



NMC-RX™ Element Management
System for Juniper Networks®
Routing Platforms

Provisioning Service User Guide

Release 5.3.x

Juniper
NETWORKS

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA
408-745-2000
www.juniper.net

Part Number: 162-01007-00, Revision A00

Juniper Networks, the Juniper Networks logo, NetScreen, NetScreen Technologies, the NetScreen logo, NetScreen-Global Pro, ScreenOS, and GigaScreen are registered trademarks of Juniper Networks, Inc. in the United States and other countries.

The following are trademarks of Juniper Networks, Inc.: ERX, ESP, E-series, Instant Virtual Extranet, Internet Processor, J2300, J4300, J6300, J-Protect, J-series, J-Web, JUNOS, JUNOScope, JUNOScript, JUNOSe, M5, M7i, M10, M10i, M20, M40, M40e, M160, M320, M-series, MMD, NetScreen-5GT, NetScreen-5XP, NetScreen-5XT, NetScreen-25, NetScreen-50, NetScreen-204, NetScreen-208, NetScreen-500, NetScreen-5200, NetScreen-5400, NetScreen-IDP 10, NetScreen-IDP 100, NetScreen-IDP 500, NetScreen-Remote Security Client, NetScreen-Remote VPN Client, NetScreen-SA 1000 Series, NetScreen-SA 3000 Series, NetScreen-SA 5000 Series, NetScreen-SA Central Manager, NetScreen Secure Access, NetScreen-SM 3000, NetScreen-Security Manager, NMC-RX, SDX, Stateful Signature, T320, T640, and T-series. 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.

Products made or sold by Juniper Networks (including the ERX-310, ERX-705, ERX-710, ERX-1410, ERX-1440, M5, M7i, M10, M10i, M20, M40, M40e, M160, M320, and T320 routers, T640 routing node, and the JUNOS, JUNOSe, and SDX-300 software) 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.

Copyright © 2004, Juniper Networks, Inc.
All rights reserved. Printed in USA.

NMC-RX™ Element Management System Provisioning Service User Guide, Release 5.3.x
Writing: Mark Barnard, Sarah Lesway-Ball, Julie Strong, Michael Taillon
Editing: Fran Mues
Revision History
22 September 2004—Revision 1

The information in this document is current as of the date listed in the revision history.

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.

Software License

The terms and conditions for using this software are described in the software license contained in the acknowledgment to your purchase order or, to the extent applicable, to any reseller agreement or end-user purchase agreement executed between you and Juniper Networks. By using this software, you indicate that you understand and agree to be bound by those terms and conditions.

Generally speaking, the software license restricts the manner in which you are permitted to use the software and may contain prohibitions against certain uses. The software license may state conditions under which the license is automatically terminated. You should consult the license for further details.

For complete product documentation, please see the Juniper Networks Web site at www.juniper.net/techpubs.

End User License Agreement

READ THIS END USER LICENSE AGREEMENT ("AGREEMENT") BEFORE DOWNLOADING, INSTALLING, OR USING THE SOFTWARE. BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE OR OTHERWISE EXPRESSING YOUR AGREEMENT TO THE TERMS CONTAINED HEREIN, YOU (AS CUSTOMER OR IF YOU ARE NOT THE CUSTOMER, AS A REPRESENTATIVE/AGENT AUTHORIZED TO BIND THE CUSTOMER) CONSENT TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT OR CANNOT AGREE TO THE TERMS CONTAINED HEREIN, THEN (A) DO NOT DOWNLOAD, INSTALL, OR USE THE SOFTWARE, AND (B) YOU MAY CONTACT JUNIPER NETWORKS REGARDING LICENSE TERMS.

- 1. The Parties.** The parties to this Agreement are Juniper Networks, Inc. and its subsidiaries (collectively "Juniper"), and the person or organization that originally purchased from Juniper or an authorized Juniper reseller the applicable license(s) for use of the Software ("Customer") (collectively, the "Parties").
- 2. The Software.** In this Agreement, "Software" means the program modules and features of the Juniper or Juniper-supplied software, and updates and releases of such software, for which Customer has paid the applicable license or support fees to Juniper or an authorized Juniper reseller.

3. License Grant. Subject to payment of the applicable fees and the limitations and restrictions set forth herein, Juniper grants to Customer a non-exclusive and non-transferable license, without right to sublicense, to use the Software, in executable form only, subject to the following use restrictions:

- a. Customer shall use the Software solely as embedded in, and for execution on, Juniper equipment originally purchased by Customer from Juniper or an authorized Juniper reseller, unless the applicable Juniper documentation expressly permits installation on non-Juniper equipment.
- b. Customer shall use the Software on a single hardware chassis having a single processing unit, or as many chassis or processing units for which Customer has paid the applicable license fees.
- c. Other Juniper documentation for the Software (such as product purchase documents, documents accompanying the product, the Software user manual(s), Juniper's website for the Software, or messages displayed by the Software) may specify limits to Customer's use of the Software. Such limits may restrict use to a maximum number of seats, concurrent users, sessions, subscribers, nodes, or transactions, or require the purchase of separate licenses to use particular features, functionalities, or capabilities, or provide temporal or geographical limits. Customer's use of the Software shall be subject to all such limitations and purchase of all applicable licenses.

The foregoing license is not transferable or assignable by Customer. No license is granted herein to any user who did not originally purchase the applicable license(s) for the Software from Juniper or an authorized Juniper reseller.

4. Use Prohibitions. Notwithstanding the foregoing, the license provided herein does not permit the Customer to, and Customer agrees not to and shall not: (a) modify, unbundle, reverse engineer, or create derivative works based on the Software; (b) make unauthorized copies of the Software (except as necessary for backup purposes); (c) rent, transfer, or grant any rights in and to any copy of the Software, in any form, to any third party; (d) remove any proprietary notices, labels, or marks on or in any copy of the Software; (e) distribute any copy of the Software to any third party, including as may be embedded in Juniper equipment sold in the secondhand market; (f) use any 'locked' or key-restricted feature, function, or capability without first purchasing the applicable license(s) and obtaining a valid key from Juniper, even if such feature, function, or capability is enabled without a key; (g) distribute any key for the Software provided by Juniper to any third party; (h) use the Software in any manner that extends or is broader than the uses purchased by Customer from Juniper or an authorized Juniper reseller; (i) use the Software on non-Juniper equipment where the Juniper documentation does not expressly permit installation on non-Juniper equipment; (j) use the Software (or make it available for use) on Juniper equipment that the Customer did not originally purchase from Juniper or an authorized Juniper reseller; or (k) use the Software in any manner other than as expressly provided herein.

5. Audit. Customer shall maintain accurate records as necessary to verify compliance with this Agreement. Upon request by Juniper, Customer shall furnish such records to Juniper and certify its compliance with this Agreement.

6. Confidentiality. The Parties agree that aspects of the Software and associated documentation are the confidential property of Juniper. As such, Customer shall exercise all reasonable commercial efforts to maintain the Software and associated documentation in confidence, which at a minimum includes restricting access to the Software to Customer employees and contractors having a need to use the Software.

7. Ownership. Juniper and Juniper's licensors, respectively, retain ownership of all right, title, and interest (including copyright) in and to the Software, associated documentation, and all copies of the Software. Nothing in this Agreement constitutes a transfer or conveyance of any right, title, or interest in the Software or associated documentation, or a sale of the Software, associated documentation, or copies of the Software.

8. Warranty, Limitation of Liability, Disclaimer of Warranty. If the Software is distributed on physical media (such as CD), Juniper warrants for 90 days from delivery that the media on which the Software is delivered will be free of defects in material and workmanship under normal use. This limited warranty extends only to the Customer. Except as may be expressly provided in separate documentation from Juniper, no other warranties apply to the Software, and the Software is otherwise provided AS IS. Customer assumes all risks arising from use of the Software. Customer's sole remedy and Juniper's entire liability under this limited warranty is that Juniper, at its option, will repair or replace the media containing the Software, or provide a refund, provided that Customer makes a proper warranty claim to Juniper, in writing, within the warranty period. Nothing in this Agreement shall give rise to any obligation to support the Software. Any such support shall be governed by a separate, written agreement. To the maximum extent permitted by law, Juniper shall not be liable for any liability for lost profits, loss of data or costs or procurement of substitute goods or services, or for any special, indirect, or consequential damages arising out of this Agreement, the Software, or any Juniper or Juniper-supplied software. In no event shall Juniper be liable for damages arising from unauthorized or improper use of any Juniper or Juniper-supplied software.

