



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

## **Release Notes**

*Release 7.0.0*

**Juniper Networks, Inc.**

1194 North Mathilda Avenue  
Sunnyvale, CA 94089

USA

408-745-2000

**[www.juniper.net](http://www.juniper.net)**

Part Number: 162-01154-00, 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, 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 6,590,7855.

Copyright © 2005, Juniper Networks, Inc.  
All rights reserved. Printed in USA.  
NMC-RX™ Element Management System: Release Notes, Release 7.0.0

Writing: John Borelli  
Editing: Fran Mues

Revision History  
1 August 2005—Revision 1

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

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](http://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:
  - 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.
  - 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.
  - 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



# Table of Contents

<b>Release 7.0.0</b>	<b>1</b>
Overview .....	1
Before You Start .....	1
About Release 7.0.0 Documentation.....	2
Contacting Customer Service.....	2
Updating the NMC-RX License Keys .....	2
New Features and Enhancements .....	2
SNMPv2c and SNMPv3 .....	2
Installation Information.....	3
Known Problems .....	3
Known Limitations.....	3
Troubleshooting .....	4
Fixed Problems .....	4
<b>Provisioning Server IDLs</b>	<b>7</b>
unspRXDeviceMgr.idl.....	7
Provisioning Objects.....	14
Managed Element .....	14
PTP .....	15
Port (OC3/OC12).....	16
Port (T3 ATM) .....	16
ATM Interface .....	17
ATM Subinterface.....	18
ATM PVC.....	20
Virtual Router.....	21
IP Interface .....	22
Customer and Customer Site.....	23
ATM Bridge Support.....	23
Bridge Interface.....	24
unspRXCustomerMgr.idl .....	25
Provisioning Objects.....	28
Customer .....	28
Customer Site.....	28



# Release 7.0.0

These *Release Notes* are for NMC-RX Element Management System Release 7.0.0. Unless specified otherwise, information in these *Release Notes* pertains to both the Windows and Sun Solaris versions of the release 7.0.0 software.



**NOTE:** If the information in these *Release Notes* differs from the information found in the product documentation, follow these *Release Notes*.

---

- Overview on page 1
- New Features and Enhancements on page 2
- Installation Information on page 3
- Known Problems on page 3
- Known Limitations on page 3
- Troubleshooting on page 4
- Fixed Problems on page 4

## Overview

---

The NMC-RX application allows you to manage, configure, and monitor the E-series routers in your network and to communicate with them to obtain a complete and accurate picture of the network services that you provide to your customers.

## Before You Start

Before you use the NMC-RX application, we suggest that you read these *Release Notes* in their entirety, especially the sections *Known Problems* and *Known Limitations*.

### **About Release 7.0.0 Documentation**

With the NMC-RX application, you receive the following documentation:

- Online Help (integral to the NMC-RX application)
- A PDF version of the *NMC-RX User Guide* (Vol. 1 and Vol. 2)
- A PDF version of the *NMC-RX Release Notes* (this document)

### **Contacting Customer Service**

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

### **Updating the NMC-RX License Keys**

If you need to update your license keys after you install the NMC-RX application, choose NMC-RX Licensing from the Help menu. In the NMC-RX Licensing Information dialog box, you can change either your NMC-RX, Config Sync Services, or Provisioning Service license key.

## **New Features and Enhancements**

---

NMC-RX Release 7.0.0 includes new features and enhancements (defined in the following sections). These features have been added to the NMC-RX application since Release 6.1.x.

This release is intended to work with JUNOS Releases 7.0.x and 6.1.x. This release is also compatible with JUNOS Releases 6.0.x and 5.3.x.

### **SNMPv2c and SNMPv3**

You can install the SNMPv2c version or the SNMPv3 version of the NMC-RX application. Each version provides authentication and privacy for users in different ways:

- SNMPv2c—Provides password protection via a community string.
- SNMPv3—Each user is associated with a group. A group is a set of users with the same access privileges to the router. For each NMC-RX user, you can configure only one SNMP user.

## Installation Information

---

See *Chapter 2, Installing and Running the NMC-RX Application*, in the *NMC-RX User Guide, Vol. 1*.

