              nfs8              0      0.0          0          0      0.0      0.0
              nfs9              0      0.0          0          0      0.0      0.0
              sd0              0      0.0          0          0      0.0      0.0
              sd0,a              0      0.0          0          0      0.0      0.0
```

```

sd0,b          0      0.0      0      0      0.0      0.0
sd0,c          0      0.0      0      0      0.0      0.0
sd1            2      0.0      3    1958      0.0      8.0
sd1,a          0      0.0      0      0      0.0      0.0
sd1,b          0      0.0      0      0      0.0      0.0
sd1,c          0      0.0      0      0      0.0      0.0
sd1,d          0      0.0      0      0      0.0      0.0
sd1,e          0      0.0      0      0      0.0      5.8
sd1,f          2      0.0      2    1956      0.0      9.0
sd1,g          0      0.0      0      0      0.0      0.0
sd2            0      0.0      0      0      0.0      0.0
  7.0      100      0.0      0
    0    7935      100      8    240      97      0      0
  0.00      0.0      0.00      0.0 143189
346430      26 22887      0.00      0.00      6541 3517768
    1      2      0
    0      0    1903      0      0      0
86/30000      0 32815/129797      0 615/615      0      0/0
  0.00      0.00
  0.00      0.00      0.00      0.00      0.00      0.00
    0.00      0.00      0.00      0.00      0.00
1548184 39927664
423872064 45155219      0 1032183808 911917808      0 165756928      0
...

```

The **sar** utility runs in the background to display metrics on the system, with an interval of every few minutes on a nominal system, or every 5 or 10 seconds on a system that is under examination or undergoing troubleshooting to help diagnose system anomalies that impact SBRC (for example, to identify periods of spikiness).

## iostat

The **iostat** utility reports I/O statistics such as terminal, disk, and tape I/O activity, as well as CPU utilization. The following is a sample output:

```
# iostat 1 10
```

tty		sd0			sd1			sd2			nfs1			cpu			
tin	tout	kps	tps	serv	kps	tps	serv	kps	tps	serv	kps	tps	serv	us	sy	wt	id
0	55	11	1	7	93	2	14	0	0	0	0	0	0	6	2	0	91
0	235	0	0	0	1021	1	13	0	0	0	0	0	0	72	25	0	3
0	81	0	0	0	1026	1	10	0	0	0	0	0	0	72	25	0	3

```
0 81 0 0 0 1024 1 13 0 0 0 0 0 0 72 25 0 3
...
```

If the CPU utilization from **prstat** is relatively low but the I/O utilization (kps on any device) is nearing the I/O bandwidth utilization, you may have a problem with I/O.

For more general usage information on the **iostat** utility, see the man page at <http://download.oracle.com/docs/cd/E19253-01/816-5166/6mbb1kq4q/index.html>.

## vmstat

The **vmstat** utility shows virtual memory usage statistics regarding kernel thread, virtual memory, disk, trap, and CPU activity. The following is a sample output:

```
# vmstat 1 10
```

```

kthr      memory          page        disk        faults        cpu
r  b  w   swap  free   re  mf pi po fr de sr s0 s1 s2 --  in  sy  cs us sy id
0  0  0 19956200 13155320 43 379 11 26 26 0 4 1 2 -0  0 8658 31671 14768 6 2 91
11  0  0 19964120 12374728 9 40 0 0 0 0 0 0 1 0 0 60815 358620 149035 72 25
3
3  0  0 19963920 12374576 0 5 0 0 0 0 0 0 0 0 0 61047 352131 146928 72 25
3
```

The **vmstat** utility is useful to determine if the memory utilization is too high. For example, any high pi/po values indicate memory thrashing (that is, more memory is in use by the working set than is available on the system), and you should diagnose for stale sessions, leaks, or add more memory.

For more general usage information on the **vmstat** utility, see the man page at <http://download.oracle.com/docs/cd/E19253-01/816-5166/6mbb1kqms/index.html>.

## prtdiag and sosreport

The **prtdiag** and **sosreport** utilities displays system configuration and diagnostic information. The following is a sample output:

```
# prtdiag
```

```
System Configuration: Sun Microsystems sun4u Sun SPARC Enterprise M3000 Server
System clock frequency: 1224 MHz
Memory size: 16384 Megabytes
```

```

===== CPUs =====
      CPU              CPU              Run   L2$   CPU   CPU
LSB   Chip              ID              MHz   MB   Impl. Mask
---   ---              -
 00    0      0,  1,  2,  3,  4,  5,  6,  7  2750  5.0           7 161
===== Memory Configuration =====
      Memory Available      Memory   DIMM   # of  Mirror  Interleave
LSB   Group   Size              Status   Size  DIMMs  Mode     Factor
---   ---   -
 00    A      8192MB             okay    2048MB   4 no     2-way
 00    B      8192MB             okay    2048MB   4 no     2-way
===== I/O Cards =====
...

```

For more general usage information on the **prtdiag** utility, see the man page at <http://download.oracle.com/docs/cd/E19253-01/816-5166/6mbb1kqdn/index.html>.

An equivalent of **prtdiag** on Linux is a **sosreport** (which may require an additional package, which can be installed with “**yum install sosreport**”). The **sosreport** generates a package of files in **/tmp**, which can be inspected or forwarded to JTAC.

**sosreport**

```
sosreport (version 2.2)
```

```

This utility will collect some detailed information about the
hardware and setup of your Red Hat Enterprise Linux system.
The information is collected and an archive is packaged under
/tmp, which you can send to a support representative.
Red Hat Enterprise Linux will use this information for diagnostic purposes ONLY
and it will be considered confidential information.

```

```

This process may take a while to complete.
No changes will be made to your system.

```

```
Press ENTER to continue, or CTRL-C to quit.
```

```

Please enter your first initial and last name [defaultusername]: juser
Please enter the case number that you are generating this report for: 12345

```

```
Running plugins. Please wait ...
```

```

Completed [51/51] ...
Creating compressed archive...

Your sosreport has been generated and saved in:
/tmp/sosreport-juser.12345-20120312121724-8658.tar.xz

The md5sum is: 68c7948130edf27d71ccc683ebe28658

Please send this file to your support representative.

```

The report can be viewed as follows:

```

# xz -d sosreport-juser.12345-20120312121724-8658.tar.xz

# tar xf sosreport-juser.12345-20120312121724-8658.tar

# cd %ltmachinename%gt-%ltdate%gt-%lttime%gt; ls

```

boot	etc	java	lspci	pstree	sos_commands	uptime
chkconfig	free	lib	mount	root	sos_logs	var
date	hostname	lsb-release	netstat	route	sos_reports	
df	ifconfig	lsmod	proc	sar12	sys	
dmidecode	installed-rpms	lsdf	ps	sestatus	uname	

The preceding generated trees may be inspected freely and remotely, and are very valuable for debugging issues.

## psrinfo

The **psrinfo** utility displays information about virtual processors. The following is an example output:

```
# psrinfo
```

```

0      on-line   since 07/14/2010 18:36:36
1      on-line   since 10/04/2010 11:41:31
2      on-line   since 07/14/2010 18:36:36
3      on-line   since 10/04/2010 11:41:31
4      on-line   since 07/14/2010 18:36:36
5      on-line   since 10/04/2010 11:41:31
6      on-line   since 07/14/2010 18:36:36
7      on-line   since 10/04/2010 11:41:31

```

For more general usage information on the **psrinfo** utility, see the man page at <http://download.oracle.com/docs/cd/E19253-01/816-5166/6mbb1kqdt/index.html>.

## psradm

The **psradm** utility changes the status of the virtual processors. The following is a sample output:

```
# psradm -f 1
```

```
# psrinfo
```

```
0      on-line   since 07/14/2010 18:36:36
1      off-line  since 01/10/2011 15:19:20
2      on-line   since 07/14/2010 18:36:36
3      on-line   since 10/04/2010 11:41:31
4      on-line   since 07/14/2010 18:36:36
5      on-line   since 10/04/2010 11:41:31
6      on-line   since 07/14/2010 18:36:36
7      on-line   since 10/04/2010 11:41:31
```

```
# psradm -n 1
```

```
# psrinfo
```

```
0      on-line   since 07/14/2010 18:36:36
1      on-line   since 01/10/2011 15:19:47
2      on-line   since 07/14/2010 18:36:36
3      on-line   since 10/04/2010 11:41:31
4      on-line   since 07/14/2010 18:36:36
5      on-line   since 10/04/2010 11:41:31
6      on-line   since 07/14/2010 18:36:36
7      on-line   since 10/04/2010 11:41:31
```

For more general usage information on the **psradm** utility, see the man page at <http://download.oracle.com/docs/cd/E19253-01/816-5166/6mbb1kqds/index.html>.

## cpuinfo on Linux

On Linux, the equivalent of **psrinfo** is **/proc/cpuinfo**, which returns information like:

```
# more /proc/cpuinfo
```

```

processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 44
model name    : Intel(R) Xeon(R) CPU           X5687  @ 3.60GHz
stepping      : 2
cpu MHz       : 1600.000
cache size    : 12288 KB
physical id   : 0
siblings      : 8
core id       : 0
cpu cores     : 4
apicid        : 0
initial apicid : 0
fpu           : yes
fpu_exception : yes
cpuid level   : 11
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp
lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm dca sse
4_1 sse4_2 popcnt aes lahf_lm ida arat epb tpr_shadow vnmi flexpriority ept vpid
bogomips      : 7197.00
clflush size  : 64
cache_alignment : 64
address sizes  : 40 bits physical, 48 bits virtual
power management:
processor      : 1
vendor_id     : GenuineIntel
cpu family    : 6
model         : 44
model name    : Intel(R) Xeon(R) CPU           X5687  @ 3.60GHz
...etc...

```

Please see the man page for “proc” for more information.

Summary information can be seen via the **lscpu** command.

```
# lscpu
```

```

Architecture:      x86_64
CPU op-mode(s):    32-bit, 64-bit
Byte Order:        Little Endian

```

```

CPU(s):                16
On-line CPU(s) list:   0-15
Thread(s) per core:    2
Core(s) per socket:    4
CPU socket(s):         2
NUMA node(s):          2
Vendor ID:              GenuineIntel
CPU family:             6
Model:                  44
Stepping:               2
CPU MHz:                1600.000
BogoMIPS:               7197.24
Virtualization:         VT-x
L1d cache:              32K
L1i cache:              32K
L2 cache:               256K
L3 cache:               12288K
NUMA node0 CPU(s):     0,2,4,6,8,10,12,14
NUMA node1 CPU(s):     1,3,5,7,9,11,13,15

```

Please see the man page for `lscpu` for more information.

## Taskset on Linux

Taskset is used to set or retrieve the CPU affinity of a running process, given a PID or a command. This could be used to increase the performance in certain instances.

```
taskset -p -c <cpuset> <PID>
```

Example:

```
taskset -p -c 0-7 20320
```

Where `-p` option indicates the last argument is a PID, the `-c` option indicates the second-to-last argument is a list of CPU IDs (rather than, the default, a bitmap representing which CPUs). The effect of this command would be to make process 20320 bias to the first 8 cores.

## Disabling a CPU on Linux

To temporarily disable a CPU on Linux, use the following command:

```
# echo 0 >> /sys/devices/system/cpu/cpu[#]/online
```



Where [#] is the processor ID of the core to disable. It appears that disabling CPU 0 is not permitted. To enable, echo the value 1. It does not appear to persist between reboots.

## System Statistics

You use the **Statistics** page in Web GUI to display summary statistics for authentication, accounting, and proxy forwarding transactions. You can measure the current, average, and peak TPS values of authentication and accounting transactions.

Authentication statistics summarize the number of authentication acceptances and rejections, with summary totals for each type of rejection or retry (see [Figure 2 on page 33](#)).

Figure 1: Viewing Authentication Statistics

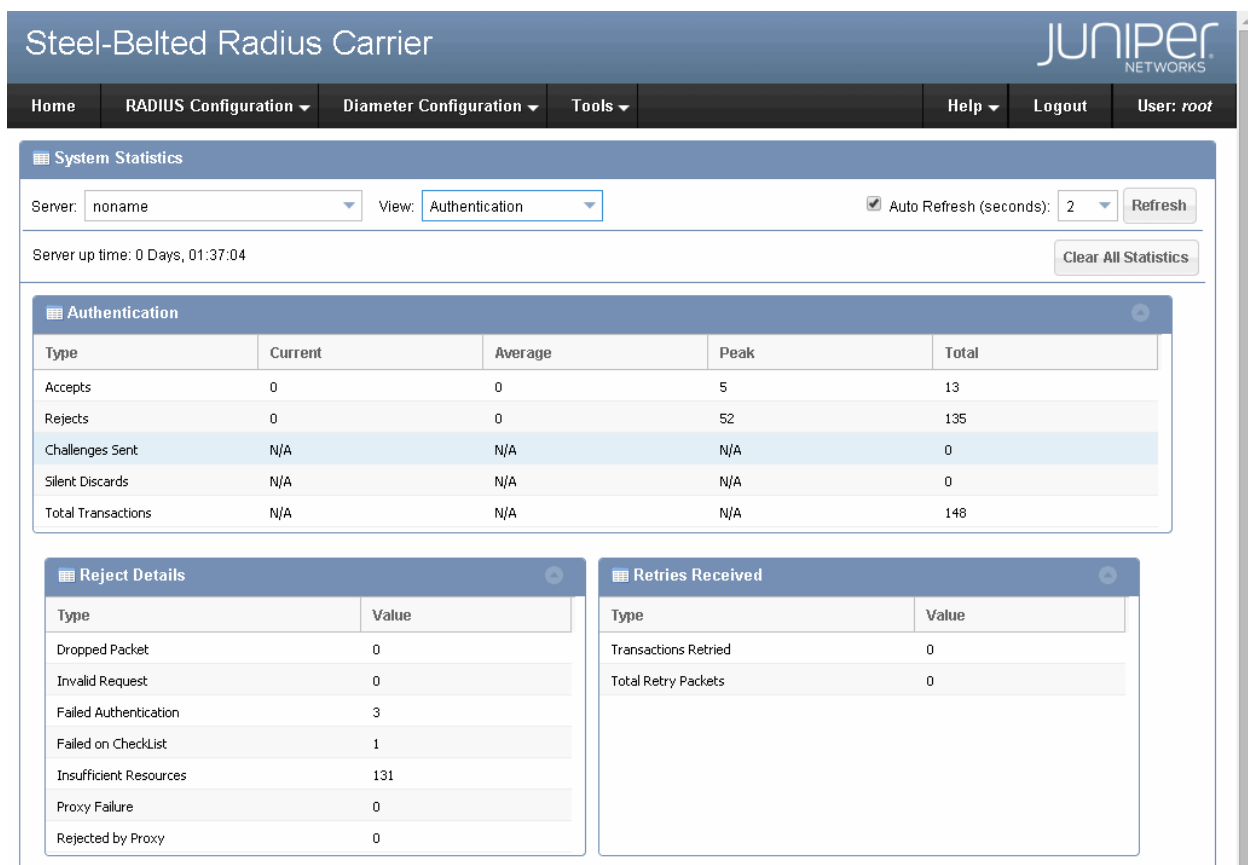
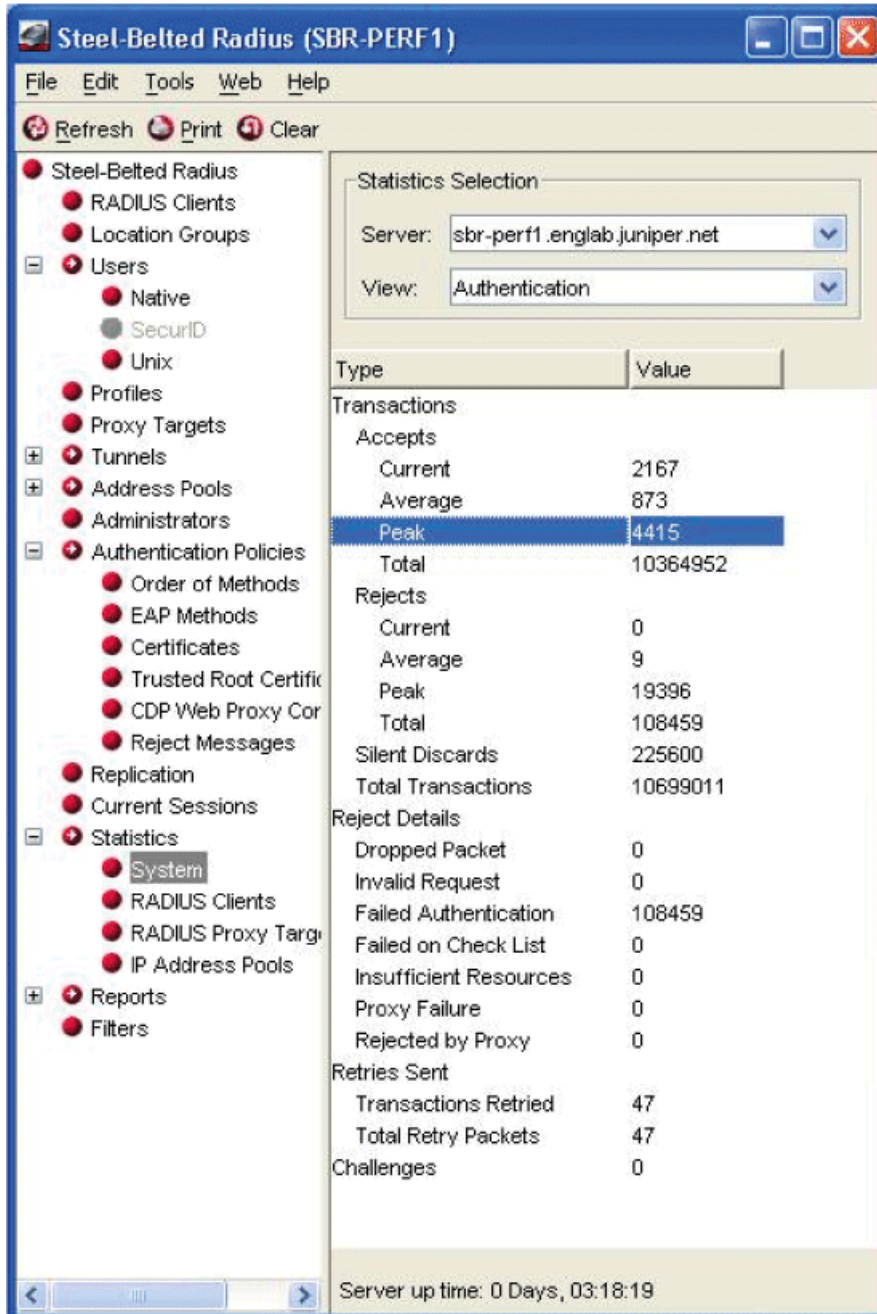



Figure 2: Viewing Authentication Statistics



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 (see [Figure 4 on page 35](#)).

Figure 3: Viewing Accounting Statistics

Steel-Belted Radius Carrier


Home
RADIUS Configuration
Diameter Configuration
Tools
Help
Logout
User: root

### System Statistics

Server: noname
View: Accounting
☒ Auto Refresh (seconds): 2

Server up time: 0 Days, 01:38:16

#### Accounting

Type	Current	Average	Peak	Total
Starts	0	0	52	143
Stops	0	0	0	0
Total Start/Stops	N/A	N/A	N/A	143
Ons	N/A	N/A	N/A	0
Offs	N/A	N/A	N/A	0
Interim Requests	0	0	0	0

#### Failure Details

Type	Value
Dropped Packet	0
Invalid Request	0
Insufficient Resources	0
Proxy Failure	0

#### Retries Received

Type	Value
Transactions Retried	0
Total Retry Packets	0

Figure 4: Viewing Accounting Statistics

The screenshot shows the Steel-Belted Radius (SBR-PERF1) web interface. The left sidebar contains a tree view with the following items: Steel-Belted Radius, RADIUS Clients, Location Groups, Users (expanded), Native, SecurID, Unix, Profiles, Proxy Targets, Tunnels, Address Pools, Administrators, Authentication Policies (expanded), Order of Methods, EAP Methods, Certificates, Trusted Root Certificate, CDP Web Proxy Configuration, Reject Messages, Replication, Current Sessions, Statistics (expanded), System (selected), RADIUS Clients, RADIUS Proxy Targets, IP Address Pools, Reports, and Filters.

The main content area displays the Accounting Statistics for the server sbr-perf1.englab.juniper.net. The view is set to Accounting. The statistics are presented in a table with two columns: Type and Value.

Type	Value
<b>Transactions</b>	
Starts	
Current	4243
Average	877
Peak	4406
Total	10415170
Stops	
Current	4255
Average	877
Peak	4417
Total	10415124
Ons	0
Offs	0
Total Transactions	20830294
<b>Failure Details</b>	
Dropped Packet	0
Invalid Request	0
Failed Authentication	0
Insufficient Resources	0
Proxy Failure	0
<b>Retries Sent</b>	
Transactions Retried	0
Total Retry Packets	0
Interim Requests	0

At the bottom of the interface, the server up time is displayed as 0 Days, 03:18:31.

# Performance Planning

## IN THIS CHAPTER

- Understanding SBRC Call Models | 36
- Understanding the Memory Utilization Levels | 38
- Understanding the CPU Requirements | 42
- Retries | 47
- Load Balancing on RADIUS Front-End Applications and Downstream Proxy Devices | 48
- Handling Downstream Latency and Traffic Spikes | 50

This chapter provides information that helps you understand the SBRC call models, memory utilization levels, CPU requirements, and load balancing on RADIUS front-end applications and downstream proxy devices.

## Understanding SBRC Call Models

A call model, also known as a session call model, is a logical description of the flow of information on the control plane of a carrier system, each part of which is termed a “leg” of the flow. It includes items such as authentication and authorization, service activation and provisioning, call and device control, and termination.

When you plan the sizing and expected performance of an SBRC as a system of systems, a key aspect to determine beforehand is the impact of the call models on the expected SBRC operation, that is, the scope and shape of the normal level of SBRC traffic.

Carriers often use the widely established call models, such as the reference models from networking working groups (such as 3G, WiMAX, and LTE) and modify them for their customer base to model their specific range of services. A subscriber base is often broken down into several key call models with subscribers (current and future), who either have or are expected to have certain classes of service, and statistically operate within identifiable patterns of system utilization. These call models and patterns of system utilization are further decomposed into the work that SBRC front-end applications and back ends (external DB, downstream proxies, and SSR) must do. This is usually done with the help of technical experts armed with existing data and statistics from the carrier or operator in consultation with the Juniper

Network's sales engineering force or value-added reseller. These often consult with Juniper Networks' product engineering team for the latest information and to seek possible ways to optimize a specific use case.

An actionable set of call models and statistical information can include information like the following:

- 6 million total subscribers
- 2 million xDSL device address authentications, always on with an interim period of 4 hours and reauthentication period of 24 hours (+/- 1 hour), randomly occurring throughout the day
- 1.3 million fixed always-on WiMAX Device TLS with an interim period of 1 hour
- 30 percent of 500,000 subscribers, simultaneously using roaming WiMAX TTLS, each with:
  - An expected CPS of 3 per day with an interim period of 10 minutes
  - An expected regular peak of 200 CSPA for 20 minutes (at 9:00 a.m., 12:00 p.m., and 5:30 p.m.)
  - A tolerable absolute peak of 2000 CSPA during certain external events (for example, earthquakes or the Superbowl)
- Each GGSN supports 250,000 subscribers, which will reauthenticate at a limited rate of 2500 CSPA on restart; as well as attempt to support 2 GGSN simultaneous restarts, load balanced across both NOCs in normal operation, on a best-effort basis
- 100,000 heavy use commercial roaming WiMAX users with:
  - An expected CSPA of 1 per hour owing to cell movement
  - An interim period of 10 minutes
  - A peak of 800 CSPA (between 7:00 a.m. and 7:30 a.m.)
  - 35 percent simultaneous dormant subscribers
- 1000 foreign agent SPSs from recognized Wi-Fi relationships
- IP not assigned from the SSR and no concurrency checking

From the above, breaking the information into the respective use cases, you can measure the limits of CPS or CSPA that the system might simultaneously be called to do. You can derive the requirements for a system in terms of SSR hits, required SSR sharding (or splitting up the subscriber base over multiple SSRs), the amount of CPU power in GHz required, and SBRC front-end applications required to support the system that would need to be in each of the two NOCs, either of which might be required to handle the entire load in a failure condition.

After a thumbnail analysis of the above, you can see that scaling for authentication proves to be strongly related to the peak WiMAX rates (CPU intensive) on the front-end applications, and further closely related to the number of interims flowing through the system, up to the limits of SSR hits-per-second that one SSR can support.

There are multiple ways to optimize the system based on the performance results for specific hardware (see “[Hardware Used](#)” on page 81 for more information). In addition to optimizing the behavior of SBRC, you need to consider such possibilities as decreasing the interim and fixed reauthentication periods to significantly reduce the background load on the SSR to permit a wider bandwidth to support GGSN resets, or limiting spikiness through the use of flood queues and lengthening retry semantics. To maintain the expected security level, you can use a non-default cipher suite, smaller prime number generation for temporary Diffie-Hellman Ephemeral (DHE) Encryption keys, or a smaller TLS server certificate with more frequent certificate updates.

## Understanding the Memory Utilization Levels

This section provides information on the memory utilization levels for SBRC standalone and SBRC SSR cluster setups. It explains:

- [Memory Utilization on Standalone on page 38](#)
- [Memory Utilization on SSR Cluster on page 39](#)

### Memory Utilization on Standalone

[Table 10 on page 38](#) describes the memory utilization levels for the SBRC standalone setup. Remember, SBR standalone is limited to a maximum of 4G of memory utilization, in which the CurrentSessions as well as the operating memory for SBR transaction processing must be maintained.

**Table 10: Memory Utilization on Standalone**

Parameter	Data Memory	SA Overhead Bytes per Record (BpR)	SA Maximum BpR	SA Reasonable BpR
Current Sessions	Memory for data (per session)	688	5294	1116
IP Addresses	Memory (per IP)	14 + (#ips*24/256)	–	14 + (#ips*24/256)
CDR Acct Records	Memory (per session)	LDAP CDR server for SS7, separate memory space, possibly separate machine, similar to cluster sizing	–	LDAP CDR server for SS7, separate memory space, possibly separate machine, similar to cluster sizing
SimAkaPseudonymKeys	Memory (per key)	64	64	64

Table 10: Memory Utilization on Standalone (*continued*)

Parameter	Data Memory	SA Overhead Bytes per Record (BpR)	SA Maximum BpR	SA Reasonable BpR
SimAkaReauthCache	Data memory (per cache)	228	1514	1514
SMSAccounts	Data memory (per Acct)	N/A	–	–
TtlsResumptionAttrs	Data memory (per TTLS session)	104	8152	Reasonable values vary widely with attribute stored
TtlsResumptionCache	Data memory (per TTLS session)	132	7620	Reasonable values vary with key size and other items
UserConcurrency	Data memory (per concurrency entry)	N/A	–	–
WiMaxDHCPRootKeys	Data memory (per DHCP root key)	208	208	208
WiMaxHaRootKeys	Data memory (per HA root key)	208	208	208
WimaxMobilityKeys	Data memory (per WiMAX session)	604	2394	Varies widely

### Memory Utilization on SSR Cluster

[Table 11 on page 40](#) describes the memory utilization levels for the SSR cluster setup.



Table 11: Memory Utilization on SSR Cluster

Parameter	Data/Index Memory	NDB Overhead Bytes per Record (BpR)	NDB Maximum BpR	NDB Reasonable BpR
Current Sessions	Data memory (per session)	344	4684	772
	Data memory for unique indexes (per session)	324	968	472
	Index memory (per session)	192	192	192
IP Addresses	Data memory (per IP)	104	104	104
	Index memory (per IP)	32	32	32
CDR Acct Records	Data memory (per session)	296	2320	764
	Index memory (per session)	16	16	16
SimAkaPseudonymKeys	Data memory (per key)	72	72	-
	Index memory (per key)	32	32	32
SimAkaReauthCache	Data memory (per cache)	600	1888	-
	Index memory (per cache)	16	16	16
SMSAccounts	Data memory (per Acct)	360	1824	-
	Index memory (per Acct)	16	16	16
TtlsResumptionAttrs	Data memory (per TTLS session)	104	8152	~3800
	Index memory (per TTLS session)	16	16	16
TtlsResumptionCache	Data memory (per TTLS session)	120	7604	~4800
	Index memory (per TTLS session)	16	16	16
UserConcurrency	Data memory (per concurrency entry)	116	200	200
	Index memory (per concurrency entry)	16	16	16

Table 11: Memory Utilization on SSR Cluster (*continued*)

Parameter	Data/Index Memory	NDB Overhead Bytes per Record (BpR)	NDB Maximum BpR	NDB Reasonable BpR
WiMaxDHCPRootKeys	Data memory (per DHCP root key)	56	120	120
	Index memory (per DHCP root key)	48	48	48
WiMaxHaRootKeys	Data memory (per HA root key)	112	112	112
	Index memory (per HA root key)	16	16	16
WimaxMobilityKeys	Data memory (per WiMAX session)	5760	5760	Varies widely
	Index memory (per WiMAX session)	128	128	128

- Overhead—In standard model, this column includes zero-length data for each variable-length field.
- Reasonable—Include the most used AVPs with expected and usual length fields. In certain cases, adding customer-defined fields can increase the memory utilization for a particular use case over the defined reasonable values.
- Maximum—In certain cases, this column is entirely filled with data, and might go over this with customer-defined fields.

**NOTE:** The data memory and index memory mentioned in [Table 11 on page 40](#) for the SSR cluster refer to settings in the /opt/JNPRhadm/config.ini file.

You can accurately determine the reasonable values by examining a few instances of the live RADIUS traffic and determining the actual lengths of typical data in the fields that are stored in the SSR. Add the actual length of data to the overhead column to construct a reasonable model of a session for your case. You can increase the data by 10 to 20 percent, as a margin of safety and for future expandability, and calculate the total usage.

The SSR memory requirement per D node for a system is calculated as follows:

1. Add the data memory and index memory values together for the expected values of sessions/addresses and other rows.
2. Multiply the value by the maximum number of concurrent sessions that will be supported.

3. Divide the value by the number of stripes (1, 2, or 3 for 2-, 4-, or 6-node SSRs, respectively).
4. Add 2G for the system overhead.
5. Round up to the next multiple of memory sizing (usually 2G or 4G increments).

## Understanding the CPU Requirements

**NOTE:** When analyzing the machine specifications for performance, you have to target throughput rates that take into account failure conditions.

For instance, to account for a GGSN reset that might attempt to reconnect 20,000 users in one minute after a failure, you would need to add the load for an Accounting-ON, which attempts to close 20,000 sessions (at approximately 1000 sessions a second, which equals to 20 seconds), plus 500 authorizations per second plus 500 accounts per seconds for 40 seconds, to whatever baseline in terms of authorizations, accounts, and interims you have as an average. Double each of those values to support two GGSNs restarting, for instance, after a power outage on a shared UPS.

When analyzing the expected performance of a given SBRC system, you must evaluate the following three items in terms of the CPU requirements:

- Number of CPUs (includes *cores*, as well as *virtual processors* [VPs])
  - A single chip may have multiple cores.
  - A *core* represents a single CPU unit and multiple cores on a chip often share a memory bus or I/O bus.
  - *Virtual processor* is a way to optimize the use of a core by permitting more threads to execute on the same core, while one thread is awaiting a memory or bus operation.
  - Cores are an accurate metric of actual performance, as the VP's optimization is not constant, but depends on the workload.
  - For example, the SPARC64 VII processor chip used in the M3000 has four cores and each has two VPs.
- Speed of single CPU or single core
  - Each core of a SPARC64 VII processor executes at 2.52 GHz, which means that one thread can execute 2.52 billion simple instructions per second.

- This is the most important item for NDB, which has fewer execute threads (two or four Local Query Handlers to do most of the work) running simultaneously.
- Total CPU in terms of GHz.
  - This number represents the amount of work that the whole system can execute.
  - Multiply the clock-rate of the processor by the number of cores.
  - For example, the SPARC64 VII running at 2.52 GHz can execute 2.52 x 4 cores, or 10.08 GHz of work.
  - This is the most important number for SBR, which is well-behaved and generally scales to use all the CPU resources available until:
    - The I/O limit is reached
    - A single thread or other threads (such as the thread that reads RADIUS transactions from the network) utilizes all of one core's worth of CPU.

Table 12 on page 43 describes the SPARC processor models. For the latest information on SPARC processor models available for shipping, refer to your Oracle Sales Engineer.

**Table 12: SPARC Processor Models**

SPARC Processor Model	Number of Cores	Number of Virtual Processors	Single-Core GHz Range	Total GHz
UltraSPARC T1	8	32	1.0-1.4	8-11.2
UltraSPARC T2	8	64	1.2-1.4	9.6-11.2
SPARC 64 VI	2	4	2.15	4.3
SPARC 64 VII	4	8	2.52-2.88	10.08-11.52
SPARC T3	16	128	1.65	26.4
SPARC T4	8	64	3.0	24.0

Table 13 on page 44 shows SPARC processors.

Table 13: SPARC Processors

System Model Number	Processor Type	Maximum Number of Processors
M3000	SPARC64 VI	1
	SPARC64 VII (recommended)	
M4000	SPARC64 VI	4
	SPARC64 VII (recommended)	
M5000	SPARC64 VI	8
	SPARC64 VII (recommended)	
M8000	SPARC64 VI	16
	SPARC64 VII (recommended)	
M9000	SPARC64 VI	32–64
	SPARC64 VII (recommended)	
T1000	UltraSPARC T1	1
T2000	UltraSPARC T1	1
T5120	UltraSPARC T2	1
T5140	UltraSPARC T2+	2
T5220	UltraSPARC T2	1
T5240	UltraSPARC T2+	2
T5440	UltraSPARC T2+	4
T3-1	SPARC T3	1
T3-2	SPARC T3	2
T3-4	SPARC T3	4
T4-1	SPARC T4	1

Table 13: SPARC Processors (*continued*)

System Model Number	Processor Type	Maximum Number of Processors
T4-2	SPARC T4	2
T4-4	SPARC T4	4

Using the total GHz, as explained in [Table 12 on page 43](#) and [Table 14 on page 46](#), you can estimate the expected performance based on the results in the current installation, or on the expected results based on test results recorded in “[SBRC System-of-Systems Performance Reference](#)” on page 81.

For example, the results recorded in “[SBRC System-of-Systems Performance Reference](#)” on page 81 are for SBRC doing TTLS authentication with a cipher suite 0x002f on an M3000 2.75 GHz is 888 accepts per second.

- 2.75 GHz x 4 cores = 11 GHz.
- A SPARC T3-4 model (which can have 4 processors) can perform 26.4 x 4= 105.6 GHz total.
- Thus, the SPARC T3-4 model should be able to perform upwards of 888 x (105.6 / 11) = 8524 TTLS accepts per second. This calculation should prove accurate, but might be limited by the single thread handling authentication network traffic as its utilization reaches the entirety of one 1.65 GHz core.
- T4 has special acceleration for single-thread throughput.

In a case of an existing SBRC system running WiMAX, for example, running on three T5120s (one SM node and two SSR D nodes), if you record an average transaction rate of 800 SPS, where the total average utilization of the SM node is 20 percent and the SSR’s single maximum thread utilization is 1.0 percent out of a maximum of 1.56 percent (100 % / 64 virtual processors = 1.56 % per VP from **prstat -L**), and you wish to upgrade to three M4000s with two CPUs:

- For the SBRC front-end application, you can expect a maximum of 8400 SPS [800 SPS x (20.16 total GHz) / (20% x 9.6 total GHz) = 8400 SPS].
- For the SBRC SSR, you can expect a maximum of 2995 SPS [800 SPS x 2.88 GHz / (1.2 GHz x 1.0/1.56) = 2995 SPS].
- The back-end CPS will be slightly higher than these numbers due to decreased contention, as discussed in “[Understanding Performance Metrics](#)” on page 13, but this methodology allows a safe estimate.
- In this example, you are limited by the back-end utilization, and a greater overall increase will be noticed going to four D nodes of the M3000 with one processor running at 2.52 GHz (for approximately 2 x 2995 = 5990 SPS), versus two D nodes of the M4000 with two processors each.

[Table 14 on page 46](#) describes some of the Intel processor examples.

**Table 14: Intel Processor Examples**

Intel Processor Model	Number of Cores	Number of Virtual Processors	Single-Core GHz Range	Total GHz
X5687	4	8	3.6	14.4
X5660	6	12	2.8	16.8
X5690	6	12	3.46	20.76
E7-4870	10	20	2.4 <sup>1</sup>	24

<sup>1</sup> E7-4870 uses Intel “Turbo Boost Technology,” which can attain a maximum frequency of 2.8 GHz under heavy use.

In the Intel processor example servers described in [Table 14 on page 46](#), the following configurations are optional and there is no model number that references all the options specifically:

### Standalone or Front-End Applications

The following server example can be optimized for throughput:

HP DL 380 2 x X5687, 8G, 200G fast (512 Mbps+) SSD (mirrored or RAID 1), or HDD (RAID 0, RAID 1, RAID 10 for highest performance). In the case of SSD, a 6 Gbps SATA with Sandforce-based SSDs (like the Intel 520) performs well for accounting and local CST because the data can be compressed. For long-term or compressed log storage, a separate HDD would be useful.

The following server examples can be used for better proxy and TTLS cases:

- HP DL580 2 x E7-4870, 8G, 200G fast SSDx4 (RAID 10), or 2xRAID 10 HDD
- HP DL580 2 x X5690, 8G, 200G fast SSDx4 (RAID 10), or 2xRAID 10 HDD

The following server example can be used for running four virtual instances of SBR:

HP DL580 4 x E7-4870, 36G memory, 2x (128G fast SSDx2), or RAID—one for each instance.

### Back Ends

The following server examples can be used for back ends:

- HP DL380 2 x X5687, 16-64G, 100G fast SSDx2, or HDD RAID 0 or 10 (mirrored or RAID 0)
- HP DL980 2 x E7-4870, 16-64G, 100G fast SSDx2, or HDD RAID 0 or 10

## Retries

The upstream RADIUS devices (such as NAS devices, load balancers, or SBRC sending to downstream proxies) must reattempt the same RADIUS transaction to the same SBRC front-end applications at least three times (with a 5-second retry period) before trying another front-end application. Another common method is five retries with a 3-second interval before trying another front-end application.

RADIUS is a UDP-based protocol with application-level retransmit semantics that can easily be configured and tuned in the client, unlike the TCP protocol retransmissions which are generally tuned at the system level.

Within the network, many network devices and operating systems drop UDP packets preferentially under certain cases of heavy traffic. Specifically, both Linux and Solaris may drop UDP packets if one of the buffers in its network stack is nearing full capacity. In a lab environment, under heavy load, you will often see some packet loss at the OS level, which occurs in accordance with the protocol design.

The retransmit semantics of three retries x 5 seconds each (3x5) or five retries x 3 seconds each (5x3) provide a good likelihood of avoiding traffic spikes on the network. This gives a reasonable amount of resilience, so that the underlying network protocols can reroute traffic and failure recovery semantics can occur in failure conditions.

A retransmit to one SBRC front-end application activates the RADIUS packet cache, which short-circuits the processing of the in-process or recently processed transactions. This means that a server receiving a second request generally returns the result immediately (with less overhead) without repeating the work, if the result has been returned already.

Using a retransmit time in hundreds of milliseconds conflicts with even the most ambitious failure recovery settings, turning what would otherwise be a recoverable failure scenario (for example, the outage of part of a cluster, or the loss of connectivity on one network adapter) into an unrecoverable one as far as the customer experience is concerned. This causes the system to do more work since the same work might be attempted over multiple front-end applications, and conflict resolution adds additional overhead with each retried attempt.

Three to four retries of 1 to 2 seconds each before trying a second front-end application (similarly for the third) is probably safe under most circumstances with sub-millisecond network latency and few-millisecond back-end DB latency. However, lesser values are entirely unsupported. Larger is better to the point where failures might seriously impact the user. Frequent failures of this mode, in any event, should be diagnosed.

In summary, the retry semantics are a function of the system resilience and should be set higher rather than lower. This avoids making a transient problem appear to be a more permanent one by, for instance, incorrectly failing an authentication and allowing a failure situation to become worse in the process.

Thus, three retries of 5 seconds each are nominal, human-level speeds that will permit wide latitude for successful failover scenarios.



## Load Balancing on RADIUS Front-End Applications and Downstream Proxy Devices

In high-end cases of multiple front-end SBRCs to one SSR, or multiple SAs operating in parallel, it is important that the load balancers in SBRC (whether they are commercial load-balancing devices or versions of SBR SA running proxy and doing round-robin) have the properties described in the following sections.

### Stateful and RADIUS-Aware

Load balancers should be stateful and continue to send packets of the same transaction cycle to the same SBRC front-end application. In the case of EAP over RADIUS, a single EAP stream under RADIUS Auth-Request Challenge-Response pairs must continue on the same front-end application until that front-end application returns a success, reject, or exhibits a failure, in which case the failover of authentication to another SBRC in the middle of an EAP negotiation will become a reject and the user must retry the connection.

The *State* attribute in the RADIUS request and response describes which front-end application processed the request. Alternately, the NAS-IP-address and port for most devices are unique identifiers, and the Calling-Station-ID can be used in some use cases.

In most cases, accounting should be processed on the same front-end application as authentication. If the load-balancing device can parse attributes, the class attribute can be used to route starts, interims, and stops to the originating NAS.

In some cases, for example, multiple SBRC SA front-end applications might be doing heavy authentication work (for example, TTLS with separate inner authentication proxying to SBRC cluster front-end applications), and the devices can send the accounting directly to the SBRC cluster front-end application, bypassing the unnecessary proxy step.

While TTLS may be cluster-enabled and thus reauthentication optimized across multiple nodes, TLS reauthentication optimization for a device, on the other hand, should attempt to use the front-end application that the previous device used, either by adding return attributes for server biasing, or using Calling-Station-Id or the User-Name, if it represents the device ID. If the front-end application fails, this reauthentication optimization does not need to occur and a full authentication cycle against any working front-end application can take its place.

FastFail is a failover mechanism in SBRC SA for proxy. It performs failover of the transactions from one downstream to another until a strobe of a failed target succeeds. This can be used to optimize performance under conditions of temporary failure. If failover might occur to SBRC front-end applications in a different NOC, you should scope your system so that each NOC can handle the work of the entire system.

## Network Configuration for SSR

A key source of contention within the SSR is the lock that must be in use to keep the two mirrored D nodes (data nodes) operating in lock step, related to the “two-phase commit” performance transaction in all redundant SQL systems. If a database node needs to update a piece of data in the row (for example, the SSR CurrentSessions table), it must request its mirrored pair to halt any other changes to that row, update both copies of the row in parallel, then release the lock. The amount of time taken to place the lock and update the data is partly a function of the CPU and the latency of the total operation from the first lock to the last release. Aggregate operations such as session deletes for multiple sessions, or IP address allocation, must take one lock for several ndb-level operations within it, and each will be slower due to latency.

## Why Latency Kills Performance: D-D and S-D

Latency between the data nodes (D) significantly reduces the overall throughput of the SSR. In any system that requires the highest levels of performance and reliability, latency of even multi-milliseconds cannot be recommended. Simple cases that look up by primary key, do not do database updates via interims, or have no phantoms, are more resilient to latency. Since each use case is different, testing for sustained throughput (in a lab environment) with latency while measuring the performance of the system is strongly recommended.

Similarly, the latency between S and D nodes (SBR and data nodes) decreases the throughput of the node due to the present NDB limitation of 120 ndb handles per API node. Each handle has a request outstanding, and these outstanding requests will take longer to process to completion when latency is high. In this case, additional S nodes are required to make up for the latency to meet the same bandwidth.

## Subscriber Scaling

Scalability for the SSR and SBRC front-end application is calculated based on the following dimensions:

- Inserts, deletes, and searches in the SSR indexes become slower as simultaneous sessions are added by a factor of approximately  $O(\log_2 n)$ .
- The number of TPS carried out by the system as a whole, required to sustain the user base, especially at the peak rates. Here, the number of auths, starts, and stops varies largely and is timed variably for mobile users, or items such as always-on devices, which may be rare.

The number of interim accounting transactions scales linearly with simultaneously connected subscribers and the interim period, and causes a significant amount of load especially with a short interim period. Reducing the interim period by half doubles the number of packets that must be managed; doubling the interim period reduces the amount of packets by half.

It is possible for devices that reconnect at set times (for example, at midnight or 2:00 a.m.) or otherwise manage to synchronize in time (due to, for example, a SGSN failure or a GGSN reset) to provide periodic spikes of traffic. Both the amount of traffic due to reconnection and the amount of traffic of future interims

can match in time. You will often see something much closer to peak usage, as opposed to spreading the load out over the hour, which may prove an issue in deployment. Various solutions can be implemented, but the biggest benefit is seen by decreasing the interims that the SSR sees, followed by injecting a time skew in the reconnection process, either through the device, or through control points in SBRC.

### Why Are Interims Hard, and When Can They Be Thrown Away?

Interim accounting is an update transaction managed by the SSR. In most cases, stops only need a primary key search and a delete operation, with limited data transmitted to or from the SSR. Interim transactions include the work of a primary key search and the database row update of an accounting start, plus returning most or all of the data for that row for reliable updating of indexed fields that changed, all under a row lock to keep the data consistent. Of all the per-RADIUS transaction operations against the SSR, the interim packet requires the most work to process.

Accounting interims may have been implemented as a way of doing a stop-loss for accounting purposes, in case the RADIUS server does not receive a stop. You can reliably bill the usage (for usage-based plans) for which you have received an interim and a stop was not received due to a failure. In this case, the SSR does not need to be updated by all interims, and the parameter **UpdateOnInterim** in **radius.ini** can be set to zero.

In certain cases, accounting interims are used to extend the session life. For example, if devices reauthenticate daily, by setting the Session-Timeout to a little over a 24-hour period, you can avoid the use of interims to extend the session life and also set the **UpdateOnInterim** parameter to zero. Interims will still be available for accounting-level processing, but need not hit the CST.

## Handling Downstream Latency and Traffic Spikes

This section explains the following:

- [Thread Control on page 51](#)
- [Flood Control on page 51](#)

**NOTE:** Proxy threads and proxy floods are used only when Block=0 is set in the [Acct] section of the **RealmName.pro** file. See the *Juniper Networks Steel-Belted Radius Carrier Reference Guide* for more information.

## Thread Control

This section provides information on the following thread control settings in the [Settings] section of the **radius.ini** configuration file:

- Max-Auth-Threads = 1 to 100,000 (default 100)
- Max-Acct-Threads = 1 to 100,000 (default 200)
- Max-Proxy-Threads = 1 to 100,000 (default 100)

Each thread setting controls the maximum number of threads allocated to handle the system load. The auth and acct thread settings manage the state and latency of the process as a whole, with the proxy threads coming into play for those proxies whose Block=0. Authentication and accounting, in this case, pass the work to another proxy thread and return a result immediately, and then wait for another transaction.

- Auth-Receive-Realtime-Thread-Priority (default value is 2,147,483,647, not real time) in the [Settings] section of the **radius.ini** file
- Acct-Receive-Realtime-Thread-Priority (default value is 2,147,483,647 not real time) in the [Settings] section of the **radius.ini** file

For *WorkerThreadStackSize* in the [Settings] section of the **radius.ini** file, the default value is 524,288 bytes (512K) for both Solaris and Linux. If a value is set for *WorkerThreadStackSize*, it overrides the system default stack size for worker threads (128K through 256K is sufficient in most cases); however if the value is too small, SBRC may core.

## Flood Control

When all the auth and acct threads 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.

To see the flood queue in action, turn on the Flood info parameter in the [Status] setting of the **radius.ini** file. See the *Juniper Networks Steel-Belted Radius Carrier Reference Guide* for more information.

## Settings

- Set the Max-Auth-Floods, Max-Acct-Floods, and Max-Proxy-Floods values to 0, which is the number of packets retained by the flood queue. The memory utilization overhead will be slightly higher than the average packet size times the number of floods.
- 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 will be 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; 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. Recommended when you know you will have to throw packets away, in certain cases, to keep up with 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. Pops a random packet in the middle of the queue as the next item. This is rarely recommended, but approximates random delays. For certain use cases, it helps to moderate the effects of DDOS attacks (for example, avoid one NAS being loaded with responses).

# Understanding the SBRC Network Architecture

## IN THIS CHAPTER

- [SBRC Architecture Overview | 54](#)
- [Understanding “Geodiversity” Between the SSR D Nodes | 55](#)
- [NDB Cluster Timeout Dependencies | 56](#)
- [High Throughput Configurations | 58](#)
- [Risks to Uptime within the Architecture | 58](#)

This chapter provides a reference to a highly reliable network architecture for SBRC. It explains some existing error-handling techniques in SBRC which can be used to improve the overall performance and reliability of the system of systems.

SBRC cluster solution is architected based on principles which permit upwards of “5 9s” (99.999 percent) system uptime as a whole, outside of maintenance windows, making the system highly resilient to failures.

Redundancy is offered at multiple levels in SBRC:

- Within the RADIUS transmission structure, built on UDP, NAS devices at the application-level can retransmit to different devices after several attempts to the same device have failed before timing out the user.
- Within authentication, accounting, and proxy functionalities, multiple back ends or downstream RADIUS servers are attempted with appropriate retry semantics.
- Within the SSR, MySQL NDB is resilient to system outages of a single component (a machine or a network component) and some multi-component failures (for example, cluster splits).
  - At the networking layer, utilizing techniques such as IP multipathing (IPMP), failures in hardware components such as network adapters, cabling, and switch outages, can be managed.
  - At the machine layer, using RAID-5 or external storage with RAID mitigates the risk of disk failures. The implementation of Oracle’s advanced system maintenance in terms of hot-swappable drives and power supplies limits the risk of total subsystem unavailability. The regular use of ECC (Error Correcting) memory helps to avoid memory corruption.

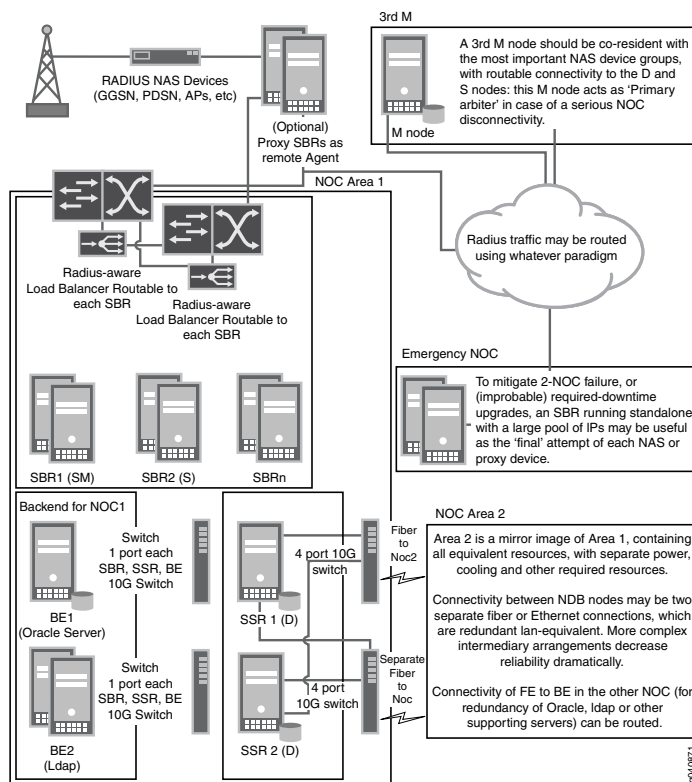
This chapter explains:

## SBRC Architecture Overview

The following two key principles are primarily considered in a high availability system:

- **Redundancy**—The function of each network component and the subsystem should be replicated completely, both in terms of functionality as well as throughput. In case of a failure, half of the SBRC system or subsystem must be able to support the work of the entire system.
- **Separation**—Components that can fail individually should be permitted to fail individually. Thus, the NOC Area 1 and NOC Area 2, representing two different NOCs or two different sections of the same NOC (see [Figure 5 on page 54](#)) should function independently when an environmental failure occurs.

Figure 5: SSR Redundancy Architecture



Routing of the RADIUS traffic can be done using reliable networking techniques, either at OSI Layer 2 or 3. Since UDP traffic is retransmitted by the application layer at a comparatively slow retransmit rate, it is resilient to routing changes and packets being dropped during that reconfiguration.

In general, the Transmission Control Protocol (TCP) and Stream Control Transmission Protocol (SCTP) also have a relatively strong tolerance for additional latency due to routing changes, and traffic to the back-end servers (SBRC front-end application to authentication, accounting databases, or SSR) may be routed, although switching generally provides the lowest latency.

However, the intra-D node traffic, which attempts to keep two copies of the same data in lock-step, requires both low latency and high reliability, and should be managed through Layer 2.

IPMP, used to configure a single virtual address, which floats between one of the two physical adapters, is recommended. This is generally used in active-standby mode with active probing and failure detection time (which may be from 200 to 400 milliseconds), and can be varied based on the relevant heartbeat timeouts, in the case of NDB, and RADIUS retransmit semantics or database timeouts for other back ends.

See the *Oracle whitepaper Highly Available and Scalable Oracle RAC Networking with Oracle Solaris 10 IPM* (<http://www.oracle.com/technetwork/articles/systems-hardware-architecture/ha-rac-networking-ipmp-168440.pdf>) for more information.

In a red-black network that separates the outward facing components from the inner “back-end” components, each SBRC S node will have a single virtual address on two physical adapters connecting to the two externally facing network devices. Each SBRC S node will also have a second virtual address connecting to the two physical adapters, which in turn connects to the back ends (SSR or authentication/accounting databases), the latter of which should generally be switched (and may provide Layer 2 routed access to a failover server in another NOC).

The emergency NOC can be placed in a separate location as a server of last resort to lessen the impact of critical, multi-system failures. A determination can be made on how promiscuous this server is: in many cases, where there is no response from the primary system (rather than a response with a reject), from a business point of view, a carrier can permit temporary access freely instead of being rejected due to these central failures.