EXCEPT AS EXPRESSLY PROVIDED HEREIN OR IN SEPARATE DOCUMENTATION PROVIDED FROM JUNIPER AND TO THE EXTENT PERMITTED BY LAW, JUNIPER DISCLAIMS ANY AND ALL WARRANTIES IN AND TO THE SOFTWARE (WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE), INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT DOES JUNIPER WARRANT THAT THE SOFTWARE, OR ANY EQUIPMENT OR NETWORK RUNNING THE SOFTWARE, WILL OPERATE WITHOUT ERROR OR INTERRUPTION, OR WILL BE FREE OF VULNERABILITY TO INTRUSION OR ATTACK.

9. Termination. Any breach of this Agreement or failure by Customer to pay any applicable fees due shall result in automatic termination of the license granted herein. Upon such termination, Customer shall destroy or return to Juniper all copies of the Software and related documentation in Customer's possession or control.

10. Taxes. All license fees for the Software are exclusive of taxes, withholdings, duties, or levies (collectively "Taxes"). Customer shall be responsible for paying Taxes arising from the purchase of the license, or importation or use of the Software.

11. Export. Customer agrees to comply with all applicable export laws and restrictions and regulations of any United States and any applicable foreign agency or authority, and not to export or re-export the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. Customer shall be liable for any such violations. The version of the Software supplied to you may contain encryption or other capabilities restricting your ability to export the Software without an export license.

12. Commercial Computer Software. The Software is "commercial computer software" and is provided with restricted rights. Use, duplication, or disclosure by the United States government is subject to restrictions set forth in this Agreement and as provided in DFARS 227.7201 through 227.7202-4, FAR 12.212, FAR 27.405(b)(2), FAR 52.227-19, or FAR 52.227-14(ALT III) as applicable.

13. Miscellaneous. This Agreement shall be governed by the laws of the State of California without reference to its conflicts of laws principles. For any disputes arising under this Agreement, the Parties hereby consent to the personal and exclusive jurisdiction of, and venue in, the state and federal courts within Santa Clara County, California. This Agreement constitutes the entire and sole agreement between Juniper and the Customer with respect to the Software, and supersedes all prior and contemporaneous agreements relating to the Software, whether oral or written (including any inconsistent terms contained in a purchase order), except that the terms of a separate written agreement executed by an authorized Juniper representative and Customer shall govern to the extent such terms are inconsistent or conflict with terms contained herein. No modification to this Agreement nor any waiver of any rights hereunder shall be effective unless expressly assented to in writing by the party to be charged. If any portion of this Agreement is held invalid, the Parties agree that such invalidity shall not affect the validity of the remainder of this Agreement.

If you have any questions about this agreement, contact Juniper Networks at the following address:

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
Attn: Contracts Administrator

Contents

About This Guide

Objectives	vii
Audience	vii
Documentation Conventions	viii
Related Juniper Networks Documentation	viii
Obtaining Documentation	ix
Comments About the Documentation	ix
Requesting Support	ix

Chapter 1 Provisioning Service Overview

Overview	1-1
Interacting with the NMC-RX Provisioning Service	1-3
IDL Interfaces	1-5
Standard IDLs	1-5
Proprietary IDLs	1-6

Chapter 2 Installing the NMC-RX Provisioning Service

Chapter 3 Starting the NMC-RX Provisioning Service

Overview	3-1
Sample Scripts for Solaris Users	3-2
Starting the NMC-RX Provisioning Service	3-4
Testing the NMC-RX Provisioning Service	3-5
Running the PVS Test Utility	3-5
Understanding PVS Test Utility Output	3-7

Appendix A Provisioning Server IDLs

unspRXDeviceMgr.idl	A-1
unspRXCustomerMgr.idl	A-11

Appendix B Sample Client Program

About This Guide

This preface provides the following guidelines for using the *NMC-RX™ Element Management System Provisioning Service User Guide*:

- Objectives on page vii
- Audience on page vii
- Documentation Conventions on page viii
- Related Juniper Networks Documentation on page viii
- Obtaining Documentation on page ix
- Comments About the Documentation on page ix
- Requesting Support on page ix

Objectives

This guide provides instructions for using the API to configure and manage your E-series routers. This guide assumes you are familiar with CORBA.



Note: *If the information in the Release Notes differs from the information in this guide, follow the Release Notes.*



Audience

This guide is intended for experienced system and network specialists who will use the NMC-RX Provisioning Service to configure Juniper Networks E-series routers.

Documentation Conventions

Table 1 defines notice icons.

Table 1 Notice icons

Icon	Meaning	Description
	Information note	Indicates important features or instructions.
	Caution	Indicates that you may risk losing data or damaging your hardware.

Related Juniper Networks Documentation

The document set comprises the books and online resources:

Table 2 Juniper Networks NMC-RX Technical Publications

<i>NMC-RX User Guide, Vol. 1</i>	Provides an introduction to NMC-RX features and information on the physical and link layer interfaces supported by the application.
<i>NMC-RX User Guide, Vol. 2</i>	Provides information on routing management, routing protocols, broadband remote access, tunnels, and services.
<i>NMC-RX Provisioning Service User Guide</i>	Provides information for using the NMC-RX Provisioning Service to manage your E-series routers.
<i>Online Documentation CD</i>	Provides an online version of this guide and the <i>Release Notes</i> .
<i>NMC-RX Help</i>	Provides an integrated help system that reflects the information contained in this document.
Release Notes	
<i>NMC-RX Release Notes</i>	<p>Include the latest information about features, changes, known problems, and resolved problems. If the information in the <i>Release Notes</i> differs from the information found in the documentation set, follow the <i>Release Notes</i>.</p> <p>Release notes are included on the corresponding software CD and are available on the Web.</p>

Obtaining Documentation

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

To order printed copies of this guide and other Juniper Networks technical documents, or to order a documentation CD, which contains this guide, contact your sales representative.

Comments About the Documentation

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation to better meet your needs. You can send your comments to techpubs-comments@juniper.net, or fill out the documentation feedback form at <http://www.juniper.net/techpubs/docbug/docbugreport.html>. 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 Support

For technical support, open a support case using the Case Manager link at <http://www.juniper.net/support/> or call 1-888-314-JTAC (within the United States) or 1-408-745-9500 (outside the United States).

1

Provisioning Service Overview

This chapter describes the NMC-RX Provisioning Service, the client classes for the provisioning service, and the IDL set; the chapter also provides examples.

Topic	Page
Overview	1-1
Interacting with the NMC-RX Provisioning Service	1-3
IDL Interfaces	1-5

Overview

The NMC-RX Provisioning Service provides a programmatic application programming interface (API) for integration with the NMC-RX Element Management System application. The service requires that the NMC-RX database and the NMC-RX ConfigSync service be fully operational. The NMC-RX database acts as a repository of information to be used both by NMC-RX client GUIs and by clients accessing the NMC-RX application via the provisioning interface.

You install the NMC-RX Provisioning Service when you install certain NMC-RX Element Management System installation sets. For information about installing the NMC-RX Provisioning Service, see the *NMC-RX User Guide, Vol. 1*. To activate the NMC-RX Provisioning Service, you must purchase a license key.

Figure 1-1 shows the NMC-RX Provisioning Service components.

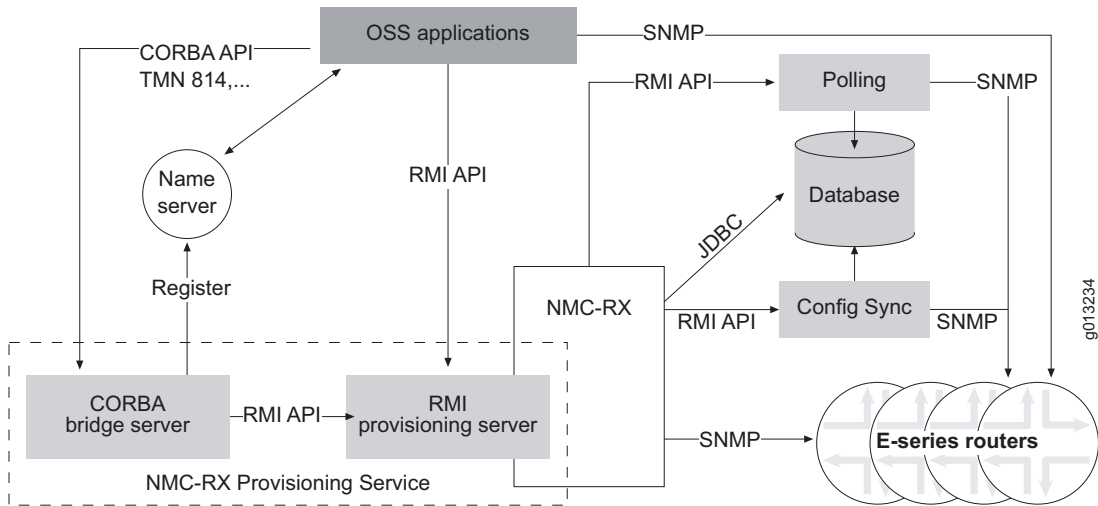


Figure 1-1 NMC-RX Provisioning Service components

You can access the provisioning interface either through common object request broker architecture (CORBA) API or Java remote method invocation (RMI) API; both allow for a wide range of integration capabilities. This guide focusses on the CORBA API.