Solaris software patches 108940-50 and 108652-66 are required to install the NMC-RX application on Solaris 2.8. To find the required patches for your system, visit:

- <http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-access>

## Known Problems

---

This section lists the known problems in release 7.0.0:

- When you use Add/Remove Programs in Windows to remove the Juniper Networks NMC-RX application, all Config Sync Services are also removed. However, entries for the Config Sync Services still appear in the Add/Remove Programs dialog box.

**Workaround:** Close, then reopen the Add/Remove Programs dialog box to refresh it and confirm that the Config Sync Services entries no longer appear.

- An SNMP error occurs when you try to configure an IP static route and select a nonbroadcast multiaccess (NBMA) or broadcast IP interface as the next hop. Only IP interfaces with a category of point-to-point can be used as the next hop for an IP static route.
- IP interfaces that you choose for any forward or next interface rules within a policy list must exist on the same virtual router. An SNMP error occurs if you add a forward or next interface rule to the policy list that specifies an IP interface from a different virtual router.

## Known Limitations

---

This section lists the known limitations in Release 7.0.0:

- When you click Install on the Pre-Install Summary dialog box during installation, there is a short amount of time when the database updates. If you click Cancel during this time, the database remains locked for the installation of additional Config Sync Services.

**Recommendation:** Do not click Cancel after the installation begins. Instead, allow the installation to complete, and then use the NMC-RX Uninstaller or Config Sync Services uninstaller to remove the desired components.

- When you use Bulk Services to create a large number of objects, limit the number of objects to a maximum of 9000. Otherwise, you run the risk of running out of memory. To create additional objects, you can exit the NMC-RX application and then restart.

- Currently, the NMC-RX application allows you to start multiple Polling Services at the same time. Only one Polling Service is actually used by the application. There will be no disruption of service if you start additional services; however, it is a waste of resources to do so, and currently no error message is displayed to indicate any displacement of the service being used. All instances of polling should be closed, and only one restarted.
- When the Config Sync Services and the Polling Service are started before the database has completely initialized, an error occurs.

**Recommendation:** Wait until the database is initialized before launching other NMC-RX components. If an error has occurred, close all NMC-RX components (including the database) and start over.

- An error may occur if, during a device update, you attempt to make changes to a scheduled task and save them.

**Workaround:** Wait until the device has been updated, and then edit the scheduled tasks.

## Troubleshooting

---

If any of the following conditions appear, try the suggested workaround(s).

- Two causes prevent WebHelp from starting on Solaris:
  - You have not defined the Netscape path in the PATH variable. This results in an error message being displayed in the NMC-RX window.
- You may not have permission to connect to the X server. This prevents Netscape from being started. When this happens, no error message is generated, and you may think that the online help does not work properly.

**Workaround:** Define the Netscape path.

**Workaround:** Enter the command `xhost < hostname >` in a term window. Doing this disables the X server security and allows the Netscape browser to be displayed.

## Fixed Problems

---

The following problems reported in previous releases have been fixed:

- Config Sync Services FTP directories on Solaris are not created with read/write privileges.

**Workaround:** After starting a Config Sync Service for the first time, and before performing a device discovery, execute the following command:

```
chmod -R 777 <local FTP root directory>/<FTP subdirectory>
```

Note that `< local FTP root directory >` and `< ftp subdirectory >` are the directories set during installation.

- If you edit the Integrated Local Management Interface (ILMI) settings (virtual path identifier [VPI] and virtual circuit identifier [VCI]) of an ATM interface while the ILMI Settings Admin Status is set to “Up,” an SNMP error occurs.

**Workaround:** Before changing VPI and VCI settings, change the ILMI Settings Admin Status to “Down,” and click Save. Next, update the VPI and VCI settings, and click Save. Then, change the ILMI Settings Admin Status back to “Up,” and click Save.

- You cannot create Frame Relay major interfaces for devices running JUNOS release 6.1.0 and release 7.0.0b. You can create Frame Relay major interfaces for devices running JUNOS release 6.0.0 or earlier.
- When statistics are running, if an SNMP error is received on another operation (create, for instance) and you do not click the OK button on the error message dialog box within a certain number of seconds, an SNMP timeout error is returned for the SNMP statistics get request. The same result occurs if the SNMP error is on one device and the statistics polling is occurring on another.

After you click OK on the error dialog box, everything returns to normal with the next poll.

