

## Chapter 11

# Using the Provisioning Service

This chapter describes how to use the NMC-RX Provisioning Service and contains the following sections:

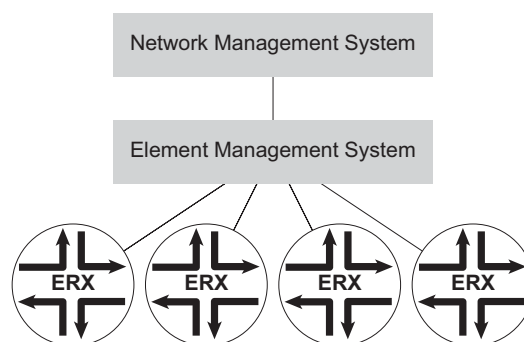
- Overview on page 149
- Starting the NMC-RX Provisioning Service on page 150
- Using the Provisioning Service on page 157
- IDL Interfaces on page 158

## Overview

---

The NMC-RX Element Management System (EMS) Provisioning Service (PVS) provides a gateway into the managed elements of that EMS. This gateway lets you interact with objects and the attributes of modeled devices. A network management system (NMS) makes requests for information or for actions on managed elements. The EMS performs the operations necessary to carry out these requests and returns the results to the NMS (Figure 7).

**Figure 7: Example of Managed Elements**



9013233

Twenty CORBA clients can be connected simultaneously to PVS.

## Starting the NMC-RX Provisioning Service

---

The NMC-RX Provisioning Service (PVS) provides an optional programmatic application programming interface (API) for integration with the NMC-RX Element Management System application.

The NMC-RX database acts as a repository of information for use by both NMC-RX client GUIs and clients that are accessing the NMC-RX application through the provisioning interface. You must install NMC-RX Element Management System software before you can run NMC-RX Provisioning Service.

### Starting the Provisioning Service

NMC-RX Provisioning Service is a separately licensed NMC-RX component. To start the Provisioning Service, you must run the following four components:

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

Three of the services—NMC-RX Database, CORBA Naming, and NMC-RX Provisioning RMI—are located in the *pvsSampleScripts* directory. For information about starting the NMC-RX Database Service, see *Chapter 2, Installing and Running the NMC-RX Application* in the *NMC-RX User Guide, Vol. 1*.

To start NMC-RX Provisioning Service:

1. (Optional—This step is an alternative to performing steps 2 through 7 and is available only on Solaris.)  
Run the **allpvs** script with or without console windows.

- a. Run **allpvs start**.

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

- b. Run **allpvs start console**.

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

2. Start the NMC-RX database.
3. To start the CORBA Naming Service, run either **naming** or **naming start console**.

An *ins.ior* file is generated and copied to the filename *JuniperERXNameService.ior*, which is located in the *pvs* directory.

- To start the NMC-RX Provisioning RMI Service, run the **nmcPvsRmiService** script.

The RMI Service is ready when you see the following message:

```
EmsSessionsFactory binded to
JUNIPER_NMCRX_TMF814_EMSSSESSIONFACTORY_8284
```

From the *PVS.rc* file in the *NMCRX\_HOME* directory, you can edit the name, “**JUNIPER\_NMCRX\_TMF814\_EMSSSESSIONFACTORY\_8384**.” 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 NMC-RX Provisioning Service CORBA Service uses the name “JUNIPER\_NMCRX\_TMF814\_EMSSSESSIONFACTORY\_8384” to find the entry reference for the RMI Service. The entry reference is the name that the CORBA client (NMS client) program uses to find the CORBA reference from the CORBA Naming Service.

- To start the NMC-RX Provisioning Service CORBA Bridge Service, run the **nmcPvsCorbaBridgeService** script.

The NMC-RX Provisioning Service software is ready for use by network management system (NMS) clients.

## Sample PVS Scripts

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 in the *PVS* directory.



**NOTE:** If a service is already running, it does not start again.

The *pvsSampleScript* directory contains the following sample scripts:

- **naming**—Starts 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**—Starts the Remote Method Invocation (RMI) Service.

- **nmcPvsCorbaBridgeService**—Starts 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**—Starts the NMC-RX Provisioning Service test utility. For additional information about this utility, see *Testing the NMC-RX Provisioning Service* on page 153.
- **allpvs** (Solaris only)—Starts all services at one time in the correct order (NMC-RX Database, Naming Service, RmiService, and CORBA Bridge Service).

For more information about the scripts, use the help option.