The Juniper Networks implementation of NMC-RX Provisioning Service is based on the TeleManagement Forum (TMF).814, Interface Definition Language (IDL) Solution Set model. The basics of the TMF.814 model consist of specifying an object by its object path and specifying an action to perform on this object. We combine our own proprietary IDLs with the TMF.814 standard IDLs.

The NMC-RX application supports the following basic actions:

- View – Allows you to view all the parameters associated with a particular object.
- List All – Allows you to list all objects of a particular type related to a selected object.
- Create – Allows you to create an instance of a particular object type relative to a selected object.
- Configure – Allows you to configure or modify the parameters of a selected object.
- Delete – Allows you to delete a selected object.

Interacting with the NMC-RX Provisioning Service

The NMC-RX Element Management System (EMS) Provisioning Service provides a gateway into the managed elements of that EMS. This gateway allows you to interact with objects and the attributes of devices. A Network Management System (NMS) makes requests for information or for actions against managed elements. The EMS performs the operations necessary to carry out these requests and returns the results to the NMS. See Figure 1-2.

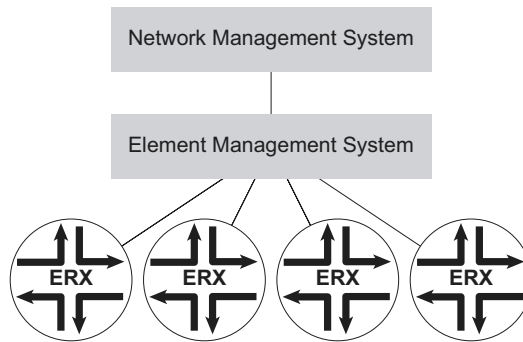


Figure 1-2 Example of managed elements

The NMC-RX Provisioning Service registers with the CORBA naming service by using a configurable name. This is the name that the CORBA client (NMS client) program uses to find the CORBA reference from the CORBA naming service. For example:

```
JUNIPER_NMCRX_TMF814_EMSSSESSIONFACTORY_8384
```

This name comprises three parts:

- JUNIPER_NMCRX – Identifies the vendor's EMS.
- TMF814_EMSSSESSIONFACTORY – Refers to the TMF.814 EMS Session Factory object, which is the entry object for the NMS client to access EMS.
- 8384 – Specifies the port that the RMI server is running on.

You can change the first part of the name, JUNIPER_NMCRX, and the port number, 8384, by modifying the provisioningserver.rc file under NMC-RX_HOME. The NMS client then retrieves the NMC-RX provisioning server entry reference `emsSessionFactory.idl` by looking up this name from the CORBA name server.

The standard IDL `emsSessionFactory.idl` is the primary class that the NMS communicates with, which acts as the focal point for gaining access

to managed elements. From the standard IDL, `emsSession.idl`, a programmer can obtain a specific `unspRXDeviceMgr.idl` to perform inquiries and actions against a specific device. Once a particular handle to a device is formed, all operations on that device can be achieved via the `unspRXDeviceMgr.idl` when you use the appropriate action and object paths.

The other proprietary IDL, `unspRXCustomerMgr.idl`, is very similar to `unspRXDeviceMgr.idl`; however, it applies to the EMS level and does not apply to devices. For `unspRXCustomerMgr.idl`, you do not need to set `ManagedElement` when you configure the object path.

Figure 1-3 illustrates the process of accessing E-series router objects and provides a listing of the objects supported in this release.

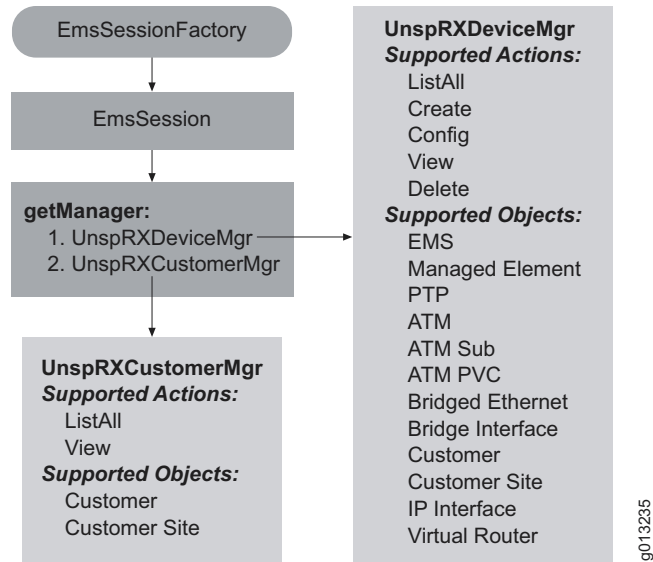


Figure 1-3 NMC-RX Provisioning Service objects

IDL Interfaces

The Interface Definition Language (IDL) solution set defines all the interfaces between the NMS and the EMS, enabling the NMS to manage elements by going through the EMS. The provisioning service IDL set includes both standard and proprietary IDLs.

Standard IDLs

The NMC-RX Provisioning Service package supports the following TeleManagement Forum (TMF) 814 standard IDLs, which build the framework interface of the provisioning service API:

- `globaldefs.idl` – Defines common object types used by other modules of the EML-NML interface. This module is intended as a common repository for definitions that need to be exported across modules.
- `common.idl` – Contains the definition of the common interface of the NML-EML interface.
- `session.idl` – Contains the definition of the common session management interface of the NML-EML interface. The `Session_I` interface provides capabilities to manage the client-server connection. Its main purpose is to enable either a client or server to detect the loss of communication with the associated party.
- `emsSession.idl` – Contains the definition of the `emsSession` interface of the NML-EML interface. The `emsSession` module provides a means for the client to interrogate the EMS to determine which manager interfaces it supports. The NMS can then retrieve the instance of the manager interface objects it requires. Retrieval is achieved with generic IDL so that new manager interfaces can be added easily.
- `nmsSession.idl` – Contains the definition of the `nmsSession` interface of the NML-EML interface. The `nmsSession` module provides a means for the server to inform the NMS of problems with the notifications.
- `emsSessionFactory.idl` – Contains the definition of the `emsSessionFactory_I` interface of the NML-EML interface. There is a single instance of `emsSessionFactory_I`. It is the entry point to the server. This is the object reference that the client uses to connect to the server. The operation `getEmsSession(...)` allows the NMS to obtain the `EmsSession_I` object from which all managers of the EMS can be obtained.
- `mtnmVersion.idl` – Contains the definition of the interface version of the NML-EML interface.

Proprietary IDLs

The NMC-RX Provisioning Service package supports the following Juniper Networks proprietary IDLs, which offer the necessary interfaces to manage Juniper Networks elements and devices:

- unspRXDeviceMgr.idl – Supported in this release. Defines all methods for the NMS to manage Juniper Networks devices. See *Appendix A, Provisioning Server IDLs*.
- unspRXCustomerMgr.idl – Supported in this release. Defines the view and list-all methods for NMS to manage the customer and customer site at the EMS level.

Installing the NMC-RX Provisioning Service

2

You can install the NMC-RX Provisioning Service Package by launching one of the NMC-RX Element Management System install sets. For instruction on which install set to use and how to install the NMC-RX Provisioning Service Package, see *NMC-RX User Guide, Vol. 1*.

3

Starting the NMC-RX Provisioning Service

This chapter presents information on starting the NMC-RX Provisioning service software.

Topic	Page
Overview	3-1
Starting the NMC-RX Provisioning Service	3-4
Testing the NMC-RX Provisioning Service	3-4

Overview

Juniper Networks uses CORBA freeware Distributed Object Group (DOG) for development and testing. We ship the DOG CORBA freeware package version 2.6 with our NMC-RX Provisioning service (PVS) installer. We offer sample scripts to start NMC-RX Provisioning service based on this CORBA package. You can use a different CORBA package; however, you must recompile `CorbaApi.java`, which is under the PVS directory.

Sample Scripts for Solaris Users



Note: *If a service is already running, it will not start again.*

The `pvsSampleScript` directory contains the following sample scripts:

- `naming` – Use to start the DOG CORBA Naming service. It uses port 2001. When this service starts, it generates an `ins.ior` file and copies it to `../pvs/JuniperERXNameService.ior` file for use in starting the CORBA Bridge service. Because this relative path is used, be sure to start the sample script from this directory, unless you choose to modify the script.
- `nmcPvsRmiService` – Use to start the remote method invocation (RMI) service.
- `nmcPvsCorbaBridgeService` – Use to start the CORBA Bridge service. If you do not designate a port number, the system assigns a port number. If you want to designate a specific port number (for example, port 12345), modify the `nmcPvsCorbaBridgeService` script by adding option `-OAPort=12345`.
- `nmcPvsTestUtility` – Use to start the NMC-RX Provisioning service test utility. For additional information about this utility, see “Testing the NMC-RX Provisioning Service” on page 3-5.
- `allpvs` (Solaris only) – Use to launch all services at one time in the correct order (NMC-RX Database, Naming service, `remiService`, and CORBA Bridge service).