The third M node used by the SSR must have at least low-bandwidth medium latency (less than 100 milliseconds round trip) routed connectivity, possibly via a firewall, to each D node and to the other M nodes. This M node serves as the final arbiter to determine which side of the cluster should be considered ‘alive’ if both fiber connections between the two NOCs split entirely. If there are only two M nodes in two different NOCs, then on a cluster split, one set of nodes survives and the other fails, but the surviving NOC might also be undergoing a connectivity failure to the rest of the system during the failure. This third M node should be collocated with the most important exterior devices, if any, with which you wish to preferentially maintain connectivity, in the event of certain major network failures.

## Understanding “Geodiversity” Between the SSR D Nodes

Being the very core of the system’s reliability, SBRC does not claim support for geodiversity among the SSR D nodes, since the underlying technology (Oracle MySQL NDB) does not claim such support. At the SBRC level, geodiversity can be supported at a split in the RADIUS layer with a second NOC installation with either its own SSR, or a load-balanced series of standalone systems to provide additional, perhaps limited, functionality through an unrelated SSR in a primary/failover way.



That disclaimer made, as a matter of prudential and engineering judgment, a pair of single-mode fibers, each end connecting to switches at two different NOCs is a legitimate mechanism of providing LAN-equivalent functionality which should provide a similar level of uptime and supportability as a non-geodiverse solution.

However, in terms of network architecture, the further away from a literal pair of fibers you get, the further from optimal uptime you will achieve: each intervening piece of equipment, no matter on which network layer it operates and even if the device itself reaches 5 9s or greater, actually gives greater risk, from both a technical ( $99.999\% * 99.999\% = 99.998\%$ , and that is multiplicative for each device in series) as well as a human point of view. This is due to the possibility of inducing certain types of network latency (for example, when being reconfigured or reprovisioned) that will lead to a temporary outage and require additional recovery time, or an extended outage due to human or system error. It is often difficult to diagnose these transient latency increases, although with SBRC Release 7.4.0, logging within the SSR D nodes has been improved.

In any managed MPLS network, even rigorous SLAs can be written in such a way that jitter, packet loss, and latency might rise above the tolerances of the ndb transport, and as such, cannot be supported.

## NDB Cluster Timeout Dependencies

This section describes timeout dependencies between various SSR data nodes. A certain ratio must exist between each OSI stack-level failure condition, specifically between NAS clients and the RADIUS front-end applications, the RADIUS S node and the D nodes, and the ndb and dbapi nodes (M nodes and D nodes). This dependency is related to timeout values associated within the network and the NDB itself.

The network between the S nodes and the D nodes has the following timeout dependencies.

- If you are using IPMP, the IPMP probe value must be lower than twice the heartbeat timeout appropriate for the connection. Widely divergent values may impact performance in a failure case, leading to unexpected outage.

**NOTE:** Default values for the S or M nodes and the D nodes are controlled by the `/opt/JNPRhadm/config.ini` file on the M nodes. This value is set by **HeartBeatIntervalDbApi** and is 1500 milliseconds by default. The inter-D node timeout is set by **HeartBeatIntervalDbDb** and is 200 milliseconds by default.

- Heartbeats are implemented in and among the D nodes so that failures are more quickly detected than the underlying TCP failure mechanism can detect. The initial detection of fault happens after the interval of  $4 \times \text{HeartBeatInterval}$ . After initial detection, the D nodes attempt to repartition and form a valid cluster. This operation can take several to many seconds, depending on the type and mode of failure.

Single D node hard failures or hard networking loss are quickest to detect. Complete cluster splits and serious network faults take longer to detect and compensate.

System overload affects fault-recovery performance. Many outstanding transactions take longer to roll back than a few outstanding transactions.

- During an extended loss of service due to a significant failure, such as loss of connectivity between two halves of a cluster, SBR Carrier might need to reconnect to the new cluster to continue processing. The failures of reconnection are managed by timers set by the [Ndb] values—**DelayBetweenConnectRetriesSec** and **ReconnectRetriesin**—in the **dbclusterndb.gen** file. Setting these values higher than the defaults can make the system more resilient at the expense of a period of dropped RADIUS traffic. Setting the values of **TimeoutForFirstAliveSec** and **TimeoutAfterFirstAliveSec** lower may also increase resiliency.
- Some NDB operations are designed to retry the network to avoid lock contention. In cases where the underlying network is prone to latency or dropped packets, increasing the values of **Retries** and **DelayBetweenRetriesMillisec** in the [Database] section can improve performance and decrease delays.
- In cases where the underlying network is prone to short or long periods of latency, fault, or other unexpected cases, setting the value of **HeartBeatInterval** higher and setting all the proportionally related values appropriately can make the system more resilient. The trade-off is a fast detection of serious failures against the acceptance of temporary processing delays. The delays are due to minor faults that are otherwise survivable.
- In cases of NDB cluster failures due to extended one-way traffic failure between the inter-D and SM-D network, you need to automatically restart a node. A correct network design should not permit this failure to happen. IPMP probes with the correct values, for instance, cause this to fail over to a working link. The **HeartBeatOrder** fixes temporary instances of this type of failure. See the *SBR Carrier Reference Guide* for more information about the **HeartBeatOrder** parameter.
- Certain failure conditions may require you to perform a manual restart. These failure conditions are usually associated with serious, extended, and pathological network dysfunctions.
- The default settings of **CacheLowWater**, **CacheHighWater**, and **CacheChunkSize** may cause badly degraded performance. The defaults cannot be made higher because one S node can pre-cache all the addresses in a small pool if the **CacheLowWater** is set higher than the number of addresses in a pool. Default values of **CacheLowWater** and **CacheChunkSize** are related to the transaction rate of new address allocations for the installation. Hence, it is impossible to run out of addresses before the threads can fill up the cache, and you can use Per-Pool settings to set any small pools much lower than the default.

Set **CacheThreadVerbose=1** and inspect the logs for Emergency allocations. If **CacheLowWater** and **CacheChunkSize** are too low, then performance is degraded. Another indicator of degraded performance is low CPU utilization on the front-end applications and high CPU utilization on NDB.

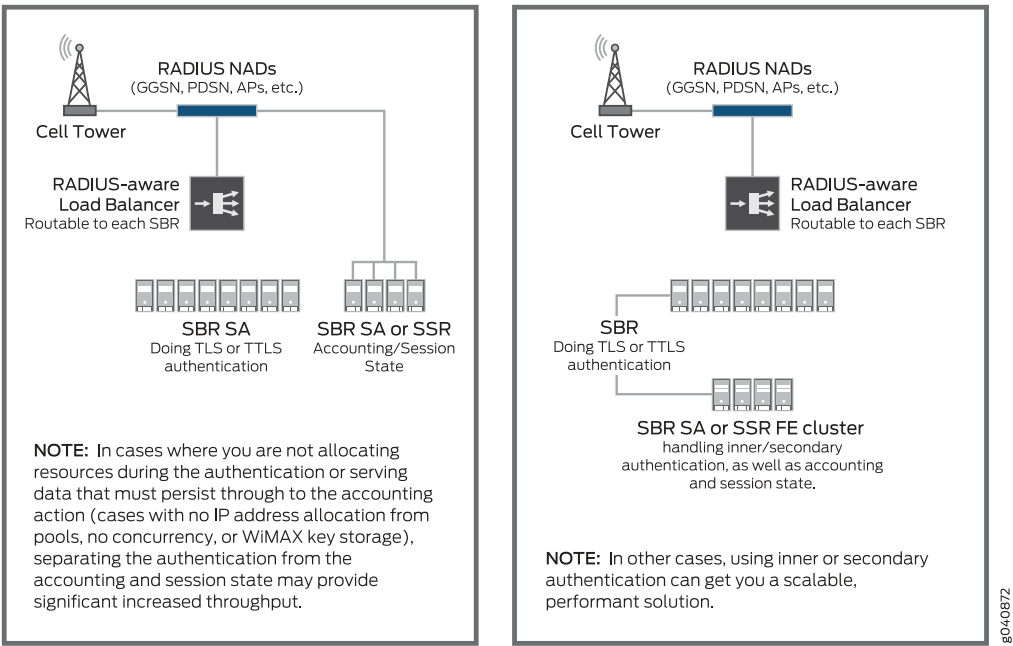
- The **ConnectCheckIntervalDelay** parameter comes into play when a heartbeat timeout occurs. At this time, the node will begin a connectivity check, pinging all nodes 3 times with the configured interval. The responses indicate whether the other nodes are trustworthy (that is, respond 100%), untrustworthy,

or down (no response). Only messages from those nodes that are trustworthy will be considered during the regular election cycle. This feature helps protect against intermittent latency spikes.

# High Throughput Configurations

In cases that require lesser levels of redundancy, a system of SBRCs can be configured with a more standalone approach to provide scaling. The separation of authentication and accounting paths can be made when the RADIUS accounting (especially, the start) carries all the information that needs to be contained within the session state.

Figure 6: High Throughput Configurations



When the advanced cryptography load sufficiently outweighs the resource-allocation work (for concurrency and IP address allocation through the SSR), load-balancing the cryptography over multiple machines while doing the resource allocation in a central way can provide a total throughput increase.

# Risks to Uptime within the Architecture

Risks can be classified under the following three categories:

- Resources on which SBRC is dependent (such as external databases) can fail to function and cause the SBRC to do additional work to recover, extending the service outage until the entire system has reached a functional stasis again. For example, with a very large number of worker threads and a high transaction rate and high disconnect or reconnect rates, SBRC can take a longer time to process the backlogged transactions after a target back-end server recovers from failure than the duration of the initial outage.
- Partial hardware-based failures can cause unexpected broader problems. For example, networking among the D nodes, consistent latency transactions, especially those that are pinpoint or peer to peer (generated by an incorrect firewall rule, or router reconfiguration update operations) of certain values can cause a temporary cluster outage.
- Software errors:
  - While most software errors are automatically recovered and affect one node only, being a complex system, there are a few forms of software errors that could cause an extended outage. To manage this risk, submit cases of transient errors with JTAC with the appropriate information as referenced in the troubleshooting section, as soon as possible, to avoid the problem being repeated or becoming worse.
  - The most dramatic software errors are often termed a “packet of death.” invariably, in concert with a complex level of functionality, this form of software error involves a packet that is well-formed from a RADIUS perspective and malformed from a processing perspective. Items such as zero or single-byte null fields can cause poorly defended customer-defined SQL statements, stored procedures, or JavaScripts to error seriously. Other items might trigger unexpected side effects within packet processing. Since a great deal of processing is done and given the vast complexity of use cases, enough corner cases are tested for within Juniper Networks this form of software error is still possible.
  - You can lessen these sorts of errors by keeping track of the uptime of the system, monitor and diagnose transient rejects and errors, and implement the recommended final failover emergency NOC. This could be an SBRC running with full logging, by default, to trap the unexpected packets that have failed other processing.

# Debugging Performance and Functionality Problems

## IN THIS CHAPTER

- Determining the Effective Performance and Peak Capacity | 60
- Calculating Performance Scaling on an Existing System | 62
- Debugging Performance Issues | 64
- Notes on Debugging Performance Issues | 70

This chapter provides information you can use to determine the scalability of an existing SBRC system and debug system performance issues.

## Determining the Effective Performance and Peak Capacity

The key to determining the capacity of an existing or a planned SBRC system is not measured by average operational performance, but rather by expected or observed periods of peak activity. SBRC performance should generally be nominal during regular day-to-day system operations with significant unused capacity. The following exercise will help you determine how close the SBRC system is to possible peak performance for the hardware platform under evaluation.

For an existing SBRC system running continuously for a period of time and experiencing peak traffic, failures, and spike conditions, do the following:

1. Determine the “Peak” throughput in authentications, account starts, and stops using the **Statistics** page in Web GUI. Record these values and add them up.
2. Measure the system by simultaneously monitoring the following over a defined period (at least 5 to 10 minutes during “peak” user-level activity timeframes):
  - a. Average auths and accts/s for the period from the Web GUI (which you can clear before starting the test), and note the total interims done over the period.
  - b. Monitor the system using **prstat -L** to view the highest utilized thread for the period and its percentage utilization. This value is the strongest indicator of the overall scalability of SBRC, which

in almost all cases performs well on multiple processors. The hard limit is often the utilization of one key thread that does slightly more work than other threads. This is more important on machines with more and slower virtual processors, as compared to machines with fewer and faster processors such as the M3000.

- c. Use **prstat** to check and note the overall CPU utilization. This helps you determine the available CPU power to do the work of all the threads.
- d. Run **sar** with an interval of 5 minutes (**sar 300 1**). The result should be similar to **prstat**, but with information categorized based on **usr** and **sys**. A large amount of **sys** utilization means the system is processing more I/O and network work.
- e. Run **iostat** with an interval of 5 minutes (**iostat 300 1**) and note any devices that are heavily used. This gives you information about logging performance.

For example, executing our own load test tools for accounting on the M3000 may see a utilization of 3 Mbps, which is an average amount. You can also use (**iostat -E 300 1**) to obtain a utilization percentage number that you can use as a reference to maximum theoretical capacity calculated by the OS.

3. Calculate the expected utilization during peak performance by dividing each result obtained in Step 2a through Step 2e by the average TPS for the test period, and multiplying the value with the peak TPS. This gives you an estimate of the CPU resources and I/O resources that will be utilized during peak performance.
4. Record the following information of the results of calculations based on Step 3:
  - a. The projected highest single-thread utilization from **prstat -L**. If this is more than 90 percent of the single-CPU performance (determined by dividing 100 by the number of virtual processors), you will run out of CPU resources to handle the workload at peak.
  - b. The expected overall CPU utilization at peak. If this is more than 90 percent, you will run out of CPU resources.
  - c. The expected I/O utilization at peak. If this is more than 60 percent of the rated throughput on the drive, which varies by model, you will soon run out of resources. I/O utilization at 60 percent assumes activities such as “seek before writing” as a disk becomes full, which induces latency.

This is a good approach for calculating the existing nominal system usage and determining the performance characteristics at seen peak usages. With these calculations in hand, you can help ensure enough throughput to sustain peak usage for an extended period.

## Calculating Performance Scaling on an Existing System

Use [Table 15 on page 62](#) as a worksheet for calculating the performance scaling numbers of an existing SBRC system.

**Table 15: Worksheet for Calculating Performance Scaling**

Category	Front End 1	Front End 2	Front End 3	Front End 4	Total
Before you start the test, you must gather the historical information of the SBRC system from the <b>Statistics</b> page in Web GUI.					
1. Peak accepts/s.					
2. Peak starts/s.					
3. Peak stops/s.					
4. Interims per start. (Divide the interim requests by total starts.)					
5. Challenges per accept. (Divide challenges by total accepts.)					
Clear the results from the Web GUI and examine the system load at high usage for a defined period (5 minutes). This should not significantly impact the system performance.					
6. Average accepts/s.					
7. Average starts/s.					
8. Average stops/s.					
9. Interims per start.					
10. Challenges per accept.					
View the system statistics during the trial period.					
11. Run <b>prstat</b> with average load during test.					

Table 15: Worksheet for Calculating Performance Scaling (*continued*)

Category	Front End 1	Front End 2	Front End 3	Front End 4	Total
12. Run <b>prstat -L</b> to view the highest thread utilization.					
13. Run <b>sar 300 1</b> to calculate the usr percentage.					
14. Run <b>sar</b> to calculate the sys percentage.					
15. Run <b>iostat 300 1</b> for I/O utilization, in kps.					
16. Run <b>psrinfo</b> to determine the virtual CPU count.					
17. One CPU as a percent of the system.  (Divide 100 by the value of #16.)					
Peak scaling is calculated by the result of #17 (assuming ratios of #4 and #5 are approximately equal to #9 and #10, respectively.)					
18. Calculate the overall CPU utilization at peak.  (Divide the value of #11 by the value of #7 and multiply the result with the value of #2.)					
If the overall CPU utilization (result of #18) is nearing 90 percent, then the system is presently constrained by the CPU utilization.					
19. Calculate the highest thread utilization at peak.  (Divide the value of #12 by the value of #7 and multiply the result with the value of #2.)					
If the highest thread utilization (result of #19) at peak is closer to the result of #17, then the system is presently constrained by the single-core CPU speed.					



Table 15: Worksheet for Calculating Performance Scaling (*continued*)

Category	Front End 1	Front End 2	Front End 3	Front End 4	Total
20. Calculate <b>sar</b> at peak.  (Add the values of #13 and #14, divide by the value of #7, and multiply by the value of #2.)					
<p>If the result of #20 is nearing 90 percent, then the system is constrained by CPU utilization. If this trend continues and exceeds 90 percent, then the system will run out of CPU resources.</p> <p>The result of #20 should be similar to the result of #18.</p>					
21. Calculate <b>iostat</b> kps at peak.  (Divide the result of #15 by the result of #7 and multiply it by the result of #2.)					
<p>If the result of #21 is nearing 60 percent of the rated speed of the drive, then the system will soon become I/O bounded.</p>					

## Debugging Performance Issues

This section describes the three common bounding scenarios of a live SBRC system, which can be verified using the tools listed in [Table 15 on page 62](#). They are:

- [CPU Bound on page 64](#)
- [I/O Bound on page 66](#)
- [Contended Systems on page 68](#)

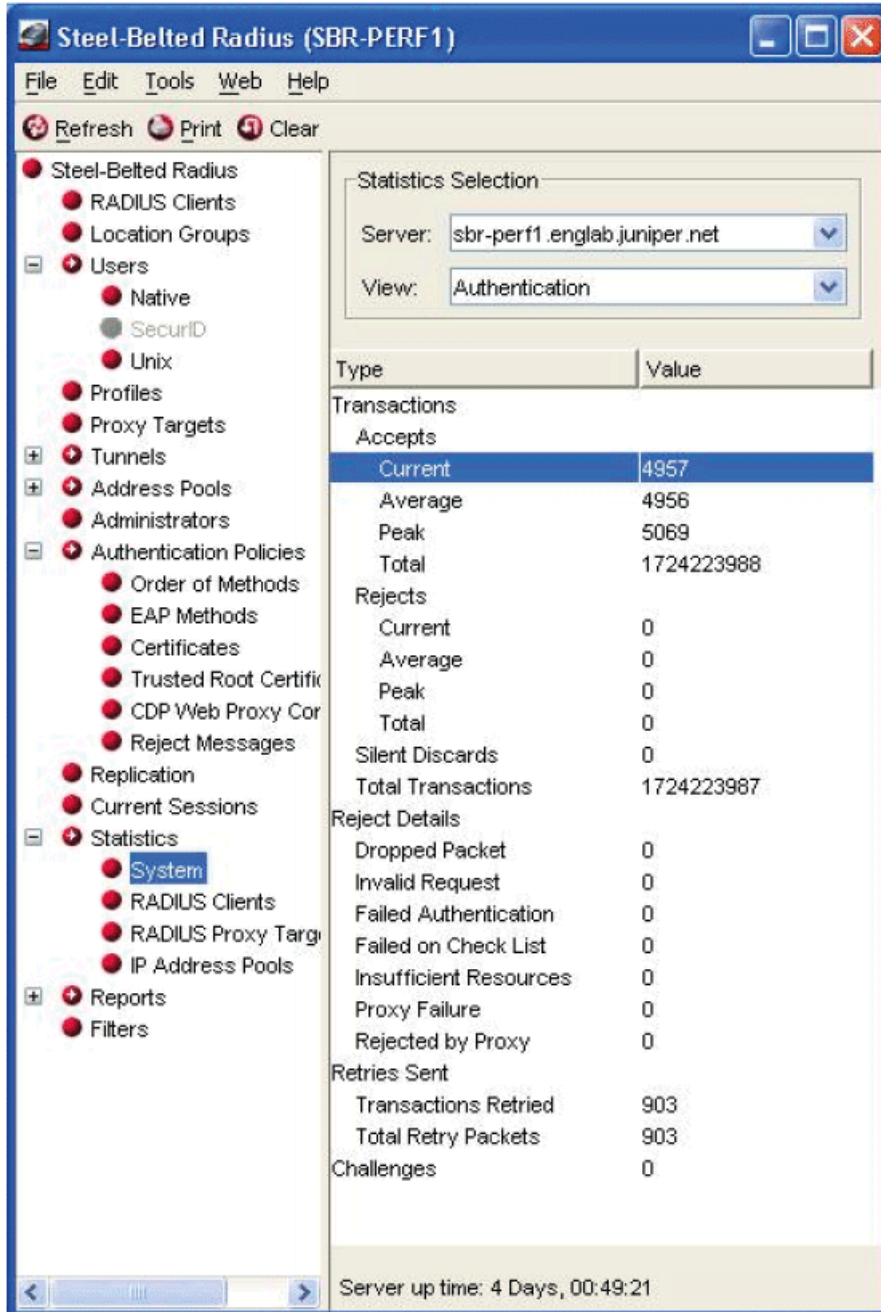
Focus is primarily on Solaris tools, but the techniques of analysis apply to Linux as well.

### CPU Bound

This section shows output in which the SBRC is well utilizing the available CPU and the CPU utilization is spread over many threads (see [Figure 7 on page 65](#)).

**NOTE:** The peak utilization rate is not much higher than the current rate.

Figure 7: CPU-Bound Utilization



## prstat output

PID	USERNAME	SIZE	RSS	STATE	PRI	NICE	TIME	CPU	PROCESS/NLWP
1624	root	244M	129M	cpu6	33	0	686:11:58	88%	radius_generic/252
29853	hadm	181M	141M	sleep	59	0	0:53:12	0.1%	mysqld/18
29766	hadm	131M	26M	sleep	59	0	0:51:51	0.1%	ndb_mgmd/21
2602	root	3808K	3072K	cpu0	49	0	0:00:00	0.0%	prstat/1