- When configuring an interface, if you display the Module Config tab (by clicking the Lower Layer button) and then close it (by clicking the X button), all other tabs are closed and the original interface remains locked.
- The OAM Loopback Enable and OAM Loopback Frequency parameters are read-only when you create and configure ATM interfaces using the NMC-RX client and NMC-RX Provisioning Service for devices that are running JUNOS release 6.1.0 . You can configure these parameters for devices running JUNOS release 6.0.0 or earlier.
- In previous releases, the PVS test utility was not reading the EMS name and port number from the PVS.rc file and did not work correctly if the values were changed in the file. The PVS test utility now correctly uses the EMS name and port number in the PVS.rc file.



## Appendix A

# Provisioning Server IDLs



**NOTE:** Appendix A, *Provisioning Server IDLs* in *NMC-RX Element Management System User Guide, Vol. 2* has been updated to include additional information on provisioning objects for the provisioning server IDLs.

This appendix describes the Juniper Networks proprietary IDL set for the Provisioning Service. The IDLs define all of the methods for the network management system (NMS) to manage Juniper Networks devices. To get these references, the NMS must go through the `getManager(...)` of `emsSession` reference.

The provisioning objects of each IDL are also described.

This appendix contains the following sections:

- `unspRXDeviceMgr.idl` on page 7
- `unspRXCustomerMgr.idl` on page 25

## `unspRXDeviceMgr.idl`

The following IDL is the definition of `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
EMS,
ManagedElement,
PTP,

AtmInterface,
AtmSubInterface,
AtmPvc,

PPP,
PPPoE,
PPPoESubInterface,
BridgedIP_rfc1483,
VirtualRouter,

// Examples:
// "Juniper NMC-RX"
// "10.6.129.6"
// "6/0" - where 6 is the slot number, and
// 0 is the port number
// "ATM6/0-atm-layer",
// "ATM6/0.1-atm1483-subif",
// "ATM6/0.1-atm1483-subif vpi=1,vci=600"

// not supported
// not supported
// not supported
// "4/0.10"
// "default", "VR1"

IpInterface,

Customer,
CustSite,
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:
*      UnsprRXDeviceMgrObjectType.AtmSubInterfaces
*      UnsprRXDeviceMgrObjectType.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 UnsprRXDeviceMgrObjectType 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 UnsprRXDeviceMgrObjectType 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
*
*      *
*      *
*      *      objectName: The object name has just been created by this method. It is assigned
*                    by device instead of being specified by the user
*
*      *
*      *      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 create(
    in globaldefs::NVList_T      objectPath,
    in UnsprXDeviceMgrObjectType objectType,
    in globaldefs::NVList_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::NVList_T      objectPath,
    in globaldefs::NVList_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 getConfigAttributesDescription(
    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 getListableObjectTypeSelections(
    in globaldefs::NVSLIST_T      objectPath,
    out UnsprXDeviceMgrObjectTypeList 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::NVSLIST_T      objectPath,
    out UnsprXDeviceMgrObjectTypeList 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 actionTypeList)
    raises(globaldefs::ProcessingFailureException);