Use the help option to obtain usage instructions for the scripts.



Note: *The help feature and all of the options (for example, start, start console, and so on) exist only on Solaris.*

The following is a sample of allpvs help:

```
# allpvs help

Usage: allpvs {[start [console]]|stop|restart [console]|status|help}

start - starts NMC-RX database, Naming Service, RMI Service,
       and Corba Bridge Service as background processes.

start console - starts NMC-RX database, Naming Service, RMI Service,
               and Corba Bridge Service in new windows.

stop - stops NMC-RX database, Naming Service, RMI Service,
       and Corba Bridge Service.

restart - stops the NMC-RX database, Naming Service, RMI Service,
          and Corba Bridge Service if they're running, and starts them
          again.

restart console - stops the NMC-RX database, Naming Service, RMI Service,
                 and Corba Bridge Service if they're running, and starts
                 them in new windows.

status - states whether or not the NMC-RX database, Naming Service, RMI
        Service, and Corba Bridge Service are currently running.

help - displays this message.
```

Starting the NMC-RX Provisioning Service

You must run the following four services in order to start the Provisioning service:

- NMC-RX Database service
- CORBA Naming service
- NMC-RX Provisioning RMI service
- NMC-RX Provisioning CORBA Bridge service

To use the NMC-RX Provisioning Service, all of these services must be running.

Three of the services—Database, Naming, and Provisioning RMI—are located in the `pvsSampleScripts` directory. For information about starting the NMC-RX Database service, see *NMC-RX User Guide, Vol. 1*.

To start the NMC-RX Provisioning service:

- 1** (Optional – This step is an alternative to performing steps 2 through 7 and available only on Solaris)

Run the `allpvs` with or without console windows.

- a** Run `allpvs start`.

The four services (NMC-RX Database, Naming, RMI, and CORBA Bridge) start automatically without individual console windows.

- b** Run `allpvs start console`.

The four services (NMC-RX Database, Naming, RMI, and CORBA Bridge) start automatically. Each service has its own console window, which allows you to monitor each process.

- 2** Start the NMC-RX database.
- 3** Start the CORBA Naming service by running either `naming` or `naming start console`.

An `ins.ior` file is generated and is copied to the filename `JuniperERXNameService.ior` located under the `pvs` directory.

- 4** Start the NMC-RX Provisioning Service RMI Service by running the `nmcPvsRmiService` script.

When you see the following message, the RMI service is ready:

```
EmsSessionsFactory binded to JUNIPER_NMCRX_TMF814_EMSSSESSIONFACTORY_8284
```



Note: The name "JUNIPER_NMCRX_TMF814_EMSSSESSIONFACTORY_8384" is configurable by using file PVS.rc under NMCRX_HOME. There are four lines in this file:

```
provisioning_server_port=8384
provisioning_server_emsname=JUNIPER_NMCRX
provisioning_server_log_enabled=true
provisioning_server_session_checking_time_interval=180000
```



Note: The name "JUNIPER_NMCRX_TMF814_EMSSSESSIONFACTORY_8384" is used for NMC-RX Provisioning Service CORBA service to find RMI Service's entry reference. This is the name that the CORBA client (NMS client) program uses to find the CORBA reference from the CORBA Naming Service.

- 5 Start the NMC-RX Provisioning service CORBA Bridge service by running the nmcPvsCorbaBridgeService script.

The NMC-RX Provisioning service software is ready to be used by network management system (NMS) clients.

Testing the NMC-RX Provisioning Service

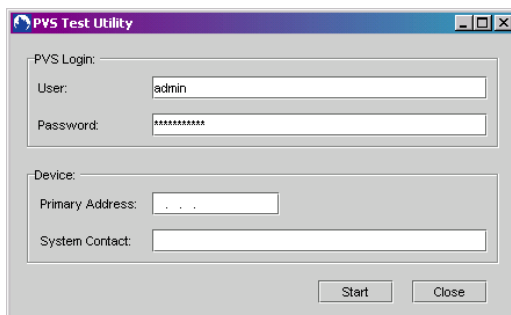
You can use the PVS test utility to test the NMC-RX Provisioning service. This utility attempts to connect to the database, the RMI server, and the CORBA server. After connecting, the utility attempts to exchange data over the CORBA bridge by viewing a specified node or all of the nodes.

Running the PVS Test Utility

To run the PVS test utility:

- 1 Launch the nmcPvsTestUtility program from the following location:
<NMC-RX-HOME>/pvsSampleScripts/nmcPvsTestUtility

The PVS Test Utility window opens.



- 2 Specify your NMC-RX username and password in the PVS Login area.
- 3 (Optional) Specify a primary IP address in the Device area for the device that you want to test.



Note: If you do not specify a primary IP address, the PVS test utility lists all devices on the system.

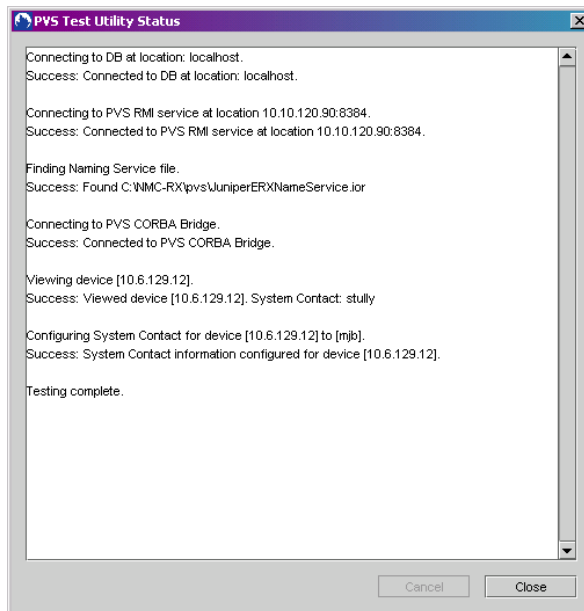
- 4 (Optional) Specify a system contact in the Device area to change the system contact.



Note: If you do not specify a system contact, the system contact on the device does not change.

- 5 Click Start.

The PVS Test Utility Status dialog box opens.



Once the PVS test utility starts, you can cancel the test or close the dialog box at any time. After the PVS test utility finishes its testing, you can close the dialog box.

Understanding PVS Test Utility Output

The PVS test utility performs a sequence of tests. These tests provide output based on test results. The following table lists each test performed, failure criteria for each test, and potential output text for each test.

Test	Procedure	Failure Criteria	Messages
Database	Obtains database location and user information from local NMC-RX resource file; attempts to contact database	<ul style="list-style-type: none"> Database is not running at the assigned location Database settings in the NMC-RX resources are not correct Various other reasons 	Initial status text: Connecting to DB at location: localhost. Final status text: Success: Connected to DB at location: localhost. Success: Connected to DB at location: <i>ipAddress</i> . Failure: Could not connect to DB at location: <i>ipAddress</i> . [Followed by exception message]
RMI Server	Obtains PVS port number from Provisioning service resources file; contacts RMI server at local IP address	<ul style="list-style-type: none"> RMI server is not running Various other reasons 	Initial status text: Connecting to PVS RMI service at location <i>ipAddress:portNumber</i> . Final status text: Success: Connected to PVS RMI service at location <i>ipAddress:portNumber</i> . Failure: Could not connect to PVS RMI service at location <i>ipAddress:portNumber</i> . [Followed by exception message]
IOR File	Searches for JuniperERXNameService.ior file in PVS subdirectory	<ul style="list-style-type: none"> Unable to find IOR file (indicates naming service has not been run) 	Initial status text: Finding Naming Service file. Final status text: Success: Found <i>d:path\filename.ior</i> . Failure: Could not find <i>d:path\filename.ior</i> . [Followed by exception message]
CORBA Bridge	Uses the Naming service to contact the PVS	<ul style="list-style-type: none"> The PVS CORBA bridge is not functioning Naming service is not running Various other reasons 	Initial status text: Connecting to PVS CORBA Bridge. Final status text: Success: Connected to PVS CORBA Bridge. Failure: Could not connect to PVS CORBA Bridge. [Followed by exception message]