```

19423 noaccess 174M 83M sleep 59 0 3:31:37 0.0% java/18
2556 root 7456K 4744K sleep 59 0 0:00:00 0.0% sshd/1
17497 root 3544K 1712K sleep 59 0 0:00:00 0.0% mountd/2
You can determine if any particular thread is taking up too much CPU by prstat -L

```

PID	USERNAME	SIZE	RSS	STATE	PRI	NICE	TIME	CPU	PROCESS/LWPID
1624	root	244M	129M	run	22	0	20:33:47	2.6%	radius_generic/64
1624	root	244M	129M	sleep	33	0	5:07:37	1.7%	radius_generic/82
1624	root	244M	129M	sleep	53	0	2:09:43	1.6%	radius_generic/110
1624	root	244M	129M	run	23	0	3:58:26	1.6%	radius_generic/112
1624	root	244M	129M	run	35	0	2:14:50	1.6%	radius_generic/116
1624	root	244M	129M	run	42	0	11:09:58	1.4%	radius_generic/65
1624	root	244M	129M	sleep	52	0	3:25:00	1.4%	radius_generic/85

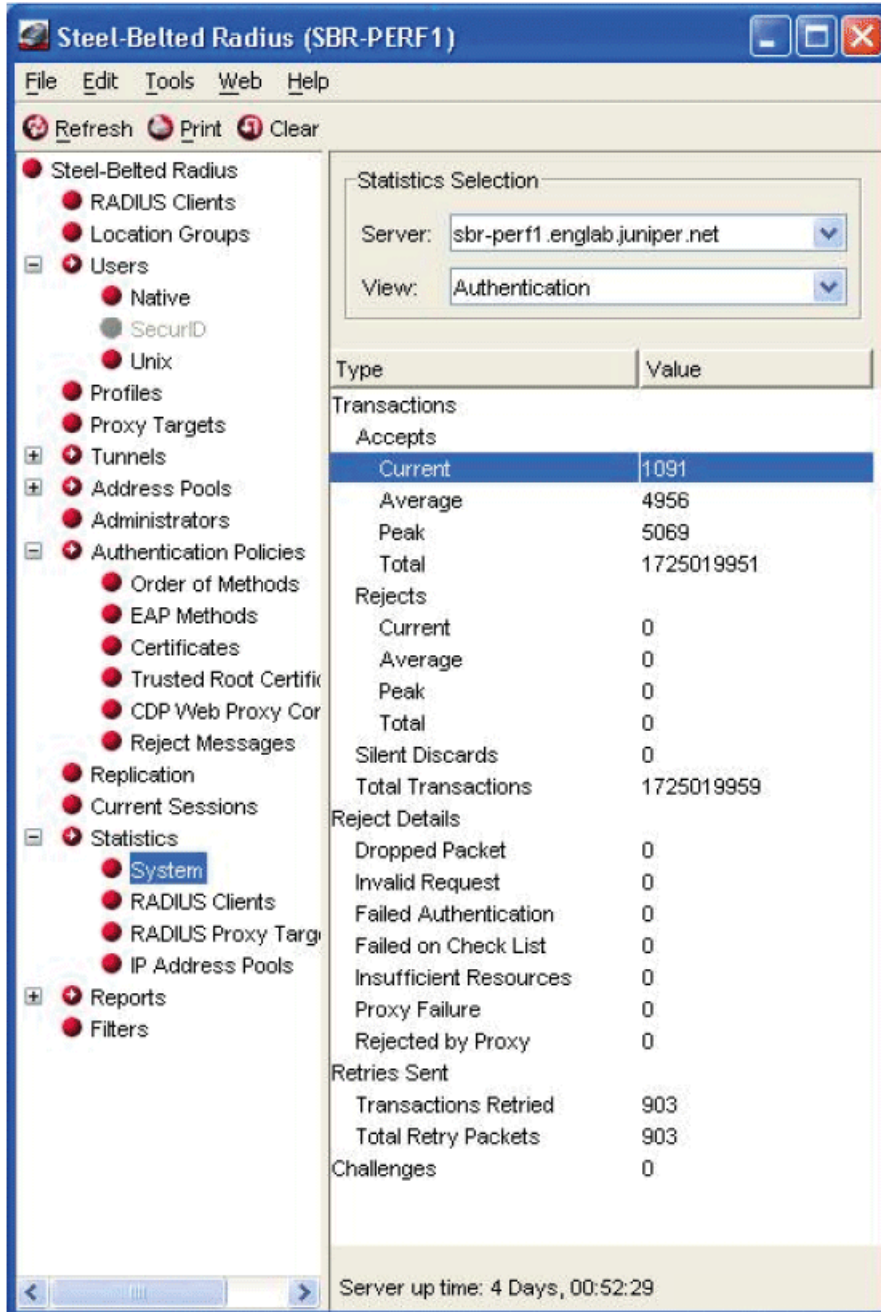
The following output shows no significant I/O usage:

tty		sd0			sd1			sd2			nfs1			cpu			
tin	tout	kps	tps	serv	kps	tps	serv	kps	tps	serv	kps	tps	serv	us	sy	wt	id
0	66	9	1	8	254	16	15	0	0	0	0	0	55	13	4	0	82
0	236	0	0	0	0	0	0	0	0	0	0	0	0	68	23	0	8
0	80	0	0	0	0	0	0	0	0	0	0	0	0	68	24	0	8
0	80	0	0	0	1	1	4	0	0	0	0	0	0	68	23	0	9
0	80	0	0	0	0	0	0	0	0	0	0	0	0	68	24	0	8

## I/O Bound

Figure 8 on page 67 shows output of a system that is more I/O bound (large packets being written to a long accounting file with LogAccept=1, LogLevel=2, TraceLevel=2). In this case, the CPU overhead of I/O is very high, and larger multiprocessor systems with more optimized I/O frameworks (such as Fibre Channel NASD devices and RAID5) will see higher rates and lower CPU utilization if they are I/O bound.

Figure 8: I/O-Bound Utilization



```

PID USERNAME  SIZE  RSS STATE PRI NICE   TIME    CPU PROCESS/NLWP
1624 root       248M 131M sleep  51   0 687:01:29 76% radius_generic/253
29853 hadm      181M 141M sleep  59   0 0:53:16 0.1% mysqld/18
29766 hadm      131M  26M sleep  59   0 0:51:54 0.1% ndb_mgmd/21
19423 noaccess  174M  83M sleep  59   0 3:31:37 0.0% java/18
2693 root       3808K 3072K cpu1   59   0 0:00:00 0.0% prstat/1

```

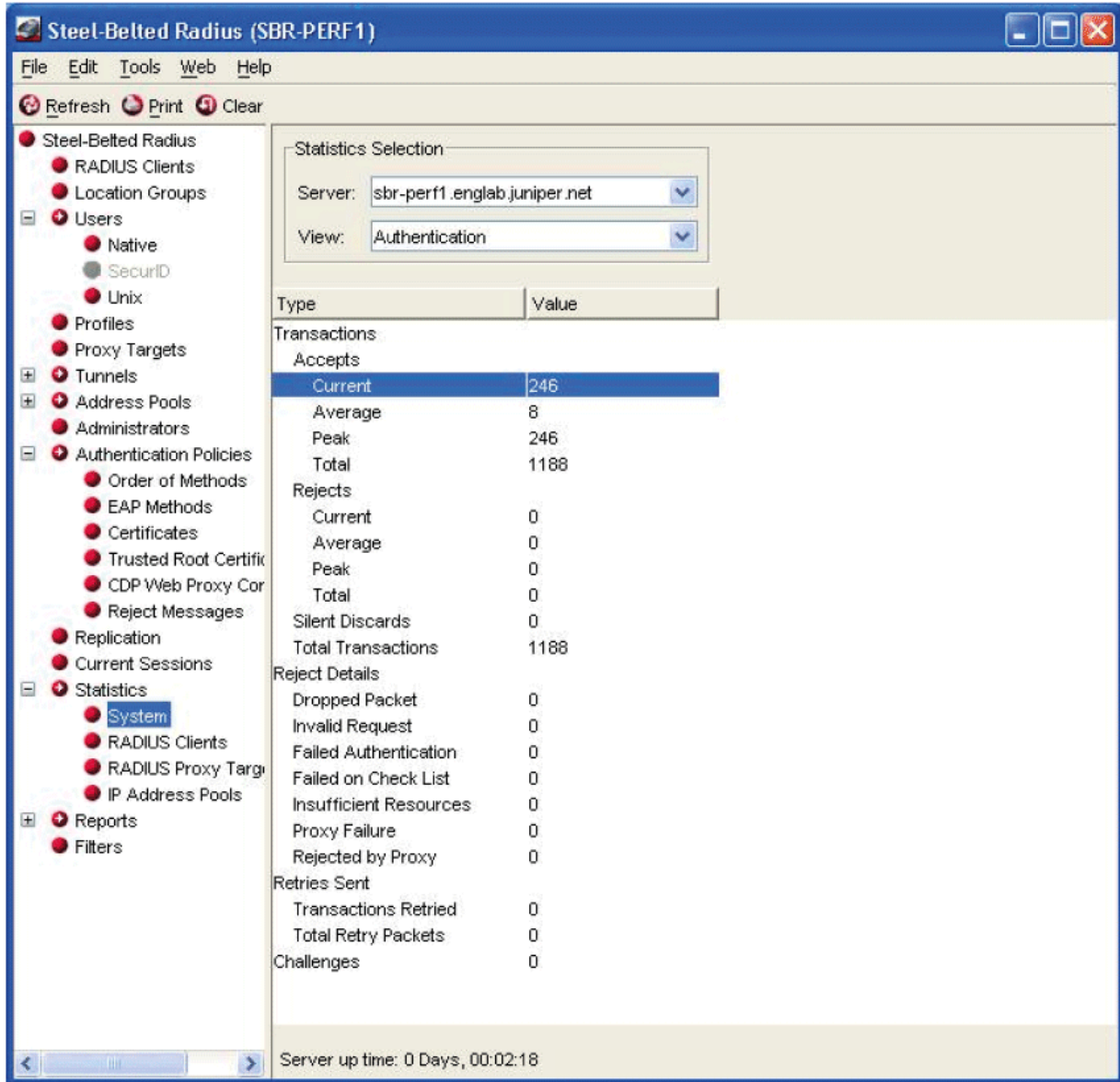
2556	root	7456K	4744K	sleep	59	0	0:00:00	0.0%	sshd/1
2609	root	3456K	2344K	sleep	49	0	0:00:00	0.0%	bash/1
17497	root	3544K	1712K	sleep	59	0	0:00:00	0.0%	mountd/2
17499	daemon	2832K	1624K	sleep	60	-20	0:04:46	0.0%	nfsd/2

tty		sd0			sd1			sd2			nfs1			cpu			
tin	tout	kps	tps	serv	kps	tps	serv	kps	tps	serv	kps	tps	serv	us	sy	wt	id
0	66	9	1	8	254	16	15	0	0	0	0	0	55	13	4	0	82
0	235	0	0	0	16333	17	12	0	0	0	0	0	0	58	24	0	18
0	82	0	0	0	17678	38	13	0	0	0	0	0	0	58	24	0	17
0	82	0	0	0	16693	18	12	0	0	0	0	0	0	58	24	0	18
0	82	0	0	0	16394	17	11	0	0	0	0	0	0	59	24	0	17

Contended Systems

Figure 9 on page 69 is an example of a contended system with many (artificially induced) errors.

Figure 9: Contended Systems Utilization



```

PID USERNAME  SIZE   RSS STATE PRI NICE   TIME    CPU PROCESS/NLWP
 3701 root      199M   83M sleep  59   0    0:00:07  1.3% radius_generic/80
29853 hadm      181M  141M sleep  59   0    0:53:19  0.0% mysqld/18
29766 hadm      131M   26M sleep  59   0    0:51:58  0.0% ndb_mgmd/21
 3717 root      3872K 3232K cpu6   59   0    0:00:00  0.0% prstat/1
 2556 root      7456K 4744K sleep  59   0    0:00:00  0.0% sshd/1

```