};
};
#endif

```

## Provisioning Objects

The following sections describe the provisioning objects for this IDL.

### Managed Element

A managed element is an E-series device or node.

- Sample Object Path
  - Name = EMS; Value = JUNIPER\_NMCRX
  - Name = ManagedElement; Value = 10.10.10.10
- Associated MIBs
  - juniSystem.mi2
  - rfc1213.mib

**Table 1: Attributes**

Attribute	Characteristic	Description	MIB Entry
SYS_CONTACT	View, configure	Contact person for the device	sysContact
SYS_LOCATION	View	Physical location of the device	sysLocation
SYS_SWVERSION	View	Software version currently running on the device	juniSystemSwVersion
SYS_SW_BUILD_DATE	View	Build date of the software currently running on the system	juniSystemSwBuildDate
SYS_UPTIME	View	Current system uptime for the device	sysUpTime
SYS_DEVICE_TYPE	View	Type of device	None

Values:

- 1 = ERX 700
- 2 = ERX 1400

**Table 1: Attributes (continued)**

Attribute	Characteristic	Description	MIB Entry
ERX_NODE_SYS_NAME	View	ERX node system name for the device  The name is displayed on the prompt of the command-line interface. It is not the same as the name assigned by the NMC-RX user when initially discovering the node.	sysName

## PTP

The Physical Termination Point (PTP) object represents either a module specified at a particular slot, or a line interface specified as a slot and port.

- Sample Object Path

The object path for a module includes the slot number.

- Name = EMS; Value = JUNIPER\_NMCRX
- Name = ManagedElement; Value = 10.10.10.10
- Name = PTP; Value = 0

The object path for a line interface or port includes the slot number and the port number.

- Name = EMS; Value = JUNIPER\_NMCRX
- Name = ManagedElement; Value = 10.10.10.10
- Name = PTP; Value = 0/0

- Associated MIBs

- juniSystem.mi2

**Table 2: Attributes**

Attribute	Characteristic	Description	MIB Entry
ADMIN_STATUS	Configure	Administrative status of the module	juniSystemModuleAdminStatus.<slotNumber>.1
ENTITY_INDEX	View	SNMP Index used to represent the object in the entity mib	entPhysicalIndex
IF_INDEX	View	Index used for SNMP management	ifIndex
SERIAL_NUMBER	View	Module Serial Number	juniSystemModuleSerialNumber.<slotNumber>.1
IOA_SERIAL_NUMBER	View	Module IOA Serial Number	juniSystemModuleSerialNumber.<slotNumber>.2
MODULE_TYPE	View	Module type	juniSystemModuleCurrentType.<slotNumber>.1

**Port (OC3/OC12)**

This type of provisioning object has the following associated MIBs:

- rfc1213.mib
- junisonet.mi2

**Table 3: Attributes**

Attribute	Characteristic	Description	MIB Entry
IF_INDEX	View	Index used for SNMP management	ifIndex
NAME	View	Name of the interface	ifDescr
DESCRIPTION	Configure	Logical description of the interface	ifAlias
ADMIN_STATUS	Configure	Administrative status of the interface	ifAdminStatus
OPER_STATUS	View	Operational status of the interface	ifOperStatus
SONET_INDEX	View	Sonet SNMP Index used to represent the object	ifIndex
LOOPBACK	Configure	Specifies the loopback mode for the interface	juniSonetMediumLoopbackConfig
SYNC_STANDARD	Configure	Specifies the Sync Standard (SDH or Sonet) to be used for the interface	juniSonetMediumType
XMIT_CLOCK	Configure	Specifies the Transmit Clock Source to use for the interface	juniSonetMediumTimingSource
LINE_INTERFACE_TYPE	View	Line interface type	3 = OC3 2-port 12 = OC12 ATM 1-port 14 = OC3 ATM 4-port

**Port (T3 ATM)**

This type of provisioning object has the following associated MIBs:

- rfc1213.mib
- junids3.mi2
- rfc1407.mib

**Table 4: Attributes**

Attribute	Characteristic	Description	MIB Entry
IF_INDEX	View	Index used for SNMP management	ifIndex
NAME	View	Name of the interface	ifDescr
DESCRIPTION	Configure	Logical description of the interface	ifAlias
ADMIN_STATUS	Configure	Administrative status of the interface	ifAdminStatus
OPER_STATUS	View	Operational status of the interface	ifOperStatus
LINE_BUILDOUT	Configure	Specifies the line build out in meters for the interface	juniDs3LineLength