Test	Procedure	Failure Criteria	Messages
Nonempty IP Address field	Requests information about the device associated with the IP address you provide	<ul style="list-style-type: none"> Unable to find the specified device Naming service is not running CORBA bridge is not running 	<p>Initial status text: Viewing device [<i>ipAddress</i>].</p> <p>Final status text: Success: Viewed device [<i>ipAddress</i>].</p> <p>Failure: Could not view information for device [<i>ipAddress</i>]. [Followed by exception message]</p>
Empty IP address field	Requests information about all devices on the system	<ul style="list-style-type: none"> Unable to list devices Naming service is not running CORBA bridge is not running 	<p>Initial status text: Listing devices.</p> <p>Successful status text: Success: No devices found.</p> <p>Successful status text: Success: <i>ipAddress1</i> <i>ipAddress2</i> <i>ipAddress3</i> <i>ipAddress4</i></p> <p>Status text for failure: Failure: Could not list devices. [Followed by exception message]</p>
Nonempty IP address field and nonempty System Contact field	Sets the system contact for the device to the specified value	<ul style="list-style-type: none"> Unable to configure system contact value 	<p>Initial status text: Configuring System Contact for device [<i>ipAddress</i>] to [<i>systemContactText</i>].</p> <p>Final status text: Success: System Contact information configured for device [<i>ipAddress</i>].</p> <p>Failure: Could not configure System Contact Information.</p>
Status	Indicates that all tests are complete	None	<p>Status text: Testing complete.</p>

Provisioning Server IDLs



This appendix presents the Juniper Networks proprietary IDL set for this release of the provisioning service. The IDLs presented here define all the methods for the NMS to manage Juniper Networks devices. To get these references, the NMS needs to go through the `getManager(...)` of `emsSession` reference.

unspRXDeviceMgr.idl

The following IDL is the definition of the `unspRXDeviceMgr.idl`:

```
#ifndef unspRXDeviceMgr_idl
#define unspRXDeviceMgr_idl

// *****
// *
// * unspRXDeviceMgr.idl
// *
// *****

//Include list
#include "globaldefs.idl"
#include "common.idl"

#pragma prefix "nmcRX.unspCORBAbridge.provisioningservice.services.redstonecom.com"

module unspRXDeviceMgr
{
/**
 * UnspRXDeviceMgrObjectType defined all possible object types for NMC-RX to manage
```

```

*/
enum UnspRXDeviceMgrObjectType
{
    // Layered Objects                                // Examples:
    EMS,                                                // "Juniper NMC-RX"
    ManagedElement,                                    // "10.6.129.6"
    PTP,                                                // "6/0" - where 6 is the slot number, and
                                                        // 0 is the port number
    AtmInterface,                                     // "ATM6/0-atm-layer",
    AtmSubInterface,                                  // "ATM6/0.1-atm1483-subif",
    AtmPvc,                                            // "ATM6/0.1-atm1483-subif vpi=1,vci=600"

    PPP,                                               // not supported
    PPPoE,                                             // not supported
    PPPoESubInterface,                               // not supported
    BridgedIP_rfc1483,                                // "4/0.10"
    VirtualRouter,                                   // "default", "VR1"

    IpInterface,

    Customer,
    CustSite,
    BridgedEthernet,                                // Deprecated
    BridgeInterface                                  // "BridgeIf6/0.1"
};

/**
 * UnspRXDeviceMgrObjectTypeList is a list UnspRXDeviceMgrObjectType
 */
typedef sequence<UnspRXDeviceMgrObjectType> UnspRXDeviceMgrObjectTypeList;

/**
 * UnspRXActionType defined all possible action types
 * for NMC-RX to manage E-series routers
 */
enum UnspRXActionType
{
    CREATE,
    CONFIGURE,
    LIST,
    VIEW,
    DELETE
};

/**

```

```

*   UnspRXActionTypeList is a list UnspRXActionType
*/
typedef sequence<UnspRXActionType>UnspRXActionTypeList;

/**
*   UnspRXDeviceMgr_I defines all methods to manage objects
*   under ERX. It is derived from common::Common_I, which is
*   one of the idls defined by TMF 814.
*/

interface UnspRXDeviceMgr_I : common::Common_I
{
    /**
    *   This method should be used to list all objects of a particular type relative
    *   to another object.
    *
    *   The retrieval mechanism follows the paradigm established in TMF 814
    *   and is described below:
    *
    *   objectPath:   The name-value pair list of the objects you want to list
relative to other objects.
    *       Example:
    *       ListAll AtmSubInterfaces relative to objectPath
    *           Name="EMS",           Value="Juniper NMC-RX"
    *           Name="ManagedElement", Value="10.6.129.6"
    *           Name="PTP",           Value="6/0"
    *           Name="AtmInterface",  Value="ATM6/0-atm-layer"
    *
    *       ListAll AtmPvc relative to objectPath
    *           Name="EMS",           Value="Juniper NMC-RX"
    *           Name="ManagedElement", Value="10.6.129.6"
    *           Name="PTP",           Value="6/0"
    *
    *   objectType:   The type of objects that you want to list
    *       Example:
    *       UnspRXDeviceMgrObjectType.AtmSubInterfaces
    *       UnspRXDeviceMgrObjectType.AtmPvc
    *
    *   howMany:      The maximum number of objects to report in the first batch
in nameList
    *
    *   nameList:     First batch of howMany objects report to NMS
    *
    *   nameIt:       Iterator to retrieve the remaining object after the first
batch
    *
    *       To allow the NMS to deal more easily with retrievals which may return a

```

```

*      very large amount of data, iterators are used. They provide a mechanism
to retrieve
*      data batch by batch - the size of a batch is specified by the NMS via the
*      how_many parameter in the next_n method of the iterator reference.
*      For a more detailed explanation of the use of iterators, please refer to
*      TMF814\supportingDocumentation\iterators.html
*      Raises: globaldefs::ProcessingFailureException
*      EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
*      EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
*      EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
*      EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
*
*      can support has been reached.
*      EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
**/
void listAll(
    in globaldefs::NVSList_T          objectPath,
    in UnspRXDeviceMgrObjectType      objectType,
    in long                            howMany,
    out globaldefs::NamingAttributesList_T    nameList,
    out globaldefs::NamingAttributesIterator_I    nameIt)
    raises(globaldefs::ProcessingFailureException);

/**
* This list works the same as the other one except that it returns only those
objects created
* after timeStamp. Also, objects are sorted by timeStamp when they are
returned.
*/
void listAllFromTime(
    in globaldefs::NVSList_T          objectPath,
    in UnspRXDeviceMgrObjectType      objectType,
    in string                          timeStamp,
    in long                            howMany,
    out globaldefs::NamingAttributesList_T    nameList,
    out globaldefs::NamingAttributesIterator_I    nameIt)
    raises(globaldefs::ProcessingFailureException);

/**
* This method should be used to create an object of a particular type.
*
* objectPath:    The name-value pair list of the objects you want to create.
* Example:
* ListAll AtmSubInterfaces relative to objectPath
* Name="EMS",          Value="Juniper NMC-RX"

```

```

*         Name="ManagedElement", Value="10.6.129.6"
*         Name="PTP",             Value="6/0"
*         Name="AtmInterface",    Value="ATM6/0-atm-layer"
*         Name="AtmSubInterfaces",Value="ATM6/0.1-atm1483-subif"
*     Name="AtmPvc",             Value="ATM6/0.1-atm1483-subif vpi=1,vci=600"
*
*     objectType:    The type of objects that you want to list
*     Example:
*     UnspRXDeviceMgrObjectType.AtmPvc
*
*     attributesList: It will hold all attributes as a name-value pair list
*                    to pass to NMC-RX to create this object
*
*     assigned     objectName: The object name has just been created by this method. It is
*                    by device instead of being specified by the user
*
*     failure      Raises: globaldefs::ProcessingFailureException
*                    EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
method      *     EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
existing      *     EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
object      *     EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS *                    can support has been reached.
*     **/
void create(
    in globaldefs::NVSList_T          objectPath,
    in UnspRXDeviceMgrObjectType     objectType,
    in globaldefs::NVSList_T          attributesList,
    out string                        objectName )
    raises(globaldefs::ProcessingFailureException);

/**
*     This method should be used to configure objects
*
*     objectPath:    The name-value pair list of the object you want to configure.
*     Example:
*     ListAll AtmSubInterfaces relative to objectPath
*     Name="EMS",             Value="Juniper NMC-RX"
*     Name="ManagedElement", Value="10.6.129.6"
*     Name="PTP",             Value="6/0"
*     Name="AtmInterface",    Value="ATM6/0-atm-layer"
*     Name="AtmSubInterfaces",Value="ATM6/0.1-atm1483-subif"
*
*     Name="AtmPvc",             Value="ATM6/0.1-atm1483-subif vpi=1,vci=600"