```

tty      sd0      sd1      sd2      nfs1      cpu
tin tout kps tps serv kps tps serv kps tps serv kps tps serv us sy wt id
  0   66   9   1   8 254 16 15   0   0   0   0   0   55 13 4 0 82

```

0	236	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	97
0	80	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	98
0	80	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	97
0	80	0	0	0	11	17	20	0	0	0	0	0	0	2	1	0	97

## Notes on Debugging Performance Issues

The following are some notes on debugging performance issues:

- If the amount of time taken to stop and start SBRC is more than two minutes, there might be too many sessions to be sustained through the current I/O bandwidth. You might also need to shard or move to cluster(s), or put the data files on a Fibre Channel storage device or similar devices. Also examine the CPU and I/O utilization at startup for added insight.
- The accounting and proxy thread usage is a function of downstream latency. Many threads will have low CPU utilization, since most of the time is spent waiting. Each thread has some overhead and takes a finite amount of memory. If the system requires more threads, it is trying to take in more packets than the downstream can receive, in which case you should consider if proxy spooling is a better solution.
- Accounting-ON and Accounting-OFF are work-intensive operations, as you can configure SBRC to generate AutoStops to be sent downstream or against the SSR. This generates a great deal of work for SBRC, the SSR, and downstream proxy targets to handle one RADIUS packet's worth of input. Taking this into consideration, the system may hit limits while processing these operations if there are many sessions to be terminated.

See [“Accounting-Off or Accounting-On with AutoStop Enabled” on page 79](#) for more information.

## Special Topics in Performance

### IN THIS CHAPTER

- [Multifactor Authentication over EAP | 71](#)
- [EAP-TLS, TTLS, and WiMAX | 72](#)
- [EAP-SIM/AKA and authGateway | 75](#)
- [External Authentication and Accounting | 76](#)
- [JavaScript | 77](#)
- [Proxy Radius | 77](#)
- [Smart Static Accounting | 78](#)
- [Spooled Accounting | 78](#)
- [IP Pools through the SSR | 79](#)
- [Accounting-Off or Accounting-On with AutoStop Enabled | 79](#)

This chapter helps you understand some of the performance-related topics that affect SBRC.

### Multifactor Authentication over EAP

Multifactor authentication is one of the most secure methods for authentication today. This authentication style is often transported by the Extended Authentication Protocol (EAP), which is embedded in multiple EAP-message attribute value pairs within a RADIUS packet.

Subscriber Identity Module (SIM) is another form of “something you have” authentication, as is Transport Layer Security (TLS), which uses client certificates to authenticate.

Tunneled TLS (TTLS) creates a cryptographically secure channel (either with just a server certificate, or both a client and a server certificate) through which other credentials can be sent (such as passwords or token cards) for greater data security for the single factor authentication, which protects against, for example, man-in-the-middle attacks.

EAP has its own negotiation protocol beneath RADIUS. One EAP negotiation can be encapsulated within many RADIUS packets, depending on the key size, the size of cryptographic material being transported,



or even the number of EAP protocols supported (each protocol allowed by the server can be rejected by the client, requiring multiple round-trips in poorly optimized clients that disregard hints sent by the server). This, combined with the cryptographic complexity of the inner protocol, can severely impact the required amount of CPU utilization to process a single-user request. There are many ways to lessen this impact while remaining cryptographically secure, which will be described in the following sections.

## EAP-TLS, TTLS, and WiMAX

This section explains the following:

- [Reauthentication Cache on page 72](#)
- [Cipher Suites and Key Sizing on page 73](#)
- [Cipher Suite Scaling on page 74](#)
- [Reducing the Number of Round Trips on page 74](#)
- [Certificate Management on page 74](#)
- [Multiple TLS/TTLS-Only Front-End Applications on page 75](#)

### Reauthentication Cache

Instead of doing a full authentication for cryptographically complex processes, session keys can be established and made to persist, and a simpler reauthentication can take place instead. This requires between 200 and 2400 bytes of data per session (which mostly, in the SSR, varies due to stored attribute value pairs) to be stored on both the client and the server. Reauthentication caching optimizes the amount of CPU utilization required and is not a required functionality for any use case. Certain use cases may not see any benefits.

TLS resumption is currently handled on a per-server basis. The cryptographic material is stored on one local SBRC. To perform reauthentication, the source of the previous transaction must be routable (using load balancers or similar, see [“Load Balancing on RADIUS Front-End Applications and Downstream Proxy Devices” on page 48](#) for more information) to the correct target SBRC.

TTLS resumption, especially in the roaming WiMAX use case, can be propagated to the SSR, so that multiple servers can handle the reauthentication. The tradeoff is that heavy cryptography work is not done at the expense of another insert into the SSR for upwards of 2400 bytes of data, plus the read in a future attempted reauthentication. In cases of high throughput, where the performance of the SSR is the bottleneck, turning off reauthentication caching might allow you to scale more linearly (no longer limited by SSR transaction rates) by adding front-end applications.

## Cipher Suites and Key Sizing

The derivation of keying material for a tunnel is completed based on functions of the key (client and server keys for TLS or server certificate for TTLS) by using a set of cryptographic operations called a cipher suite. The default SBRC cipher suite cryptography is estimated on the strength of the suites and occurs in the order listed in [Table 16 on page 73](#).

**Table 16: Cipher Suite**

Cipher Suite	Cipher Suite Code
TLS_RSA_WITH_AES_256_CBC_SHA256	0x003D
TLS_RSA_WITH_AES_128_CBC_SHA256	0x003C
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	0x0067
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	0x006B
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	0x0040
TLS_RSA_WITH_AES_256_CBC_SHA	0x0035
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	0x0039
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	0x0038
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	0x0033
TLS_RSA_WITH_AES_128_CBC_SHA	0x002F
TLS_RSA_WITH_3DES_EDE_CBC_SHA	0x000A
TLS_RSA_WITH_RC4_128_SHA	0x0005
TLS_RSA_WITH_RC4_128_MD5	0x0004
TLS_RSA_WITH_IDEA_CBC_SHA	0x0007

The performance of the above algorithms is loosely related to their relative security, in the order shown in [Table 16 on page 73](#). RADIUS will always attempt to use the highest security cipher suite for which it is configured and that the client can also utilize. Removing the lower cipher suites increases the relative security of your system. Removing the higher cipher suites can result in a significant performance boost.

In the Diffie-Hellman Ephemeral (DHE)-based cipher suites, the temporary keys derived by the client and server are thrown away after the session is completed, providing perfect forward security—nothing directly derived from the server certificate is sent over the wire. However, using DHE keys is expensive and requires the creation of large relative prime numbers. Using the settings of DHE-based suites, the key generation is modified by the number of DHE prime bits of the field in which to search, which is the number of bits to look for in a relatively prime number on which to base the session keys.

### **Cipher Suite Scaling**

A few index values reproduced in Chapter 7 for the M3000 2.53 GHz should scale linearly with total CPU GHz. The relative cryptographic costs will be commensurate between TLS and TTLS. With the default 1024 DHE bits using 0x39, the default cipher suite includes accounting to SSR but not reauthentication caching.

### **Reducing the Number of Round Trips**

SBRC can be configured to use a larger TLS fragment length per RADIUS transaction. Determine the expected length of the non-RADIUS attributes (with some buffer) in an authentication transaction and subtract that from 4000 (the maximum size of a RADIUS transaction). Set the TLS fragment length towards that length, preferably as a number evenly divisible by 255 (maximum RADIUS AVP length). The default of 1020 is intentionally short to emphasize operative safety in cases where many long AVPs might be sent from NAS equipment; 2040 may be a better value for many purposes.

### **Certificate Management**

The cryptographic keys of a TLS-based or TTLS-based server have a strong impact on the amount of work each server must process. A cryptographically secure system is one that has no known breaks or limitations on how long it would take to deconstruct the entire system (for example, in Triple-DES the key field is  $2^{168}$ , but an attack reduces that complexity to  $2^{112}$ , and because of that, cryptographers consider Triple-DES to be “broken”—although not completely so). Keys in such a secure system are cryptographically secure when the amount of time in which they are in use is less than the amount of time it would take to break the key using optimized brute force methods. One way of optimizing the CPU utilization is to generate shorter keys for the server certificate, and change them more often. If you have never changed your server certificates, then decreasing their size and changing them once a year may be both a security and performance improvement.

To give an appropriate reference point, in 2007 it was reported that by using a special number field sieve (filter), the equivalent operation of breaking a single 700-bit RSA key was executed on 400 computers over 11 months.

## Multiple TLS/TTLS-Only Front-End Applications

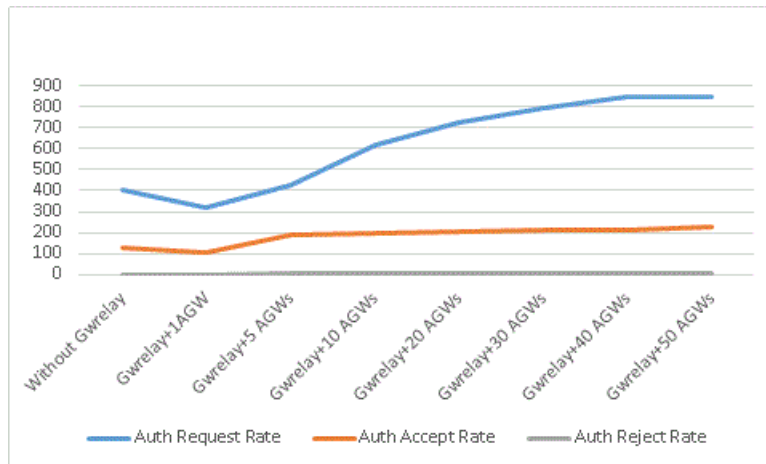
In cases using highly cryptographically secure algorithms and keys, running multiple standalone versions of SBRC, outside of use cases requiring cluster resumption, using TLS with secondary authentication to a cluster front-end application, or using TTLS inner-auth to a cluster front-end application provides linear scaling with hardware. Whether TLS or TTLS--since it all depends on CPU and scales well through higher end, massive multiple-CPU hardware--faster CPU is generally better than more CPU, GHz for GHz, but more total CPU among cores will always be a benefit to this case.

## EAP-SIM/AKA and authGateway

The authGateway functionality is implemented by multiple instances of finite state machines that accept SIM requests from the SBR Carrier simauth plug-in through the GWrelay application, and formats and transmits the requests to the HLR over the SS7 network or SIGTRAN. [Figure 10 on page 75](#) shows statistics of authentication rates with different numbers of authGateway instances on M3000 2.53 GHz.

**NOTE:** The graph is illustrative only, and the results may vary.

**Figure 10: Statistics of Authentication Rates on Solaris**



**NOTE:** These numbers are only for authentication and not for authorization, against an HLR with bearer codes, service type, and so on. We recommend that you use 5 to 10 authGateway instances for better performance results.

## External Authentication and Accounting

Much of the performance of a RADIUS system using external authentication and accounting is related to the latency and throughput of the back end. There is fixed overhead per transaction in terms of CPU utilization, but this should generally scale linearly with total CPU GHz of the system. The overhead for external server latency is accountable to additional memory utilization to keep a copy of the state of the in-process transaction in memory, along with the overhead for threads to manage that state.

The amount of processing increases if multiple external authentication servers must be tried in order. Setting the order so that the most utilized method is attempted first is the easiest way to increase the performance.

Most external authentication and accounting systems accept multiple concurrent connections. Increasing values such as the MaxConcurrent in `radsql.aut/acc` increases the throughput of the system towards the capacity of the back-end system. This value is closely related to the number of authentication or accounting threads that are in use, and there must be more threads than the MaxConcurrent (so that while concurrent transactions are processing on the back-end server, RADIUS-level processing for additional transactions can still be done within SBRC), often up to double or more, depending on latency. Beyond that point, you manage spike traffic using flood queues (see [“Flood Control” on page 51](#) for more information) to avoid having too many attempted concurrent transactions, which might cause the downstream server to stop processing efficiently.

LDAP, even more than SQL, can have poor performance with badly constructed searches (which might cause an unnecessary object tree walk) and filters, or with the use of functionality such as `BindName` (although the latter is sometimes necessary in some use cases of non-reversibly encrypted passwords with certain types of authentication protocols). Having multiple open connections to an LDAP repository can send more simultaneous requests.

Storing less data is usually better than storing more data. Selects, updates, and deletes by primary key are generally faster than by secondary indexes, which in turn are much faster than by non-indexed fields. Inserts into tables with many indexes outside the primary key are slower than inserts into tables with one or few indexes. Optimizing your back end to avoid extraneous indexes, querying regularly only on items that are necessary, and using well-optimized primary keys can produce a large throughput increase and latency decrease.

Other SQL items worth investigating include validating whether stored procedures are optimized and having no hidden table scans or other iterative behavior that could be optimized out with a different data model. In general terms, raw inserts into one table that might be subsequently batch-processed outside of the transaction path (by an external job) are faster than manipulated inserts into many tables within a stored procedure; however, the ability to do batch-processing depends greatly on the use case.

Measuring the performance of common inserts and selects with SQL analyzers and keeping tabs on the performance of back-end servers in terms of CPU and I/O (as recommended for SBRC) will help determine scalability issues with back-end servers, and can help reduce the extraneous overhead on the front-end applications as well.

Finally, load balancing between multiple back-end servers, as well as enabling failover behavior, can reduce performance issues caused by back-end throughput in a relatively linear manner.

See the *SBR Carrier Reference Guide* for information on configuring the following parameters:

- *Max-Auth-Threads*
- *Max-Acct-Threads*
- *MaxConcurrent* (radsql.aut and .acc)
- SQL load balancing
- *Ldapauth.aut*

## JavaScript

The number of JavaScript engines used at any one time is controlled by the **MaxEngines** setting in the **radius.ini** file. This need not be more than the number of total threads that will be running engines. See the *SBR Carrier Reference Guide* for information on configuring the **MaxEngines** parameter.

**NOTE:** Juniper Networks is not responsible for decreases in system performance that might have resulted due to the customization of JavaScripts implemented by the customer. Good programming can alleviate.

## Proxy Radius

Similar to external authentication and accounting, the fixed overhead per proxied transaction is approximately the value of one regular request/response pair, either authentication or accounting. This is because it takes about the same amount of time to parse the incoming RADIUS packet and generate a response as it does to generate a downstream packet and parse the response.

When the block is set as 0 in the .pro file [Acct] section, SBRC returns an ACK for the proxy accounting transactions and then passes the transaction state to a proxy thread (which is controlled by *Max-Proxy-Threads*). These functions are similar to *MaxConcurrent*. Unlike the relationship between *Max-Acct-Threads* and *MaxConcurrent* for external accounting, when it goes to proxy downstream, a RADIUS transaction is subsequently handled by the *Max-Proxy-Thread* pool for the response. So, the *Acct Thread* hands off the work to the *Proxy Thread* entirely, and the initial thread is free to handle another request. In general, the proxy threads may be larger than the *Max-Acct-Threads*, unless the external

latency of the back end is higher than the external latency of the downstream, in which case they should be about equal.

When the block is not set to 0, the authentication or accounting thread handles everything and waits for the proxied response, and the number of threads must be related to the latency of the downstream (unless spooled accounting is used). See the *SBR Carrier Reference Guide* for information on configuring the Max-Proxy-Threads parameter.

## Smart Static Accounting

With smart static accounting, accounting requests can be sent to multiple downstream servers. As with the basic proxy case, the amount of work required for each request will also approximately equal one request/response pair. For example, with 10 downstream servers:

- With Block=0, proxying to 10 static accounting servers downstream occurs at the same time, up to the limit of proxy threads, which may be ten times the number of accounting threads to sustain the throughput (to the limits of memory).
- With Block=1, proxying to 10 static accountings downstream is serial, and has ten times the latency.

In both cases, the amount of CPU per transaction from downstream is ten times since the same work must be done, effectively ten times, with different encryption (for shared secrets).