FRAMING_TYPE	Configure	Specifies the framing to be used for the interface	juniDs3LineType
LOOPBACK	Configure	Specifies the loopback mode for the interface	juniSonetMediumLoopbackConfig

**Table 4: Attributes (continued)**

Attribute	Characteristic	Description	MIB Entry
SCRAMBLE	Configure	Specifies the Scramble mode to use for the interface	juniDs3CellScramblerConfig
XMIT_CLOCK	Configure	Specifies the Transmit Clock Source to use for the interface	dsx3TransmitClockSource
LINE_INTERFACE_TYPE	View	Line interface type	4 = T3 ATM 3-port

## ATM Interface

The OC3-2 modules do not support MAX VPI and MAX VCI.

### ■ Sample Object Path

The object path for a module includes the slot number.

- Name = EMS; Value = JUNIPER\_NMCRX
- Name = ManagedElement; Value = 10.10.10.10
- Name = PTP; Value = 0/0
- Name = AtmInterface; Value = ATM0/0

### ■ Associated MIBs

- rfc1213.mib
- rfc1695.mib
- juniAtm.mib
- af-nm-0095\_001\_mib.mib  
(atmfM4IfLoopbackLocationCode defined in this MIB)

**Table 5: Attributes**

Attribute	Characteristic	Description	MIB Entry
IF_INDEX	View	Index used for SNMP management	ifIndex
NAME	View	Name of the interface	ifDescr
DESCRIPTION	Create, configure	Logical description of the interface	ifAlias
ADMIN_STATUS	Configure	Administrative status of the module	juniSystemModuleAdminStatus. < slotNumber > . 1
OPER_STATUS	View	Operational status of the interface	ifOperStatus
MAX_VPI_BITS	Create	Maximum configurable value for a VPI on the interface	atmInterfaceMaxActiveVpiBits
MAX_VCI_BITS	Create	Maximum configurable value for a VCI on the interface	atmInterfaceMaxActiveVciBits
ILMI_VPI	Create, configure	VPI of the ILMI of a major ATM interface	juniAtmIfIlmiVpi
ILMI_VCI	Create, configure	VCI of the ILMI of a major ATM interface	juniAtmIfIlmiVci

**Table 5: Attributes (continued)**

Attribute	Characteristic	Description	MIB Entry
ILMI_VCD	Create	An integer identifier for the ILMI, used in conjunction with the command-line interface. The value must be unique among VCs configured on the same ATM interface.	juniAtmIfIlmiVcd
ILMI_ADMIN_STATUS	Create, configure	Administrative status of ILMI in the ATM interface	juniAtmIfIlmiAdminState
ILMI_POLL_FREQUENCY	Create, configure	The amount of time in seconds between successive transmissions of ILMI messages on the interface for the purpose of detecting loss of ILMI connectivity.  The distinguished value zero disables ILMI connectivity procedures on the interface.	juniAtmIfIlmiPollFrequency
UNI_VERSION	Create, configure	Use to specify the User Network Interface (UNI) the router should use when ILMI link auto determination is unsuccessful or ILMI is disabled.	juniAtmIfUniVersion
OAM_ADMIN_STATE	Create, configure	Administrative status of OAM for the interface	juniAtmIfOamCellRxAdminState
CAC_ADMIN_STATE	Create, configure	Administrative status of CAC on the ATM major interface	juniAtmIfCacAdminState
CAC_UBR_WEIGHT	Create, configure	Bandwidth associated with every UBR and UBR with PCR connection configured on the ATM major interface	juniAtmIfCacUbrWeight
CAC_SUBSCRIPTION_BANDWIDTH	Create, configure	The subscribed bandwidth of the ATM major interface. If this value is not specified or set to 0, the effective port bandwidth is used.	juniAtmIfCacSubscriptionBandwidth
CAC_AVAILABLE_BANDWIDTH	View	Available bandwidth of the ATM major interface	juniAtmIfCacAvailableBandwidth
LOOPBACK_LOCATION_ID	Create, configure	Specifies the code that shall exist in incoming OAM Loopback cells that are to be looped back at the interface	atmfM4IfLoopbackLocationCode

## ATM Subinterface

### ■ Sample Object Path

The object path for a module includes the slot number.

- Name = EMS; Value = JUNIPER\_NMCRX
- Name = ManagedElement; Value = 10.10.10.10
- Name = PTP; Value = 0/0
- Name = AtmInterface; Value = ATM0/0
- Name = AtmSubInterface; Value = ATM0/0.33

- Associated MIBs
  - rfc1213.mib
  - juniAtm.mi2
  - juniAutoconf.mi2