```

```

*
* attributesList: It will hold all attributes as name value pair list
* to pass to NMC-RX to configure this object.
*
* Raises: globaldefs::ProcessingFailureException
* EXCPT_INTERNAL_ERROR - Raised in case of a nonspecific EMS internal
failure
* EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
* EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
* EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
*
* can support has been reached.
* EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
**/
void config(
    in globaldefs::NVSLIST_T objectPath,
    in globaldefs::NVSLIST_T attributesList)
    raises(globaldefs::ProcessingFailureException);

/**
* This method should be used to delete an object
*
* objectPath: The name-value pair list of the object you want to delete.
* Example:
* ListAll AtmSubInterfaces relative to objectPath
* Name="EMS", Value="Juniper NMC-RX"
* Name="ManagedElement", Value="10.6.129.6"
* Name="PTP", Value="6/0"
* Name="AtmInterface", Value="ATM6/0-atm-layer"
* Name="AtmSubInterfaces", Value="ATM6/0.1-atm1483-subif"
* Name="AtmPvc", Value="ATM6/0.1-atm1483-subif vpi=1,vci=600"
* Raises: globaldefs::ProcessingFailureException
* EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
* EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
* EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
* EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
*
* can support has been reached.
* EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
**/
void delete(in globaldefs::NVSLIST_T objectPath)
    raises(globaldefs::ProcessingFailureException);

/**
* This method should be used to view an object

```

```

*
* objectPath:      The name-value pair list of the object you want to view.
*   Example:
*   ListAll AtmSubInterfaces relative to objectPath
*       Name="EMS",           Value="Juniper NMC-RX"
*       Name="ManagedElement", Value="10.6.129.6"
*       Name="PTP",           Value="6/0"
*       Name="AtmInterface",  Value="ATM6/0-atm-layer"
*       Name="AtmSubInterfaces",Value="ATM6/0.1-atm1483-subif"
* Name="AtmPvc",           Value="ATM6/0.1-atm1483-subif vpi=1,vci=600"
*
* attributesList: It will hold all attributes as the name value pair list
*                 to pass back to NMS to view this object.
* Raises: globaldefs::ProcessingFailureException
*   EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
*   EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
*   EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
*   EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
*
*                                     can support has been reached.
*   EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
**/
void view(
    in globaldefs::NVSLIST_T      objectPath,
    out globaldefs::NVSLIST_T     attributesList)
    raises(globaldefs::ProcessingFailureException);

/**
* This method should be used to get the attributes descriptions in order
* to create an object
*
* objectPath:      The name-value pair list of the object you want to view.
*   Example:
*   ListAll AtmSubInterfaces relative to objectPath
*       Name="EMS",           Value="Juniper NMC-RX"
*       Name="ManagedElement", Value="10.6.129.6"
*       Name="PTP",           Value="6/0"
*       Name="AtmInterface",  Value="ATM6/0-atm-layer"
*       Name="AtmSubInterfaces",Value="ATM6/0.1-atm1483-subif"
*
*
* objectType:      The type of objects that you want to create
*   Example:
*   UnspRXDeviceMgrObjectType.AtmPvc
*

```

```

*   attributesDescription: It will hold all attributes with their description
*                           as name-value pair list to pass back to NMS.
*                           The name will be the attribute name, but the value
*                           will have the attribute type and the description of
*                           this attribute, seperated by the first ":".
*   Raises: globaldefs::ProcessingFailureException
*           EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
*           EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
*           EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
*           EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
*
*                           can support has been reached.
*   EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
**/
void getCreateAttributesDescription(
    in globaldefs::NVSLIST_T      objectPath,
    in UnsprXDeviceMgrObjectType  objectType,
    out globaldefs::NVSLIST_T     attributesDescription)
    raises(globaldefs::ProcessingFailureException);

/**
*   This method should be used to get the attributes description in order
*   to config an object
*
*   objectPath:   The name-value pair list of the object you want to view.
*   Example:
*   ListAll AtmSubInterfaces relative to objectPath
*       Name="EMS",           Value="Juniper NMC-RX"
*       Name="ManagedElement", Value="10.6.129.6"
*       Name="PTP",           Value="6/0"
*       Name="AtmInterface",  Value="ATM6/0-atm-layer"
*       Name="AtmSubInterfaces",Value="ATM6/0.1-atm1483-subif"
*
*   attributesDescription: It will hold all attributes with their description
*                           as name-value pair list to pass back to NMS.
*                           The name will be the attribute name, but the value
*                           will have the attribute type and the description of
*                           this attribute, seperated by the first ":".
*   Raises: globaldefs::ProcessingFailureException
*           EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
*           EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
*           EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object

```

```

    *      EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
    *
    *      can support has been reached.
    *      EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
    **/
void getConfigureAttributesDescription(
    in globaldefs::NVSList_T      objectPath,
    out globaldefs::NVSList_T      attributesDescription)
    raises(globaldefs::ProcessingFailureException);

/**
 * This method should be used to get a list of object types on which you can
perform a list of
 * actions relative to other objects.
 *
 * objectPath:    The name-value pair list of the object you want to view.
 * Example:
 * ListAll AtmSubInterfaces relative to objectPath
 *      Name="EMS",           Value="Juniper NMC-RX"
 *      Name="ManagedElement", Value="10.6.129.6"
 *      Name="PTP",           Value="6/0"
 *      Name="AtmInterface",  Value="ATM6/0-atm-layer"
 *      Name="AtmSubInterfaces",Value="ATM6/0.1-atm1483-subif"
 *
 * objectTypeList: To hold a list of object types
 *
 * Raises: globaldefs::ProcessingFailureException
 *      EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
 *      EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
 *      EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
 *      EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
 *
 *      can support has been reached.
 *      EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
    **/
void getListableObjectTypesSelections(
    in globaldefs::NVSList_T      objectPath,
    out UnspRXDeviceMgrObjectTypesList objectTypeList)
    raises(globaldefs::ProcessingFailureException);

/**
 * This method should be used to get a list of object types on which you can
create
 * actions relative to other objects.
 *

```

```

*   objectPath:   The name-value pair list of the object you want to view.
*   Example:
*   ListAll AtmSubInterfaces relative to objectPath
*       Name="EMS",           Value="Juniper NMC-RX"
*       Name="ManagedElement", Value="10.6.129.6"
*       Name="PTP",           Value="6/0"
*       Name="AtmInterface",   Value="ATM6/0-atm-layer"
*       Name="AtmSubInterfaces",Value="ATM6/0.1-atm1483-subif"
*
*   objectTypeList: To hold a list of object types
*
*   Raises: globaldefs::ProcessingFailureException
*           EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
*           EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
*           EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
*           EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
*
*               can support has been reached.
*   EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
**/
void getCreatableObjectTypeSelections(
    in globaldefs::NVList_T      objectPath,
    outUnsprXDeviceMgrObjectTypeList objectTypeList)
    raises(globaldefs::ProcessingFailureException);

/**
*   This method should be used to get a list of types of actions you can perform
*   relative to other objects.
*
*   objectPath:   The name-value pair list of the objects you want to view.
*   Example:
*   ListAll AtmSubInterfaces relative to objectPath
*       Name="EMS",           Value="Juniper NMC-RX"
*       Name="ManagedElement", Value="10.6.129.6"
*       Name="PTP",           Value="6/0"
*       Name="AtmInterface",   Value="ATM6/0-atm-layer"
*       Name="AtmSubInterfaces",Value="ATM6/0.1-atm1483-subif"
*
*   objectTypeList: To hold a list of action types
*
*   Raises: globaldefs::ProcessingFailureException
*           EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
*           EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method

```

```

        *          EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
        *          EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
        *
        *          can support has been reached.
    *          EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
    **/
void getSupportedActions(
    in globaldefs::NVSLIST_T    objectPath,
    outUnspRXActionTypeList    actionTypesList)
    raises(globaldefs::ProcessingFailureException);

};
};
#endif

```

unspRXCustomerMgr.idl

The following IDL is the definition of the unspRXCustomerMgr.idl:

```

#ifndef unspRXCustomerMgr_idl
#define unspRXCustomerMgr_idl

// *****
// *
// * unspRXCustomerMgr.idl *
// *
// *****

//Include list
#include "globaldefs.idl"
#include "common.idl"

#pragma prefix "nmcRX.unspCORBAbridge.provisioningservice.services.redstonecom.com"

module unspRXCustomerMgr
{
    /**
     * UnspRXCustomerMgrObjectType defined all possible object types for NMC-RX to
manage
     */
    enum UnspRXCustomerMgrObjectType
    {