## Spooled Accounting

When the throughput of the SBRC system might temporarily be greater than the throughput of the downstream, and you want reliable transport so that the accounting gets downstream eventually, you can use the spooled accounting mechanism. In this efficient method, the packets are sent downstream synchronously about as fast as the downstream can go plus latency, but are sent outside of the transaction path. The overhead is in additional I/O to the local file system, which is approximately double the same throughput used for the equivalent network operation. For a 2000 RADIUS transaction, 2000 is written to the file, then later the file is read and the 2000 is written to the network. Currently, spooled accounting logs are drained by one thread per realm and have one in-flight transaction outstanding. Contact the Juniper Network Sales Engineer or manager for additional throughput of multiple in-flight transactions or multiple threads per realm.

If the average performance requirements of the system are greater than the downstream realm can handle, the backlog can be significant, and delays in draining the local spool to the downstream proxy can reach a point where the downstream will never catch up. See the *SBR Carrier Reference Guide*, [SpooledAccounting] section, for more information.

## IP Pools through the SSR

Because of the (well-known) problems in the ability of a SQL-derived system to implement efficient, reliable queuing across different processes, the current implementation of SSR has a performance limit with recovering addresses from very large IP pools on multiple threads. Multiple threads (running across multiple SBRC instances) attempting to pre-fetch possible lock-up keys to reclaim unused IP addresses can contend for the rows. This is a form of lock contention that can interfere with continued high-throughput SBRC operations. The semantics for lock back-off are controlled by the value for `TransactionDeadlockDetectionTimeout` in the `config.ini` file. One thread can wait up to this timeout value before retrying the reclaim operation, leading to a serious throughput limit.

There are several ways of managing this contention.

- Use different IP pools for different NAS devices, with different NAS devices biasing themselves (or being biased by a load balancer) to a given SBRC front-end application. This reduces cross-front-end contention.
- Shard the users or profiles so that different users have a *Framed-IP-Address* set from different pools. This evenly spreads the contention so that the usual case of random sleep used to manage contention (configured in `dbclusterndb.gen` as a period between `CacheThreadSleepMin` and `CacheThreadSleepMax`) will be sufficient to ensure adequate back-off for a request to reclaim a chunk of IP addresses. The added benefit is you can easily identify such items as class of service by linking given addresses to certain pools with well-defined ranges.
- Have enough addresses to set the `CacheLowWater` and `CacheHighWater` marks in `dbclusterndb.gen` very high, so that contended operations do not impact the effective throughput (so that you can take a multi-hundred millisecond pause in reclaiming addresses without any transaction errors); however, values over 20,000 may cause SBRC to take longer to shut down in order to return the cached addresses to the available state. Also, a `CacheHighWater` value near the available size of the pool can cause one SBRC to cache all available addresses, leaving other SBRCs with none, which will lead to incorrectly failed authentications.

## Accounting-Off or Accounting-On with AutoStop Enabled

One of the few cases where a single RADIUS transaction can turn into an excessive amount of work on the back end is in processing an Accounting-Off or Accounting-On from a NAS device with many current sessions active. In this case, an accounting stop record is generated for each session. With high-end GGSN and similar devices, which can serve 100,000 users or more, the amount of work the SBRC must do and the number of operations made against an SSR are exceedingly high. In any SLA, this should be considered to be a “failure recovery” case, with the agreement that any transactions being processed during this time are processed on a best-effort basis.



Forwarding Accounting-Off and Accounting-On transactions to a server dedicated to handle such overflow operations can provide an optimal solution to avoid unnecessary user-level outage while the work is being executed.

**NOTE:** In general, Accounting-ON or Accounting-OFF typically is processed in one thread per transaction, reliably at approximately 1000 sessions per second on the highest single-CPU performing platforms (3–3.6 GHz). Retries of Accounting-ON or Accounting-OFF must be set so that the same SBR is tried multiple times for the length of the retry period required to terminate the sessions. For instance, if there are 20,000 sessions on a given NAS, the total retry length for the Accounting-ON or Accounting-OFF should be more than 20 seconds. Trying a second SBR in less than 20 seconds results only in duplicate work attempts, which results in conflicts and decreases the total throughput.

If multiple NAS devices may start in tandem—for instance, they are on the same UPS—load balancing each device with different primary and secondary starting points helps balance out the initial spike of work from processing an Accounting-ON.

# SBRC System-of-Systems Performance Reference

## IN THIS CHAPTER

- [Hardware Used | 81](#)
- [CPU Utilization | 82](#)
- [Test Results Summary | 82](#)

This chapter provides performance summary results of tests to estimate the SBRC load in a given use case or collection of use cases on a given system-of-systems. You can break down the effects of the tests and use them as a baseline to make reasonable estimates of expected performance. The tests represented in this chapter are not exhaustive, but are intended to give an indication of scalability.

This chapter also covers the limits of hardware performance. For example, disk I/O is rarely a limit for performance other than local file accounting or logging, and is ignored in most cases. This chapter focuses primarily on the following key performance measures:

- Overall CPU utilization (most relevant for SBRC)
- Per-thread CPU utilization (most relevant for SSR D nodes), with a focus on the single-highest thread

Throughout this chapter, the results are preceded by an explanatory text. Front-end application is an instance of SBRC and back end usually refers to a D node running `ndbmt`.

## Hardware Used

The results shown throughout this chapter were tested on the following hardware:

- Two M3000 front-end applications with CPU 2.75 GHz, 16 G, running SBRC Release 7.4.0.
- Two or four M3000 back ends with CPU 2.52 GHz, 16 G, with half of the virtual CPUs disabled (unless otherwise specified) through `psradm`, running MySQL `ndbmt`
- M4000 Cluster, M9000 Standalone, and T3 standalone.

The focus primarily is on M3000 servers, as they are the recommended platform to run all the requisite tests.

The test clients used were executed with custom developed software, LoadTest, running between one and four VMs, executing on a 3.33 GHz XEON with 24 G.

**NOTE:** LogLevel and TraceLevel are set to zero and the local accounting is disabled unless otherwise specified.

## CPU Utilization

In most cases, the back-end CPU utilization percentage is a single percentage number, representing the highest single thread utilization, with the other threads being under control. In a few cases, the back-end CPU utilization percentage has multiple numbers (see [Table 17 on page 82](#)), which represents the order of thread execution—the top two (or four in the case of eight Execute Thread tests) are Local Query Handler (LQH) threads, which execute a major part of the work.

Table 17: CPU Utilization

CPS	Front-End CPU Utilization	SSR Node Max Single Thread Utilization
4300	78 %	2.3 %, 2.1 %, 1.2 %, 1.1 %

## Test Results Summary

### IN THIS SECTION

- [Pap Authentication, Accounting Start, Accounting Stop | 83](#)
- [Pap Authentication, Accounting Start, Accounting Stop—Sessions Preloaded | 84](#)
- [Accounting Only—Sessions Preloaded, One Start and One Stop, Four D Nodes | 84](#)
- [Accounting Only—Sessions Preloaded, One Start and One Stop, Two D Nodes | 84](#)
- [Standalone: Auth/start/stop CPS | 85](#)
- [Standalone: Accounting for 200 Threads | 85](#)
- [Standalone: Authentication Only | 86](#)
- [LDAP Authentication Only | 86](#)
- [Oracle 11g | 87](#)
- [LCI Queries against the SSR through SBRC | 87](#)

- IP Address Allocation | 88
- IP Address Allocation with Eight Threads, Two D Nodes | 88
- IP Address Allocation with Eight Threads, Four D Nodes | 89
- TTLS | 89
- TTLS Plus Storing Resumption Context | 90
- WiMAX | 91
- 4D node system: M5000 | 91
- Standalone: M9000 | 91
- Standalone: T3 | 92
- SSR and Standalone Performance: E4870 and X5687 CPUs | 93

This section provides a summary of the results tested in a given set of systems.

**NOTE:** All of the results shown in this section are for one front-end application unless otherwise stated.

### Pap Authentication, Accounting Start, Accounting Stop

Tested on four D nodes with zero sessions preloaded.

**NOTE:** The number of test client threads is varied to show the effect of simultaneous operations.

**Table 18: Pap Authentication, Accounting Start, Accounting Stop**

Client Test Threads	CPS	Front-End Utilization	SSR Node Single Thread Utilization
100	4290	91 %	3.8 %
80	4252	91 %	3.8 %
120	4239	90 %	3.8 %

### Pap Authentication, Accounting Start, Accounting Stop—Sessions Preloaded

Tested on four D nodes with sessions preloaded.

**Table 19: Pap Authentication, Accounting Start, Accounting Stop—Sessions Preloaded**

Number of Sessions Preloaded	CPS	Front-End Utilization	SSR Node Thread Utilization
1M	4210	89 %	5.7-6.1 % <sup>1</sup>
4M	4185	91 %	6.5-6.8 %
7M	4064	89 %	7.2-7.4 %

<sup>1</sup>CPU utilization varied with Local and Global Checkpoints (LCP and GCP). Higher figures for other results are shown below.

### Accounting Only—Sessions Preloaded, One Start and One Stop, Four D Nodes

Tested on four D nodes with sessions preloaded.

**Table 20: Accounting Only—Sessions Preloaded, One Start and One Stop, Four D Nodes**

Number of Sessions Preloaded	Accts/Sec	SSR Node Single Thread Utilization
0	11,900	
4M	11,446	
7M/100 threads	11,070	8.7 %
7M/120 threads	11,160	8.8 %

### Accounting Only—Sessions Preloaded, One Start and One Stop, Two D Nodes

Tested on two D nodes with sessions preloaded, variation of accounting to local file.

**Table 21: Accounting Only—Sessions Preloaded, One Start and One Stop, Two D Nodes**

Number of Sessions Preloaded	Accts/Sec	SSR Node Single Thread Utilization
0 sessions/no account logging	13,900	4.2 %

Table 21: Accounting Only—Sessions Preloaded, One Start and One Stop, Two D Nodes (*continued*)

Number of Sessions Preloaded	Accts/Sec	SSR Node Single Thread Utilization
0 sessions/default account logging	12,280	4.4 %
1M sessions/default account logging	12,340	6.6 %
4M sessions/default account logging	12,300	7.5 %
4M sessions/minimal account logging	12,660	8.0 %
4M sessions/no account logging	13,740	8.3 %
7M sessions/no account logging	13,400	8.6 %

**Standalone: Auth/start/stop CPS**

Tested on standalone SBR for Auth/start/stop CPS.

Table 22: Standalone: Auth/start/stop CPS

Case	CPS	CPU Utilization
0 Sessions	6330	97 %
1M sessions	5930	96 %
3M sessions	5760	95 %

**Standalone: Accounting for 200 Threads**

Tested on standalone SBR for accounting only for 200 threads.

**NOTE:** Standalone being a 32-bit application, is limited to a 4 GB working set.

Table 23: Standalone: Accounting for 200 Threads

Case	Accts/s	CPU Utilization	Disk I/O
1M sessions	22,500	95 %	1.8 M/s

Table 23: Standalone: Accounting for 200 Threads (*continued*)

Case	Accts/s	CPU Utilization	Disk I/O
3M sessions  (This maximum varies with each use case, depending on data stored in the SSR compared to the 4G limit.)	20,300	95 %	2.1 M/s
3M Sessions/Account.ini Enabled	15,200	96 %	6.0 M/s

### Standalone: Authentication Only

Tested on standalone SBR for authentication only.

Table 24: Standalone: Authentication Only

Case	Auths/s	CPU Utilization	Disk I/O
PAP	14,800	98 %	–
CHAP	14,760	98 %	–
Rejects/s	~22,000	98 %	–
LogAccept=1	13,170	96 %	1.2 M/s
LogLevel=2	3,890	77 %	6.8 M/s
LogLevel=1	9,590	86 %	3.6 M/s

### LDAP Authentication Only

Tested for LDAP authentication.

Table 25: Authentication Only

Case	Auths/Sec	Front-End Overall	Front-End Single-Thread
1 client threads, maxconnect = 1	454	11 %	3.4 %
10 client threads, maxconnect = 1	1030	26 %	6.2 %

**Table 25: Authentication Only (continued)**

Case	Auths/Sec	Front-End Overall	Front-End Single-Thread
100 client threads/ 10 [server/*] sections	10,175	39 % <sup>2</sup>	— <sup>3</sup>

<sup>2</sup> CPU on LDAP server maxed out—running 166 percent of two CPU VMware at 3.3 GHz.

<sup>3</sup> Evenly split, below the transport threads level.

## Oracle 11g

Tested on an Oracle 11g database running on an M3000, 2.52 GHz.

**Table 26: Oracle 11g**

Case	Auths/Sec or CPS	Front-End %	Oracle DB Process Max %
Auth only—10 client threads	9200 auths/sec	24 %	22 % overall
Auth only—15 client threads	12,500 auths/sec	37 %	30 % overall
Directed realms, 20 client threads over two instances	17,000–24,000 <sup>4</sup> auths/sec	71 %	70 %
Auth/start/stop, MaxConnections = 1	2610 CPS	37 %	4.6 %
Auth/start/stop, MaxConnections = 10	4255 CPS	91 %	6.9 % (spread over 8 Oracle processes)

<sup>4</sup> This result varied widely due to latency of the single Oracle DB processing requests from too many simultaneous connections.

## LCI Queries against the SSR through SBRC

The number of LCI server threads is hard-limited to eight by the server to avoid interference with regular RADIUS traffic processing.<sup>5</sup>

**Table 27: LCI Queries against the SSR through SBRC**

Case	LCI Queries/Sec	Front-End CPU	SSR Node CPU
100 query client threads	1452	53 %	3.7 %



Table 27: LCI Queries against the SSR through SBRC (continued)

Case	LCI Queries/Sec	Front-End CPU	SSR Node CPU
200 query client threads	1716	62 %	4.4 %

<sup>5</sup> SBRC 7.4.0 performance improvement permits greater throughput at less front-end CPU utilization.

## IP Address Allocation

Tested on two D nodes with IP address allocation.

Table 28: IP Address Allocation with Two D Nodes

Case	CPS	SSR Node CPU
1M sessions, 1 pool, 1 front end	3050	18 %, 18 %, 5.2 %, 3.5 %
1M sessions, sharing 1 pool, 2 front ends	1600 + 1550 = 3150	19 %, 18 %, 5.3 %, 3.5 %
1M sessions, 2 pools, each front end using its own	1885 + 1857 = 3742	22 %, 20 %, 5.1 %, 3.5 %
1M sessions, LCI queries, 1 front end	2650 + 550 LCI TPS	17 %
1M sessions, LCI queries, 2 front ends	3200 + 750 LCI TPS	22 %

## IP Address Allocation with Eight Threads, Two D Nodes

Tested on two D nodes with eight execute threads on eight virtual processors. This test case gives an indication of performance on the M4000. In this case, two threads are running on each physical processor for the M3000 case; it will be one thread per physical processor for the M4000.

Table 29: IP Address Allocation with Two D Nodes and Eight Execute Threads

Case	CPS	Max Single Thread (out of 12.5 %) SSR Node	Front-End CPU
0 sessions, 1 front end	3635	5.3, 5.1, 5.1, 5.1, 3.3, 2.0	75 %
0 sessions, 2 front ends, 2 pools	3270 + 3390 = 6660	8.3, 8.2, 8.2, 8.2, 4.0, 2.0	68 %
1M sessions, 2 front ends, 2 pools	2509 + 2492 = 5001	10.0, 9.7, 9.7, 9.6, 3.4, 1.9	
1M sessions, 2 front ends, 1 pool <sup>6</sup>	1853 + 1875 = 3728	8.4, 8.3, 8.0, 8.0, 3.3, 1.9	

<sup>6</sup> In the two front end applications, one pool case, the limiting factor is the ndb lock collision associated with multiple threads reaping the old IP address at the same time.

### IP Address Allocation with Eight Threads, Four D Nodes

Tested on four D nodes with eight execute threads on eight virtual processors. This test case gives an indication of performance on the M4000. In this case, two threads are running on each physical processor.

**Table 30: IP Address Allocation with Four D Nodes and Eight Execute Threads**

Case	CPS	Max Single Thread (out of 12.5 %) SSR Node	Front-End CPU
No IP, 1M sessions, 2 front ends  ("No IP" is used as a baseline reference)	$5514 + 5472 = 10,986$ <sup>7</sup>	3.8, 3.8, 3.2, 3.0, 2.9, 1.7	
1M sessions, 1 front end	3650	4.0, 3.9, 3.7, 3.7, 2.2, 1.9	74 %
1M sessions, 2 front ends, 2 pools	$2550 + 2600 = 5150$	5.4, 5.3, 5.1, 5.1, 2.7, 2.1	45 %
1M sessions, 2 front ends, 1 pool	$2300 + 2400 = 4700$	3.9, 3.9, 3.8, 3.7, 2.0, 1.6	35 %

<sup>7</sup> 10 percent variation due to GCP and LCP.

### TTLS

TTLS with five RADIUS auth/challenge pairs per accept using various cipher suites and dh key sizes.

**Table 31: TTLS with Five RADIUS Auth/Challenge Pairs per Accept**

Case	Accepts/Second	Front-End %
1024 dh bits, 0x39	565	95 %
512 dh bits, 0x39	871	95 %
1536 dh bits, 0x39	$256$ <sup>8</sup>	73 %
0x38	574	(Maximum, around 95 %)
0x33	575	(Maximum, around 95 %)
0x32	578	(Maximum, around 95 %)

Table 31: TTLS with Five RADIUS Auth/Challenge Pairs per Accept (*continued*)

Case	Accepts/Second	Front-End %
0x16	575	(Maximum, around 95 %)
0x13	872	(Maximum, around 95 %)
0x66	886	(Maximum, around 95 %)
0x35	890	(Maximum, around 95 %)
0x2f	888	(Maximum, around 95 %)
0x15	575	(Maximum, around 95 %)
0x12	1112	(Maximum, around 95 %)
0x0a	1116	(Maximum, around 95 %)
0x05	1114	(Maximum, around 95 %)
0x04	1120	(Maximum, around 95 %)
0x07	1116	(Maximum, around 95 %)
0x09	1117	(Maximum, around 95 %)

<sup>8</sup> The clients tested were out of CPU.

## TTLS Plus Storing Resumption Context

Tested with two D nodes on TTLS plus storing resumption context storage overhead.

Table 32: TTLS Plus Storing Resumption Context

Case	Accepts/Second	NDB SSR Node Thread Utilization
Resumption (0x38)	575	0.7, 0.7, 0.4, 0.4
Resumption (0x09)	1117	0.9, 0.8, 0.5, 0.5

## WiMAX

WiMAX tested with TTLS plus one HA plus two starts and two stops. 10 RADIUS transactions = 6 NDB hits.

**Table 33: WiMAX**

Case	Accepts/Second	NDB SSR Node Thread Utilization	Front-End %
WiMAX (0x38)	355	3.5, 3.5, 1.3, 1.1	91 %
WiMAX (0x09)	475	4.7, 4.7, 1.6, 1.4	98 %

### 4D node system: M5000

These results were tested on the M5000 with four CPUs (virtual CPUs disabled) at 2.66 GHz and switch-connected with 10G networking.

Simple accountings per second = 49,200 (million per row stored):

- Network bandwidth (NB) for D nodes recorded upwards of 42 MBps = 336 Mbps
- Disk bandwidth 5 MBps = 40 Mbps

Realistic accountings per second (7M rows preloaded) and more data per transaction (approximately 512b per row stored):

- Accountings per second = 23,240
- Bandwidth (up to 7 MBps) = 56 Mbps
- Network bandwidth = 80 MBps

IP address allocations: 7701 CPS

WiMAX with resumptions: 3300 CPS