  - juniSubscriber.mi2
  - juniTpl.mi2
  - juniQos.mi2

**Table 6: Attributes**

Attribute	Characteristic	Description	MIB Entry
IF_INDEX	View	The index used for SNMP management	ifIndex
NAME	View	The name of the interface	ifDescr
DESCRIPTION	Create, configure	The logical description of the interface	ifAlias
ADMIN_STATUS	Create, configure	Administrative status of the interface	ifAdminStatus
OPER_STATUS	View	Operational status of the interface	ifOperStatus
IP_AUTOCONFIG	Create, configure	Enables the auto configuration feature for the IP layer	juniAutoConfEnable.ifIndex.ip
PPP_AUTOCONFIG	Create, configure	Enables the auto configuration feature for the PPP layer	juniAutoConfEnable.ifIndex.ppp
PPPOE_AUTOCONFIG	Create, configure	Enables the auto configuration feature for the PPPoE layer	juniAutoConfEnable.ifIndex.pppoe
SUBSCRIBER_ENABLE	Create, configure	Enables appending of subscriber information for the IP layer	juniSubscrLocalControl
SUBSCRIBER_NAME_PREFIX	Create, configure	Indicates whether the Subscriber Name is a prefix rather than a full name	juniSubscrLocalNamePrefix
SUBSCRIBER_PASSWORD_PREFIX	Create, configure	Indicates whether the Subscriber Password is a prefix rather than a full password	juniSubscrLocalPasswordPrefix
IP_PROFILE_NAME	Create, configure	Name of the Profile to associate with the IP Encapsulation	None
IP_PROFILE_INDEX	View	SNMP Index of the Profile to associate with the IP Encapsulation	juniProfAssignIfProfileId.ifIndex.ip
PPP_PROFILE_NAME	Create, configure	Name of the Profile to associate with the PPP Encapsulation	None
PPP_PROFILE_INDEX	View	SNMP Index of the Profile to associate with the PPP Encapsulation	juniProfAssignIfProfileId.ifIndex.ppp
PPPOE_PROFILE_NAME	Create, configure	Name of the Profile to associate with the PPPoE Encapsulation	None

**Table 6: Attributes (continued)**

Attribute	Characteristic	Description	MIB Entry
PPPOE_PROFILE_INDEX	View	SNMP Index of the Profile to associate with the PPPoE Encapsulation	juniProfAssignIfProfileId.ifIndex.pppoe
ANY_PROFILE_NAME	Create, configure	Name of the Profile to be used as a wildcard	None
ANY_PROFILE_INDEX	View	SNMP Index of the Profile to be used as a wildcard	juniProfAssignIfProfileId.ifIndex.any
SUBSCRIBER_NAME	Create, configure	Local name that distinguishes the subscriber	juniSubscrLocalName
SUBSCRIBER_PASSWORD	Create, configure	Local password to be used for the subscriber	juniSubscrLocalPassword
SUBSCRIBER_DOMAIN	Create, configure	Subscriber's domain	juniSubscrLocalDomain
QOS_PROFILE_INDEX	View	SNMP index of the attached QoS Profile	juniQosIfAttachQosProfileIndex
QOS_PROFILE_NAME	Create, configure	Name of the attached QoS Profile	None

## ATM PVC

- Sample Object Path
  - Name = EMS; Value = JUNIPER\_NMCRX
  - Name = ManagedElement; Value = 10.10.10.10
  - Name = PTP; Value = 0/0
  - Name = AtmInterface; Value = ATM0/0
  - Name = AtmSubInterface; Value = ATM0/0.33
  - Name = AtmPVC; Value = ATM0/0.33.44.55
- Associated MIBs
  - juniAtm.mi2

**Table 7: Attributes**

Attribute	Characteristic	Description	MIB Entry
NAME	View	Name of the interface	ifDescr
VPI	Create	Virtual path identifier	INDEX
VCI	Create	Virtual circuit identifier	INDEX
AAL5_ENCAP	Create	Encapsulation for the ATM Adaptation Layer 5 (AAL5)	juniAtmSubIfVccType
MBS	Create, configure	Maximum burst size (in cells)	juniAtmSubIfVccMbs
PCR	Create, configure	Peak cell rate (kbps)	juniAtmSubIfVccPcr
SCR	Create, configure	Sustainable cell rate (kbps)	juniAtmSubIfVccScr

**Table 7: Attributes (continued)**

Attribute	Characteristic	Description	MIB Entry
SERVICE_CATEGORY	Create, configure	Servicecategory	juniAtmSubIfVccServiceCategory
OAM_STATUS	Create, configure	Enables generation of OAM F5 loopback cells	juniAtmCircuitOamAdminStatus
OAM_LOOPBACK	Create, configure	Time interval in seconds between transmissions of OAM F5 loopback cells	juniAtmCircuitOamLoopbackFrequency
VCD	View	Virtual circuit descriptor	juniAtmSubIfVccVcd