```

```

        // Layered Objects                                // Examples:
        EMS,                                             // "Juniper NMC-RX"
        Customer,                                       // "ABC Store Customer"
        CustSite                                        // "Westford Customer Site"
    };

/**
 * UnspRXCustomerMgrObjectTypeList is a list UnspRXCustomerMgrObjectType
 */
typedef sequence<UnspRXCustomerMgrObjectType> UnspRXCustomerMgrObjectTypeList;

/**
 * UnspRXActionType defined all possible action types
 * for NMC-RX to manage ERX/MRX
 */
enum UnspRXCustomerActionType
{
    LIST,
    VIEW
};

/**
 * UnspRXActionTypeList is a list UnspRXActionType
 */
typedef sequence<UnspRXCustomerActionType>UnspRXCustomerActionTypeList;

/**
 * UnspRXCustomerMgr_I defines all methods to manage objects
 * under ERX/MRX. It is derived from common::Common_I, which is
 * one of the idls defined by TMF 814.
 */

interface UnspRXCustomerMgr_I : common::Common_I
{
    /**
     * This method should be used to list all objects of a particular type relative
     * to another object.
     *
     * The retrieval mechanism follows the paradigm established in TMF 814
     * and is described below:
     *
     * objectPath:    The name-value pair list of objects you want to list relative
     to another object.
     *
     * Example:
     * ListAll Customer relative to objectPath
     *
     * Name="EMS",           Value="Juniper NMC-RX"
    */

```

```

*
* ListAll Customer Site relative to objectPath
*     Name="EMS",           Value="Juniper NMC-RX"
*     Name="Customer",     Value="ABC Store Customer"
*
* objectType:    The type of objects that you want to list
*     Example:
*     UnspRXCustomerMgrObjectType.Customer
*     UnspRXCustomerMgrObjectType.CustSite
*
* howMany:       The maximum number of objects to report in the first batch
in nameList
*
* nameList:      First batch of howMany objects report to NMS
*
* nameIt:        Iterator to retrieve the remaining number of objects after
the first batch
*     In order to allow the NMS to deal more easily with retrievals, which may
return a
*     very large amount of data, iterators are used. They provide a mechanism
to retrieve
*     data batch by batch - the size of a batch is specified by the NMS via the
*     how_many parameter in the next_n method of the iterator reference.
*     For a more detailed explanation of the use of iterators, please refer to
*     TMF814\supportingDocumentation\iterators.html
* Raises: globaldefs::ProcessingFailureException
*     EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
*     EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
*     EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
*     EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
*
*                                     can support has been reached.
*     EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
**/
void listAll(
    in globaldefs::NVSList_T           objectPath,
    in UnspRXCustomerMgrObjectType    objectType,
    in long                             howMany,
    out globaldefs::NamingAttributesList_T    nameList,
    out globaldefs::NamingAttributesIterator_I    nameIt)
    raises(globaldefs::ProcessingFailureException);

/**
* This method should be used to view objects
*
* objectPath:    The name-value pair list of the object you want to configure.

```

```

*      Example:
*      View Customer Site relative to objectPath
*          Name="EMS",           Value="Juniper NMC-RX"
*          Name="Customer",      Value="ABC Store Customer"
*          Name="CustSite",      Value="Westford Customer Site"
*
*      attributesList: It will hold all attributes as a name-value pair list
*                      to pass back to NMS to view this object.
*      Raises: globaldefs::ProcessingFailureException
*          EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
*          EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
*          EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
*          EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
*
*                      can support has been reached.
*      EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
**/
void view(
    in globaldefs::NVSList_T      objectPath,
    out globaldefs::NVSList_T     attributesList)
    raises(globaldefs::ProcessingFailureException);

/**
*      This method should be used to get a list of object types on which you can
perform list
*      actions relative to other objects.
*
*      objectPath:   The name-value pair list of the object you want to view.
*      Example:
*      ListAll AtmSubInterfaces relative to objectPath
*          Name="EMS",           Value="Juniper NMC-RX"
*
*      objectTypeList: To hold a list of object types
*
*      Raises: globaldefs::ProcessingFailureException
*          EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure
*          EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method
*          EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
*          EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
*
*                      can support has been reached.
*      EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
**/

```

```

void getListableObjectTypesSelections(
    in globaldefs::NVList_T      objectPath,
    out UnspRXCustomerMgrObjectTypeList  objectTypeList)
    raises(globaldefs::ProcessingFailureException);

/**
perform * This method should be used to get a list of types of action that you can
* relative to other objects.
*
* objectPath:    The name-value pair list of the object you want to view.
* Example:
* ListAll AtmSubInterfaces relative to objectPath
* Name="EMS",      Value="Juniper NMC-RX"
*
* objectTypeList: To hold a list of action types
*
* Raises: globaldefs::ProcessingFailureException
* EXCPT_INTERNAL_ERROR - Raised in case of nonspecific EMS internal
failure * EXCPT_INVALID_INPUT - Raised when the invalid input is passed into this
method * EXCPT_ENTITY_NOT_FOUND - Raised when objectPath does not reference an
existing object
* EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators
that the EMS
* can support has been reached.
* EXCPT_NE_COMM_LOSS - Raised when communication to managedElement is lost
**/
void getSupportedActions(
    in globaldefs::NVList_T      objectPath,
    outUnspRXCustomerActionTypeList  actionTypesList)
    raises(globaldefs::ProcessingFailureException);

};
};
#endif