**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 the 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.
```

## Running Multiple PVS Services to One NMC-RX Database

You can name and run multiple uniquely named PVS Services to one NMC-RX database.

To run multiple uniquely named PVS Services to one NMC-RX database, follow these steps. Note that there must be one dedicated Provisioning Service for each CORBA client:

1. Start the Naming Service.
2. Modify the PVS.rc file by assigning a unique `emsName` and server port for the Provisioning Service.
3. Start the NMC RMI Service.
4. Start the NMC CORBA Service.
5. Start the PVS Test Utility, and confirm that the `emsName` (Provisioning Service) is up and functioning as expected. (See *Testing the NMC-RX Provisioning Service*.)
6. Modify the PVS.rc file again by assigning a different `emsName` and server port for the Provisioning Service.
7. Repeat step 3–5.

## 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 Service, 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.

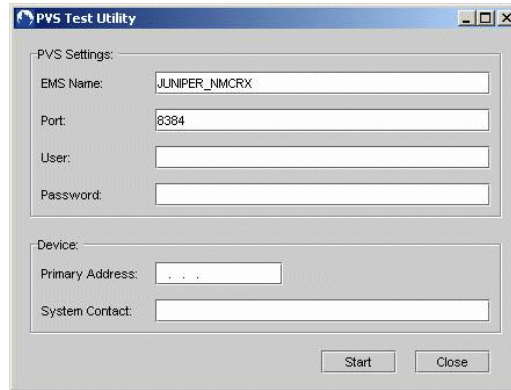
### Running the PVS Test Utility

To run the PVS test utility:

1. Start the `nmcPvsTestUtility` program from the following location:

```
< NMC-RX-HOME > /pvsSampleScripts/nmcPvsTestUtility
```

The PVS Test Utility window opens.



2. Enter the Element Management System (EMS) name and port.
3. Specify your NMC-RX username and password.
4. (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.

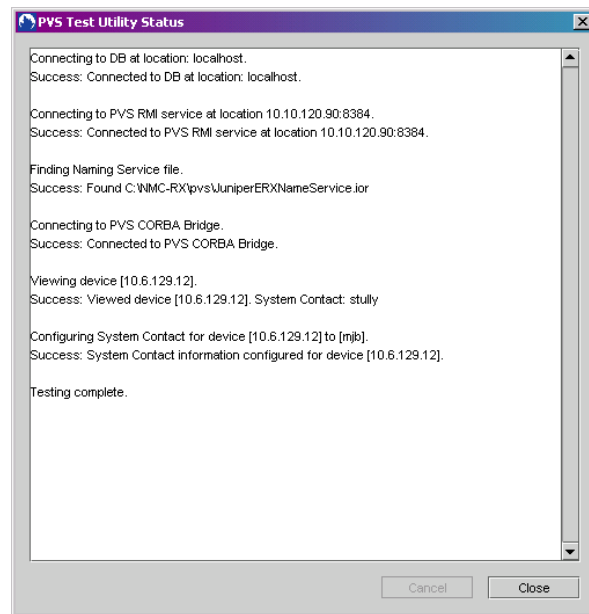
5. (Optional) Specify a system contact in the Device area to change the system contact.



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

6. Click Start.

The PVS Test Utility Status dialog box opens.



After the PVS test utility starts, you can cancel the test or close the dialog box at any time. When the PVS test utility is finished 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 the test results. Table 52 lists each test performed, the failure criteria for each test, and the potential output text for each test.

**Table 52: PVS Test Utility Output**

Test	Procedure	Failure Criteria	Messages
Database	Gets database location and user information from local NMC-RX resource file; attempts to contact database	<ul style="list-style-type: none"> <li>■ Database is not running at assigned location</li> <li>■ Database settings in NMC-RX resources are not correct</li> <li>■ Various other reasons</li> </ul>	<p>Initial status text:</p> <p>Connecting to DB at location: localhost.</p> <p>Final status text:</p> <p>Success: Connected to DB at location: localhost.</p> <p>Success: Connected to DB at location: ipAddress.</p> <p>Failure: Could not connect to DB at location: ipAddress. [Followed by exception message]</p>
RMI Service	Gets PVS port number from Provisioning Service resources file; contacts RMI Service at local IP address	<ul style="list-style-type: none"> <li>■ RMI Service is not running</li> <li>■ Various other reasons</li> </ul>	<p>Initial status text:</p> <p>Connecting to PVS RMI Service at location ipAddress:portNumber.</p> <p>Final status text:</p> <p>Success: Connected to PVS RMI Service at location ipAddress:portNumber.</p> <p>Failure: Could not connect to PVS RMI Service at location ipAddress:portNumber. [Followed by exception message]</p>

**Table 52: PVS Test Utility Output (continued)**

Test	Procedure	Failure Criteria	Messages
IOR file	Searches for <i>JuniperERXNameService.ior</i> file in PVS subdirectory	Unable to find IOR file (indicates that Naming Service has never run)	Initial status text: Finding Naming Service file. Final status text: Success: Found d:path\filename.ior. Failure: Could not find d:\path\filename.ior. [Followed by exception message]
CORBA Bridge	Uses Naming Service to contact PVS	<ul style="list-style-type: none"> <li>■ PVS CORBA Bridge is not functioning</li> <li>■ Naming Service is not running</li> <li>■ Various other reasons</li> </ul>	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]
Nonempty IP Address field	Requests information about device associated with IP address that you provide	<ul style="list-style-type: none"> <li>■ Unable to find specified device</li> <li>■ Naming Service is not running</li> <li>■ CORBA bridge is not running</li> </ul>	Initial status text: Viewing device [ipAddress]. Final status text: Success: Viewed device [ipAddress]. Failure: Could not view information for device [ipAddress]. [Followed by exception message]
Empty IP address field	Requests information about all devices on system	<ul style="list-style-type: none"> <li>■ Unable to list devices</li> <li>■ Naming Service is not running</li> <li>■ CORBA Bridge is not running</li> </ul>	Initial status text: Listing devices. Successful status text: Success: No devices found. Successful status text: Success: ipAddress1 ipAddress2 ipAddress3 ipAddress4 Status text for failure: Failure: Could not list devices. [Followed by exception text]
Nonempty IP address and nonempty System Contact field	Sets the system contact for device to specified device	<ul style="list-style-type: none"> <li>■ Unable to configure system contact value</li> </ul>	Initial status text: Configuring System Contact for device [ipAddress] to [systemContactText]. Final status text: Success: System Contact information configured for device [ipAddress]. Failure: Could not configure System Contact Information.
Status	Indicates that all tests are complete	None	Status text: Testing complete.

## Using the Provisioning Service

---

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 EMS of the vendor.
- **TMF814\_EMSSSESSIONFACTORY**—Refers to the TMF.814 EMS Session Factory object, which is the entry object for the NMS client to access the EMS.
- **8384**—Specifies the port on which the remote method invocation (RMI) server is running.

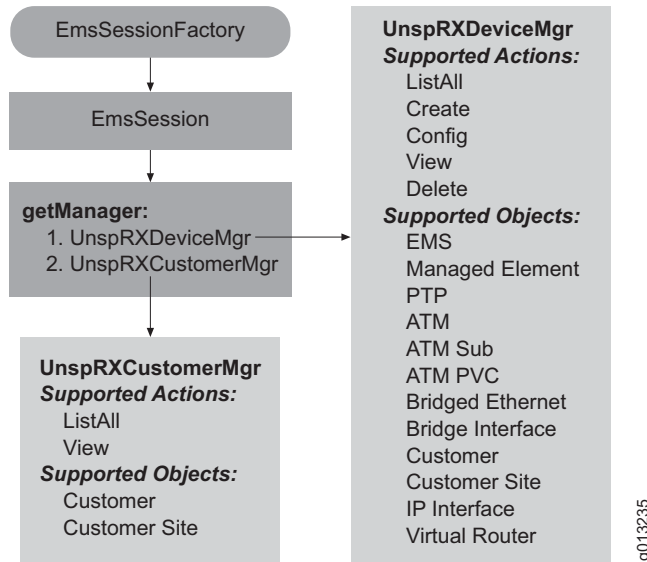
You can change the first part of the name, JUNIPER\_NMCRX, and the port number, 8384, by modifying the *provisioningserver.rc* file in the `<NMC-RX_HOME>` directory. The NMS client then retrieves the NMC-RX provisioning server entry reference *emsSessionFactory.idl* by looking up this name from the CORBA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

The standard Interface Definition Language (IDL) *emsSessionFactory.idl* is the primary class that the NMS communicates with and acts as the focal point for gaining access to managed elements. From the standard IDL *emsSession.idl* a programmer can get a specific *unspRXDeviceMgr.idl* to perform inquiries and actions against a specific device. When a particular handle to a device is formed, all operations on that device are achieved by means of 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, *unspRXCustomerMgr.idl* applies to the EMS level and does not apply to devices. For *unspRXCustomerMgr.idl*, do not set the ManagedElement object when you configure the object path.

Figure 8 illustrates the process of accessing E-series router objects and provides a listing of the objects supported in this release.

**Figure 8: NMC-RX Provisioning Service Objects**



## IDL Interfaces

The IDL solution set defines all of the interfaces between the NMS and the EMS, which enables the NMS to manage elements 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 element management layer-network management layer (EML-NML) interface. This module is intended as a common repository for definitions that are exported across modules.
- *common.idl*—Contains the definition of the common interface of the network management layer-element management layer (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 lets the client 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 IDLs so that you can easily add new manager interfaces.

- *nmsSession.idl*—Contains the definition of the `nmsSession` interface of the NML-EML interface. The `nmsSession` module lets the server inform the NMS about 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 `getEmsSession(...)` operation lets the NMS get the `EmsSession_I` object from which you can obtain all managers of the EMS.
- *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*—Defines all methods by which the NMS manages Juniper Networks devices. The NMS must go through `getManager(...)` of the `emsSession` reference to get this reference.

For more information, see *Appendix A, Provisioning Server IDLs*.

- *unspRXCustomerMgr.idl*—Defines the view and list-all methods for the NMS to manage the customer and customer site at the EMS level.