## Virtual Router

- Sample Object Path
  - Name = EMS; Value = JUNIPER\_NMCRX
  - Name = ManagedElement; Value = 10.10.10.10
  - Name = VirtualRouter; Value = default
  - Name = AtmPVC; Value = ATM0/0.33.44.55
- Associated MIBs
  - junRouter.mi2
  - junSscClient.mi2
  - junAaa.mi2
  - junRadClient.mi2
  - junDhcp.mi2

**Table 8: Attributes**

Attribute	Characteristic	Description	MIB Entry
VRROUTER_NAME	View	Name of the virtual router	juniRouterName
SSCC_PRIMARY_PORT	View	TCP port number for the primary SSCC server. A value of zero indicates the port is unconfigured.	juniSscClientPrimaryPort
SSCC_SECONDARY_PORT	View	TCP port number for the secondary SSCC server. A value of zero indicates the port is unconfigured.	juniSscClientSecondaryPort
SSCC_TERTIARY_PORT	View	TCP port number for the tertiary SSCC server. A value of zero indicates the port is unconfigured.	juniSscClientTertiaryPort
SSCC_SWITCHOVER_TIMEOUT	View	Server switchover timeout in seconds. The SSCCClient begins with the primary server and proceeds rotationally to secondary, tertiary, primary, etc., as timeouts occur.	juniSscClientServerSwitchoverTimeout

**Table 8: Attributes (continued)**

Attribute	Characteristic	Description	MIB Entry
SSCC_SNMP_READWRITE_COMMUNITY	View	SSCC SNMP Read Write community string	None
SSCC_SNMP_READ_COMMUNITY	View	The SSCC SNMP read only Community String	None
SSCC_PRIMARY_ADDRESS	View	IP address of the primary SSCC server. A value of 0.0.0.0 indicates the server address is unconfigured.	juniSscClientPrimaryAddress
SSCC_SECONDARY_ADDRESS	View	IP address of the secondary SSCC server. A value of 0.0.0.0 indicates the server address is unconfigured.	juniSscClientSecondaryAddress
SSCC_TERTIARY_ADDRESS	View	IP address of the tertiary SSCC server. A value of 0.0.0.0 indicates the server address is unconfigured.	juniSscClientTertiaryAddress
PROTOCOL_STATUS_SSCC_CLIENT	View	Status of the SSCC Client Protocol	juniRouterProtocolRowStatus

## IP Interface

IP interfaces are stacked on a wide variety of interface types and configurations. Object paths reflect these various configurations.

- Sample Object Path

The following shows the object path for an IP interface that is stacked on top of an ATM subinterface.

- Name = EMS; Value = JUNIPER\_NMCRX
- Name = ManagedElement; Value = 10.10.10.10
- Name = PTP; Value = 0/0
- Name = AtmInterface; Value = ATM0/0
- Name = AtmSubInterface; Value = ATM0/0.33
- Name = IpInterface; Value = IP0/0.33

- Associated MIBs

- junip.mi2

**Table 9: Attributes**

Attribute	Characteristic	Description	MIB Entry
IF_INDEX	View	Index used for SNMP management	ifIndex
NAME	View	Name of the interface	ifDescr
DESCRIPTION	View	Logical description of the interface	ifAlias
ADMIN_STATUS	View	Administrative status of the interface	ifAdminStatus
OPER_STATUS	View	Operational status of the interface	ifOperStatus

**Table 9: Attributes (continued)**

Attribute	Characteristic	Description	MIB Entry
CUSTOMER_NAME	View	Name of the customer	None
CUSTOMER_IDENTITY	View	Unique identifier for a particular customer	None
CUSTOMER_ACCOUNT_ID	View	Account ID for the customer	None
CUSTOMER_SITE_NAME	View	Name to represent the site for a particular customer	None
CUSTOMER_SITE_IDENTITY	View	Unique identifier to represent the site for a particular customer	None
CUSTOMER_SITE_ACCOUNT_ID	View	Account ID to represent the site for a particular customer	None
IP_CREATE_TIME	View	Timestamp of the creation of the IP Interface.  Creation is considered to be relevant to the NMC-RX application. If the interface is created via the CLI then the timestamp will reflect the first time the NMC-RX application was aware of the interface, not the time the interface was created on the CLI.	None

## Customer and Customer Site

Customer objects can be associated with IP interfaces. The Device Manager provides methods for managing customer and customer site objects. These objects are discussed in “unspRXCustomerMgr.idl” on page 25.