```


Sample Client Program

B

This appendix provides a sample NMS client Java program.

```
import org.tmforum.mtnm.globaldefs.*;
import org.tmforum.mtnm.common.*;
import org.tmforum.mtnm.session.*;
import org.tmforum.mtnm.emsSession.*;
import org.tmforum.mtnm.nmsSession.*;
import org.tmforum.mtnm.emsSessionFactory.*;

import org.omg.CORBA.*;
import com.redstonecom.services.corbaevents.CorbaApi;
import
    com.redstonecom.services.provisioningservice.unspcorbabridge.tmf814.nmsSession.Tmf814NmsSessionImpl;
import
    com.redstonecom.services.provisioningservice.unspcorbabridge.nmcx.unspRXDeviceMgr.*
    ;
import
    com.redstonecom.services.provisioningservice.unspcorbabridge.nmcx.unspRXCustomerMgr.*
    .*;

import java.util.*;
import java.lang.*;
import java.text.SimpleDateFormat;
import java.io.*;

class NmcxPSSampleClient
{
    private CorbaApi corbaApi = CorbaApi.getInstance();

    private String[] cmd_line_args;
    private java.util.Properties properties;
```

```
EmsSession_I emsSession_I;
EmsSessionFactory_I emsSessionFactory_I;

static public void main( String[] args )
{
    NmcrcxPSSampleClient client = new NmcrcxPSSampleClient( args, null );

    client.start();
    if ( client.emsSessionFactory_I != null )
    {
        client.initialRequests();
        System.exit(0);
    }
}

private NmcrcxPSSampleClient( String[] args, java.util.Properties properties )
{
    this.cmd_line_args = args;
    this.properties = properties;
}

private void start()
{
    Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
    corbaApi.connectToORB( cmd_line_args, properties );
    corbaApi.connectToPOA();
    try
    {
        // Get the Name Service object:
        // Connecting to Name Service
        corbaApi.connectToNameServiceByIORFile("JuniperERXNameService.ior");
        // getEmsSessionFactoryServer
        emsSessionFactory_I = getEmsSessionFactoryServer();
    }
    catch ( Exception e )
    {
        System.out.println(
            "Connecting to Name Service failed : " + e.getMessage());
    }
}

private EmsSessionFactory_I getEmsSessionFactoryServer()
{
    try
    {
```

```
        org.omg.CORBA.Object Obj = corbaApi.findObjectForName(
            "JUNIPER_NMCRX_TMF814_EMSSSESSIONFACTORY_8384" );
        return EmsSessionFactory_IHelper.narrow( Obj );
    }
    catch ( Exception e )
    {
        System.out.println( "EmsSessionFactory Server will not be available." );
        return null;
    }
}

private void initialRequests()
{
    try{
        // Create NmsSession client_nms to pass to EmsSession
        org.omg.CORBA.Object Obj;
        Tmf814NmsSessionImpl tmf814NmsSessionServer =
            new Tmf814NmsSessionImpl(cmd_line_args);
        // Export NmsSession
        Obj = tmf814NmsSessionServer.export(
            "TMF814_NMSSESSION", false );

        NmsSession_I client_nms = NmsSession_IHelper.narrow( Obj );

        // Try to get emaSession by call getEmsSession of EmsSessionFactory
        EmsSession_IHolder emsSession_IHolder = new EmsSession_IHolder();
        emsSessionFactory_I.getEmsSession(
            "admin", "nmc-rxadmin", client_nms, emsSession_IHolder);

        // set EmsSession in NmsSession, this is not standard method, but only use it
        at local
        tmf814NmsSessionServer.setAssociatedSession(emsSession_IHolder.value);

        emsSession_I = EmsSession_IHelper.narrow(emsSession_IHolder.value);

        Common_IHolder common_IHolder = new Common_IHolder();

        // Try to get Device Manager
        emsSession_I.getManager("UNSPRXDEVICEMGR", common_IHolder );
        UnsprXDeviceMgr_I deviceMgr =
            UnsprXDeviceMgr_IHelper.narrow(common_IHolder.value);

        // Try to get Customer Manager
        emsSession_I.getManager("UNSPRXCUSTOMERMGR", common_IHolder );
        UnsprXCustomerMgr_I customerMgr =
            UnsprXCustomerMgr_IHelper.narrow(common_IHolder.value);
    }
}
```

```
// ping EMS Corba Server
emsSession_I.ping();

String IP = "65.194.140.194";
String ptp = "5/0";

NameAndStringValue_T[] objectPath = new NameAndStringValue_T[3];
objectPath[0] = new NameAndStringValue_T("EMS", "JUNIPER_NMCRX");
objectPath[1] = new NameAndStringValue_T("ManagedElement", IP);
objectPath[2] = new NameAndStringValue_T("PTP", ptp);

this.testListAllByDeviceMgr(deviceMgr, IP, ptp, "8", "2.0");
this.testListAllByCustomerMgr(customerMgr, "ABC1's");

deviceMgr = null;
emsSession_I.endSession();
emsSession_I = null;
emsSessionFactory_I = null;
}
catch(ProcessingFailureException e)
{
    System.out.println("ProcessingFailureException: " + e.errorReason );
}
catch(Exception e)
{
    System.out.println("testEmsSessionFactory failed: " + e.getMessage());
}
}

private void testListAllByDeviceMgr(UnspRXDeviceMgr_I deviceMgr, String IP, String
ptp, String index, String name)
{
    this.ListAllNode(deviceMgr);
    this.ListAllATM(deviceMgr, IP, ptp);
    this.ListAllATMSUB(deviceMgr, IP, ptp);
    this.ListAllPVC(deviceMgr, IP, ptp, index);
}

private void ListAllNode(UnspRXDeviceMgr_I deviceMgr)
{
    NameAndStringValue_T[] objectPath = new NameAndStringValue_T[1];
    NamingAttributesList_THolder namingAttributesList_THolder =
        new NamingAttributesList_THolder();
    NamingAttributesIterator_IHolder namingAttributesIterator_IHolder =
        new NamingAttributesIterator_IHolder();
```

```
objectPath[0] = new NameAndStringValue_T("EMS", "JUNIPER_NMCRX");

try{
    deviceMgr.listAll( objectPath,
                      UnsprXDeviceMgrObjectType.ManagedElement,
                      10,
                      namingAttributesList_THolder,
                      namingAttributesIterator_IHolder);
}
catch(ProcessingFailureException e)
{
    System.out.println("ProcessingFailureException: " + e.errorReason );
}
}

private void ListAllATM(UnsprXDeviceMgr_I deviceMgr, String IP, String ptp)
{
    NameAndStringValue_T[] objectPath = new NameAndStringValue_T[3];
    NamingAttributesList_THolder namingAttributesList_THolder =
        new NamingAttributesList_THolder();
    NamingAttributesIterator_IHolder namingAttributesIterator_IHolder =
        new NamingAttributesIterator_IHolder();

    objectPath[0] = new NameAndStringValue_T("EMS", "JUNIPER_NMCRX");
    objectPath[1] = new NameAndStringValue_T("ManagedElement", IP);
    objectPath[2] = new NameAndStringValue_T("PTP", ptp);

    try{
        deviceMgr.listAll( objectPath,
                          UnsprXDeviceMgrObjectType.AtmInterface,
                          10,
                          namingAttributesList_THolder,
                          namingAttributesIterator_IHolder);

        org.tmforum.mtnm.globaldefs.NamingAttributesIterator_I nai =
            namingAttributesIterator_IHolder.value;
    }
    catch(ProcessingFailureException e)
    {
        System.out.println("ProcessingFailureException: " + e.errorReason );
    }
}

private void ListAllATMSUB(UnsprXDeviceMgr_I deviceMgr, String IP, String ptp)
{
```

```

NameAndStringValue_T[] objectPath = new NameAndStringValue_T[4];
NamingAttributesList_THolder namingAttributesList_THolder =
    new NamingAttributesList_THolder();
NamingAttributesIterator_IHolder namingAttributesIterator_IHolder =
    new NamingAttributesIterator_IHolder();

objectPath[0] = new NameAndStringValue_T("EMS", "JUNIPER_NMCRX");
objectPath[1] = new NameAndStringValue_T("ManagedElement", IP);
objectPath[2] = new NameAndStringValue_T("PTP", ptp);
objectPath[3] = new NameAndStringValue_T("AtmInterface", "ATM" + ptp);

try{
    deviceMgr.listAll( objectPath,
                      UnsprRXDeviceMgrObjectType.AtmsubInterface,
                      10,
                      namingAttributesList_THolder,
                      namingAttributesIterator_IHolder);

    org.tmforum.mtnm.globaldefs.NamingAttributesIterator_I nai =
        namingAttributesIterator_IHolder.value;
}
catch(ProcessingFailureException e)
{
    System.out.println("ProcessingFailureException: " + e.errorReason );
}
}

private void ListAllPVC(UnsprRXDeviceMgr_I deviceMgr, String IP, String ptp, String
Index)
{
    NameAndStringValue_T[] objectPath = new NameAndStringValue_T[5];
    NamingAttributesList_THolder namingAttributesList_THolder =
        new NamingAttributesList_THolder();
    NamingAttributesIterator_IHolder namingAttributesIterator_IHolder =
        new NamingAttributesIterator_IHolder();

    objectPath[0] = new NameAndStringValue_T("EMS", "JUNIPER_NMCRX");
    objectPath[1] = new NameAndStringValue_T("ManagedElement", IP);
    objectPath[2] = new NameAndStringValue_T("PTP", ptp);
    objectPath[3] = new NameAndStringValue_T("AtmInterface", "ATM" + ptp);
    objectPath[4] = new NameAndStringValue_T("AtmSubInterface", "ATM" + ptp + "." +
Index);

    try{
        deviceMgr.listAll( objectPath,
                          UnsprRXDeviceMgrObjectType.AtmPvc,
                          10,

```

```
        namingAttributesList_THolder,
        namingAttributesIterator_IHolder);

        org.tmforum.mtnm.globaldefs.NamingAttributesIterator_I nai =
            namingAttributesIterator_IHolder.value;
    }
    catch(ProcessingFailureException e)
    {
        System.out.println("ProcessingFailureException: " + e.errorReason );
    }
}

private void testListAllByCustomerMgr(UnsprXCustomerMgr_I customerMgr, String name)
{
    this.ListAllCustomerByCustomer(customerMgr);
    this.ListAllCustSiteByCustomer(customerMgr, name);
}

private void ListAllCustomerByCustomer(UnsprXCustomerMgr_I customerMgr)
{
    NameAndStringValue_T[] objectPath = new NameAndStringValue_T[1];
    NamingAttributesList_THolder namingAttributesList_THolder =
        new NamingAttributesList_THolder();
    NamingAttributesIterator_IHolder namingAttributesIterator_IHolder =
        new NamingAttributesIterator_IHolder();

    objectPath[0] = new NameAndStringValue_T("EMS", "JUNIPER_NMCRX");

    try{
        customerMgr.listAll( objectPath,
                            UnsprXCustomerMgrObjectType.Customer,
                            10,
                            namingAttributesList_THolder,
                            namingAttributesIterator_IHolder);

        org.tmforum.mtnm.globaldefs.NamingAttributesIterator_I nai =
            namingAttributesIterator_IHolder.value;
    }
    catch(ProcessingFailureException e)
    {
        System.out.println("ProcessingFailureException: " + e.errorReason );
    }
}

private void ListAllCustSiteByCustomer(UnsprXCustomerMgr_I customerMgr, String
customerName)
```

```
{
    NameAndStringValue_T[] objectPath = new NameAndStringValue_T[2];
    NamingAttributesList_THolder namingAttributesList_THolder =
        new NamingAttributesList_THolder();
    NamingAttributesIterator_IHolder namingAttributesIterator_IHolder =
        new NamingAttributesIterator_IHolder();

    objectPath[0] = new NameAndStringValue_T("EMS", "JUNIPER_NMCRX");
    objectPath[1] = new NameAndStringValue_T("Customer", customerName );

    try{
        customerMgr.listAll( objectPath,
                            UnsprXCustomerMgrObjectType.CustSite,
                            10,
                            namingAttributesList_THolder,
                            namingAttributesIterator_IHolder);

        org.tmforum.mtnm.globaldefs.NamingAttributesIterator_I nai =
            namingAttributesIterator_IHolder.value;
    }
    catch(ProcessingFailureException e)
    {
        System.out.println("ProcessingFailureException: " + e.errorReason );
    }
}
```