### Standalone: M9000

These results were tested on M9000 (16 CPUs x 8 cores) at 3.0 GHz running in one global zone, or six non-global zones.

**NOTE:** When SBR is running in non-global zones, the performance is limited because it is unable to set the RealTime priority on the receiving threads.

- Maximum authentications per second for one global zone = 31,000
- Maximum accountings per second for one global zone (with logging) = 38,000
- Maximum authentications per second across six non-global zones = 105,804
- Maximum accountings per second across six non-global zones = 91,470
- CSPA (without logging) across six non-global zones = 59,252
- TTLS (512RSA):
  - Maximums TTLS per second for one global zone = 3900
  - Maximums TTLS per second for six non-global zones = 11,900

### Standalone: T3

These results were tested on T3 (4 CPUs x 128 cores) at 1.65 GHz running in six non-global zones.

**NOTE:** Network bandwidth—disabling virtual CPUs decreases performance in all cases.

- Maximum authentications per second across six non-global zones = 69,850. Total CPU utilization is 18%.
- Maximum accountings per second across six non-global zones: 18,800. Total CPU utilization is 15.6%.

**NOTE:** Accounting and local session performance is excessively limited by single-CPU speed and spindle I/O performance. Consequently, T3 is not recommended for accounting performance, but has high ROI for TLS/TTLS and WiMAX cases.

- TTLS (512 DHE)—Maximum TTLS per second across six non-global zones = 12,100.  
Total CPU utilization is 78% and represents maximum server utilization available on this server.
- TTLS (1024RSA)—Maximum TTLS per second across six non-global zones = 17,000.  
Total CPU utilization is 66%.
- TTLS (1024 DHE)—Maximum TTLS per second across six non-global zones = 6300.  
Total CPU utilization is 52%.
- TTLS (2048 DHE)—Maximum TTLS per second across six non-global-zones = 2046.  
Total CPU utilization is 43.2%.

## SSR and Standalone Performance: E4870 and X5687 CPUs

These results were tested on E4870 CPUs and X5687 CPUs.

**Table 34: SSR Performance**

Case	6D - E7-4870 x 1 CPU	4D - E7-4870 x 1 CPU	2D - E7-4870 x 1 CPU	2D X5687 x 2 CPU
Accountings per second	137,900	97,000 *	78,100	89,000
Accountings per second 512 byte sessions, 15M preload	65,600	47,700	24,400	–
SBR 7.4.1 with new NDB optimizations	140,700	120,000 *	107,000	107,000

**Table 35: Standalone Performance**

Case	X5687 x 2 CPUs	E7-4870 x 1 CPU (2 for TTLS)	E7-4870 x 8 CPUs (with 6 to 12 Virtual machines)
Pap Authentications per second	44,900	48,000	179,000
Accountings per second (CST and logging to disk)	42,000	40,000 *	190,000
TTLS 512 DHE (0x39) authentications per second	3,800	4,300	–
TTLS 1024 RSA (0x2F) authentications per second	4,700	5,400	11,600
TTLS 1024 DHE (0x39) authentications per second	2,900	3,000	–

**NOTE:** \* –These numbers are based on estimation.

Proxy authentications per second, accountings per second = 22,000. Round-robin is used for multiple downstreams.

JDBC/MySQL authentication = 18,800. Total CPU utilization is 15.6%.

JavaScript Engine Overhead = 10%.

# Performance Metrics for 64-Bit Standalone SBR Carrier

IN THIS CHAPTER

- Components of a Sample Use Case | 95
- Test Results Summary of Standalone SBR Carrier | 96
- Test Result Summary of Standalone Running on KVM Hypervisor | 101

This chapter provides performance summary results of tests run to analyze the performance of the 64-bit version of SBR Carrier in a given use case or collection of use cases on a given set of systems.

**NOTE:** The performance results shown in this chapter are obtained based on our lab environment, and the results may vary depending on your environment.

## Components of a Sample Use Case

Each use case can be considered as a collection of work done by the components of 64-bit version of standalone SBR Carrier, in terms of CPU, memory, and total transactions.

Table 36 on page 96 shows the breakdown of use case tasks done by the SBR Carrier components.



Table 36: Use Cases Performed on a Standalone SBR Carrier

Task Execution Sequence	Subtasks	Description
Native	<ol style="list-style-type: none"> <li>1. SBR Carrier validates user credentials against the native user database.</li> <li>2. Local accounting is used for all accounting transactions.</li> </ol>	<p>SBR Carrier performs native user transactions under the following load scenarios:</p> <ul style="list-style-type: none"> <li>• PAP Authentication Only</li> <li>• Accounting Start Only</li> <li>• PAP Authentication, Accounting Start and Accounting Stop</li> </ul>
SQL	<ol style="list-style-type: none"> <li>1. SBR Carrier validates user credentials against a SQL database.</li> <li>2. SBR Carrier returns Accept/Reject based on the response received from the SQL database server (any SQL database supported by JDBC) database.</li> <li>3. Local accounting (not SQL accounting) is used for all accounting transactions.</li> </ol>	<p>SBR Carrier performs SQL user transactions under the following load scenarios:</p> <ul style="list-style-type: none"> <li>• MySQL Authentication Only</li> <li>• MySQL Authentication, Accounting Start and Accounting Stop</li> </ul> <p><b>NOTE:</b> Session information is stored locally (.hst file).</p>
LDAP	<ol style="list-style-type: none"> <li>1. SBR Carrier validates user credentials against an LDAP database using BindName authentication.</li> <li>2. SBR Carrier returns Accept/Reject based on the query response received from the LDAP database.</li> <li>3. Local accounting is used for all accounting transactions.</li> </ol>	<p>SBR Carrier performs LDAP user transactions under the following load scenarios:</p> <ul style="list-style-type: none"> <li>• LDAP Authentication Only</li> <li>• LDAP Authentication, Accounting Start and Accounting Stop</li> </ul> <p><b>NOTE:</b> Session information is stored locally, as LDAP is used for authentication only.</p>

## Test Results Summary of Standalone SBR Carrier

This section provides performance summary results of tests to estimate the maximum SBR Carrier load in a given use case or collection of use cases on a given set of systems. You can break down the effects of the tests and use them as a baseline to make reasonable estimates of expected performance. The tests represented in this section are not exhaustive, but are intended to give an indication of scalability.

**NOTE:** In most cases, excessive load was used to determine the maximum capacity of SBR Carrier, so the test results are not representative of actual scenarios.

The results shown in this section are tested on the following hardware running on RHEL 7.2:

- HP ProLiant DL380 Gen 9 front-end server with 32 CPUs with 8 cores per CPU, 2400 MHz, 128 GB RAM, running SBR Carrier Release 8.4.0.

**NOTE:** The results shown in this section are obtained in an environment where very few applications or services are left running during the tests.

**NOTE:** The maximum CPU utilization measured for a given process is equal to 100% times the number of virtual CPUs active on the system. In these cases, the maximum CPU utilization would be 3200%, since there are 32 CPUs. Therefore, the percentage of the total processing power of a physical machine can be calculated by dividing this measurement by the number of CPUs in the machine. Since all CPU utilization is less than 3200% and evenly distributed across multiple threads, only 81% of the total CPU on the system is utilized in these tests. In this case, SBR Carrier works well under the load.

## Native User Authentication and Logging

Table 37 on page 97 describes the results of tests run on the 64-bit version of standalone SBR Carrier (with 32 CPUs x 8 cores) for native user authentication using PAP and **LogLevel=0** (the default).

**NOTE:** The native user subscriber database was used to eliminate network and back-end effects and is not typically used as a subscriber database in production.

Table 37: Native User Authentication and Logging

Case	Authentication Thread Count	CPU Utilization	Auth-Accepts/Sec	Disk I/O
LogAccept=0	100	996%	42,787	–
LogAccept=1	100	1121%	43,500	2.40 M/s

## LDAP Authentication and Thread Configuration

Table 38 on page 98 and Table 39 on page 98 describe the results of tests run on the 64-bit version of standalone SBR Carrier (with 32 CPUs x 8 cores) for LDAP authentication (OpenLDAP) using BindName.

Increasing the authentication thread count along with TPS results in increased performance as shown in Table 38 on page 98. See Table 38 on page 98 for the recommended authentication thread configuration based on the TPS.

**Table 38: LDAP Authentication Only Without Latency**

TPS	Authentication Thread Count	CPU Utilization	MaxConcurrent in Idapauth.aut File	Number of Server Sections (round-robin fashion) <sup>1</sup>
5000	100	121%	100	10
10,000	100	230%	100	10
15,000	200	498%	100	10
20,000	500	2490%	500	20

<sup>1</sup>See our Knowledge Base

<https://kb.juniper.net/InfoCenter/index?page=content&id=KB27024&actp=METADATA> for more information on how the target host is determined when multiple server sections are configured in the LDAP and SQL authentication files.

With an increase in the backend latency for LDAP authentication-only load (see Table 39 on page 98), the transaction rate is reduced even though more authentication threads are configured. This will impact the performance of SBR Carrier.

**Table 39: LDAP Authentication Only With Backend Latency**

Latency	TPS <sup>2</sup>	Authentication Thread Count <sup>3</sup>	CPU Utilization	MaxConcurrent in Idapauth.aut File	Number of Server Sections (round-robin fashion)
1 millisecond	6500	5,000	481%	100	20
3 milliseconds	2840	10,000	198%	100	20
5 milliseconds	1812	10,000	124%	100	20

<sup>2</sup>TPS = Total accepts / 600 seconds

<sup>3</sup>More threads are required with higher latency because each thread holds the transaction state until a response is received or the request times out.

## SQL Authentication Only

[Table 40 on page 99](#) and [Table 41 on page 99](#) describe the results of tests run on the 64-bit version of standalone SBR Carrier (with 32 CPUs x 8 cores) for Oracle SQL authentication only.

**Table 40: SQL Authentication Only Without Latency**

TPS	Authentication Thread Count	CPU Utilization	Number of Server Sections (round-robin fashion)
5000	100	190%	10
10,000	100	391%	10
15,000	200	503%	10

**Table 41: SQL Authentication Only With Backend Latency**

Latency	TPS	Authentication Thread Count	CPU Utilization	Number of Server Sections (round-robin fashion)
1 millisecond	3830	10,000	142.3%	10
3 milliseconds	1492	10,000	69%	10

## Native Accounting Start Only

[Table 42 on page 99](#) describes the results of tests run on the 64-bit version of standalone SBR Carrier (with 32 CPUs x 8 cores) for native accounting start only.

When TPS is increased along with the accounting thread count, CPU utilization and Disk I/O will also be increased. See [Table 42 on page 99](#) for the recommended accounting thread configuration based on TPS.

**Table 42: Native Accounting Start Only**

TPS <sup>4</sup>	Accounting Thread Count	CPU Utilization	Disk I/O
5000	200	134%	3 M/s

Table 42: Native Accounting Start Only (continued)

TPS <sup>4</sup>	Accounting Thread Count	CPU Utilization	Disk I/O
10,000	200	232%	4 M/s
15,000	200	466%	6.8 M/s
20,000	500	537%	8 M/s

<sup>4</sup>TPS = (Total accounting requests - Dropped accounting packets) / 600 seconds (drops are usually caused by thread exhaustion).

### Native User Authentication, Accounting Start, Accounting Stop

Table 43 on page 100 describes the results of tests run on the 64-bit version of standalone SBR Carrier (with 32 CPUs x 8 cores) for native user authentication, accounting start, and accounting stop.

With native user authentication, accounting start, and accounting stop load, the CPU utilization increases with an increase in the CPS rate as shown in Table 43 on page 100.

Table 43: Standalone: Native User Authentication, Accounting Start, Accounting Stop

CPS <sup>5</sup>	CPU Utilization	Disk I/O
10,262	2092%	2.3 M/s
12,400	2220%	2.7 M/s
13,300	2283%	3 M/s

<sup>5</sup>CPS = [Total accepts + (Total accounting requests - Dropped accounting packets)] / 600 seconds

### LDAP Authentication, Accounting Start, Accounting Stop

Table 44 on page 101 describes the results of tests run on the 64-bit version of standalone SBR Carrier (with 32 CPUs x 8 cores) for LDAP authentication, accounting start, and accounting stop.

With LDAP user authentication, accounting start, and accounting stop load, a higher number of threads is required to handle increased transaction rates, resulting in higher CPU utilization as shown in Table 44 on page 101.

**Table 44: Standalone: LDAP Authentication, Accounting Start, Accounting Stop**

CPS	Authentication Thread Count	Accounting Thread Count	CPU Utilization	Number of Server Sections	MaxConcurrent in Idapauth.aut	LDAP Server CPU Utilization <sup>6</sup>
5000	100	200	426%	10	100	75%
10,000	2,000	3,000	2598%	10	100	140%

<sup>6</sup>LDAP server hardware configuration—25 GB RAM, 8 CPUs with up to 4 cores per CPU.

### SQL Authentication, Accounting Start, Accounting Stop

[Table 45 on page 101](#) describes the results of tests run on the 64-bit version of standalone SBR Carrier (with 32 CPUs x 8 cores) for SQL authentication, accounting start, and accounting stop.

With SQL user authentication, accounting start, and accounting stop load, the CPU utilization increases with an increase in the authentication or accounting thread count and CPS rate as shown in [Table 45 on page 101](#).

**Table 45: Standalone: SQL Authentication, Accounting Start, Accounting Stop**

CPS	Authentication Thread Count	Accounting Thread Count	CPU Utilization	Number of Server Sections	MaxConcurrent in Idapauth.aut	SQL Server CPU Utilization
5,000	100	200	412%	10	100	60%
10,000	2,000	500	1235%	10	100	200%

## Test Result Summary of Standalone Running on KVM Hypervisor

The standalone SBR Carrier server is qualified to support Kernel-based Virtual Machine (KVM) hypervisor on a Red Hat Enterprise Linux 7.3 machine. The results shown in this section are tested with guest RHEL 7.3 VMs hosted on a RHEL 7.3 KVM hypervisor with the following hardware configurations:

- The host has 128 GB RAM, 32 CPUs with 8 cores per CPU, and 1.5 TB storage capacity. SBR Carrier 8.4 is running on the clients.

**NOTE:** Overprovisioning of VMs is not recommended.

**NOTE:** The results shown in this section are obtained in an environment where very few applications or services are left running during the tests.

### KVM Hypervisor Running with One VM and Varying CPU Counts

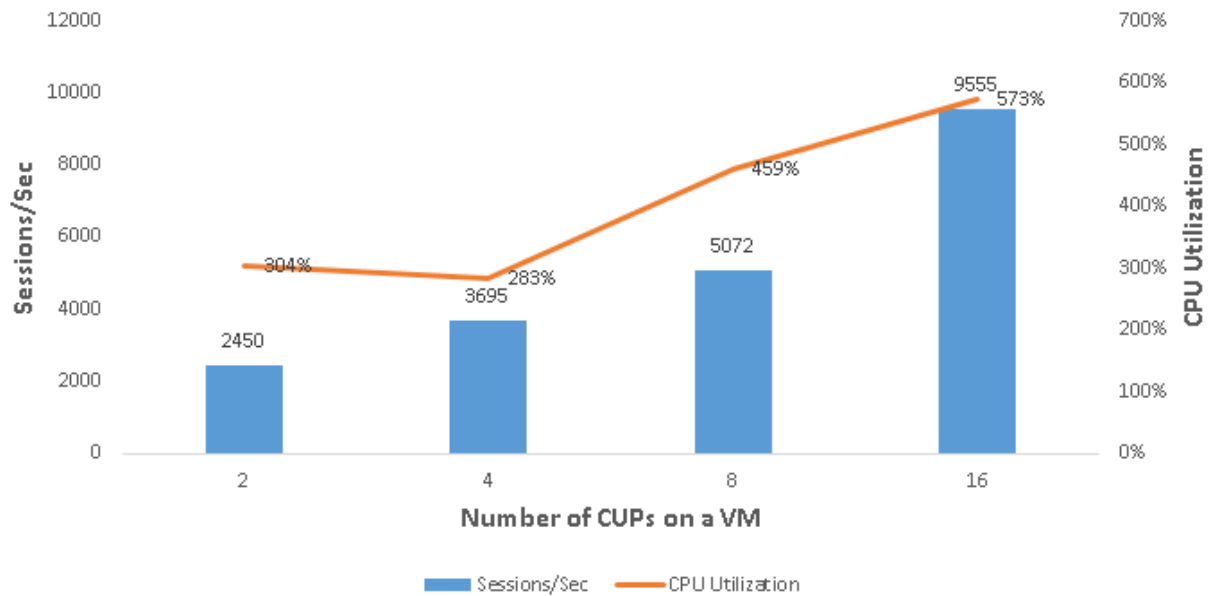
Table 46 on page 102 describes the results of tests run on the KVM hypervisor running with one VM for local accounting–start only (no SQL, proxy, or directed accounting methods). The following results are tested with a thread configuration of 500.

On a VM, if the CPU count is increased, the session rate and CPU utilization are also increased as shown in Table 46 on page 102. With the local accounting–start only load, the RADIUS process size increases with an increase in number of sessions in the in-memory session database.

**Table 46: KVM Hypervisor Running with One VM**

VM Configuration	Sessions/Second	CPU Utilization	RADIUS Process Size
2 CPUs with 2 Cores, 10 GB RAM	2450	304%	6 GB
4 CPUs with 4 Cores, 20 GB RAM	3695	283%	8.6 GB
8 CPUs with 4 Cores, 20 GB RAM	5072	459%	10.09 GB
16 CPUs with 4 Cores, 20 GB RAM	9555	573%	15.4 GB

Figure 11: KVM Hypervisor Running with One VM—Sessions per Second Versus CPU Utilization



### KVM Hypervisor Running with Six VMs

Table 47 on page 103 describes the results of tests run on the KVM hypervisor running with six VMs for local accounting only. Each VM is configured with 16 GB RAM and 4 CPUs with 4 Cores per CPU.

**NOTE:** The six VMs do not communicate with each other. They are included as a baseline for having six standalone SBRs in the same hypervisor.

Table 47: KVM Hypervisor Running with Six VMs (4 CPUs x 4 Cores)

VM	Sessions/Second	CPU Utilization
VM1	9878	274%
VM2	5346	286%
VM3	5883	276%
VM4	5574	277%
VM5	4918	238%
VM6	4493	272%



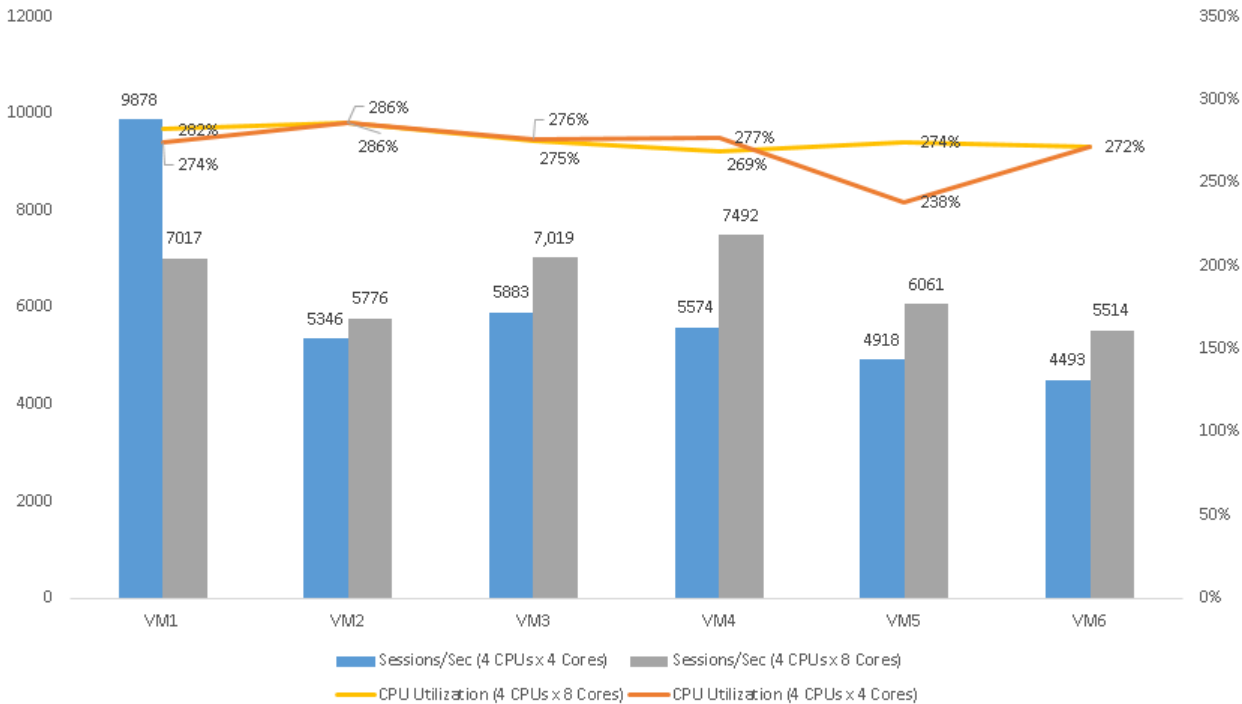
Table 48 on page 104 describes the results of tests run on the KVM hypervisor running with six VMs for local accounting only. Each VM is configured with 16 GB RAM and 4 CPUs with 8 cores.

Table 48: KVM Hypervisor Running with Six VMs (4 CPUs x 8 Cores)

VM	Sessions/Sec	CPU Utilization
VM1	7017	282%
VM2	5776	286%
VM3	7019	275%
VM4	7492	269%
VM5	6061	274%
VM6	5514	272%

Figure 12 on page 104 compares the results of VMs configured with 4 CPUs x 4 cores and 4 CPUs x 8 cores. Doubling the number of cores per CPU increases the throughput without greatly increasing the CPU utilization.

Figure 12: KVM Hypervisor Running with Six VMs: 4 CPUs x 4 Cores Versus 4 CPUs x 8 Cores



## Accounting Start Only for Proxy

[Table 49 on page 105](#) describes the results of tests run in an environment where the proxy SBR Carrier is running on a VM and the target SBR Carrier is running on a separate physical machine. The following results are tested with an accounting thread configuration of 200, **RecordLocally=1**, and **Block=1** (with **Block=1**, the accounting thread processes the proxy transaction; a separate thread is used for the transaction when **Block=0**).

**Table 49: Accounting Start Only: Proxy SBR Carrier on a VM and the Target SBR Carrier on a Separate Physical Machine**

Configuration	TPS <sup>7</sup>	CPU Utilization	HST File Size <sup>8</sup>
Proxy: 20 CPUs with 8 Cores, 100 GB RAM	5956	300%	921 MB
Target: 32 CPUs with 8 Cores, 250 GB RAM	6091	382%	913 MB

<sup>7</sup>TPS = (Total accounting requests - Dropped accounting packets) / 600 seconds

<sup>8</sup>Sessions are stored in the **.hst** file.

[Table 50 on page 105](#) describes the results of tests run in an environment where both the proxy and target are hosted as VMs on the same physical machine. The following results are tested with an accounting thread configuration of 200, test duration of 16 minutes, and **Block=1**.

**Table 50: Accounting Start Only: Proxy and Target Hosted as VMs on the Same Physical Machine**

Configuration	TPS	CPU Utilization	HST File Size
Proxy: 10 CPUs with 8 Cores, 50 GB RAM	6034	276%	926 MB
Target: 10 CPUs with 8 Cores, 50 GB RAM	6152	177%	916 MB

Comparing results of proxy and target that are hosted as VMs on the same physical machine ([Table 50 on page 105](#)) and proxy on a VM and target on a separate physical machine ([Table 49 on page 105](#)) provides approximately the same performance.