## ATM Bridge Support

There are two object types added to support Bridge Interfaces over ATM: BridgedIP\_rfc1483 (Bridged Ethernet) and Bridge Interface.

### ■ Sample Object Path

The following shows the object path for an IP interface that is stacked on top of an ATM subinterface.

- Name = EMS; Value = JUNIPER\_NMCRX
- Name = ManagedElement; Value = 10.10.10.10
- Name = PTP; Value = 0/0
- Name = AtmInterface; Value = ATM0/0
- Name = AtmSubInterface; Value = ATM0/0.33
- Name = BridgedIP\_rfc1483; Value = BRG-ET0/0.33

### ■ Associated MIBs

- RFC1213.mi2

**Table 10: Attributes**

Attribute	Characteristic	Description	MIB Entry
IF_INDEX	View	Index used for SNMP management	ifIndex
NAME	View	Name of the interface	ifDescr
DESCRIPTION	Create, configure	Logical description of the interface	ifAlias
ADMIN_STATUS	Create, configure	Administrative status of the interface	ifAdminStatus
OPER_STATUS	View	Operational status of the interface	ifOperStatus

## Bridge Interface

- Sample Object Path
  - Name = EMS; Value = JUNIPER\_NMCRX
  - Name = ManagedElement; Value = 10.10.10.10
  - Name = PTP; Value = 0/0
  - Name = AtmInterface; Value = ATM0/0
  - Name = AtmSubInterface; Value = ATM0/0.33
  - Name = BridgedIP\_rfc1483; Value = BRG-ET0/0.33
  - Name = BridgeInterface; Value = BridgeIf0/0.33
- Associated MIBs
  - juniBridge.mi2
  - juniBridgingMgr.mi2
  - rfc1213.mi2

**Table 11: Attributes**

Attribute	Characteristic	Description	MIB Entry
IF_INDEX	View	Index used for SNMP management	ifIndex
NAME	View	Name of the interface	ifDescr
ADMIN_STATUS	Create, configure	Administrative status of the interface	ifAdminStatus
OPER_STATUS	View	Operational status of the interface	ifOperStatus
BRIDGE_GROUP_NAME	Create	Name of the associated bridge group	juniBridgingMgrBridgeGroupName
MAX_LEARN_COUNT	Create, configure	Maximum number of MAC addresses that can be learned on the interface	juniBridgeIfMaxLearnCount
SUBSCRIBER_POLICY	Create, configure	Indicates whether the type of the interface is subscriber (value = 1) or subscriber trunk (value = 2)	juniBridgingMgrBridgeGroupSPolicy Index
BRIDGE_GROUP_INDEX	View	Index of the associated bridge group	ifIndex of the bridge group the interface is referencing

## unspRXCustomerMgr.idl

The following IDL is the definition of `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::NVList_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 getListableObjectTypeSelections(
    in globaldefs::NVSLIST_T      objectPath,
    out UnsprXCustomerMgrObjectTypeList objectTypeList)
    raises(globaldefs::ProcessingFailureException);

/**
* This method should be used to get a list of types of action that you can perform
* 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::NVSLIST_T      objectPath,
    out UnsprXCustomerActionTypeList actionTypesList)
    raises(globaldefs::ProcessingFailureException);
};
};
#endif

```

## Provisioning Objects

The following sections describe the provisioning objects for this IDL.

### Customer

- Sample Object Path
  - Name = EMS; Value = JUNIPER\_NMCRX
  - Name = Customer; Value = ACME Inc
- Associated MIBs
  - There are no associated MIBs

**Table 12: Attributes**

Attribute	Characteristic	Description	MIB Entry
CUSTOMER_NAME	View	Name of the customer	None
CUSTOMER_IDENTITY	View	Unique identifier for a particular customer	None
CUSTOMER_ACCOUNT_ID	View	Account ID for the customer	None

### Customer Site

- Sample Object Path
  - Name = EMS; Value = JUNIPER\_NMCRX
  - Name = Customer; Value = ACME Inc
  - Name = CustomerSite; Value = Anytown
- Associated MIBs
  - There are no associated MIBs

**Table 13: Attributes**

Attribute	Characteristic	Description	MIB Entry
CUSTOMER_SITE_NAME	View	Name of the customer site	None
CUSTOMER_SITE_IDENTITY	View	Unique identifier to represent this site for a particular customer	None
CUSTOMER_SITE_ACCOUNT_ID	View	Account ID for the customer site	None



