

# 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

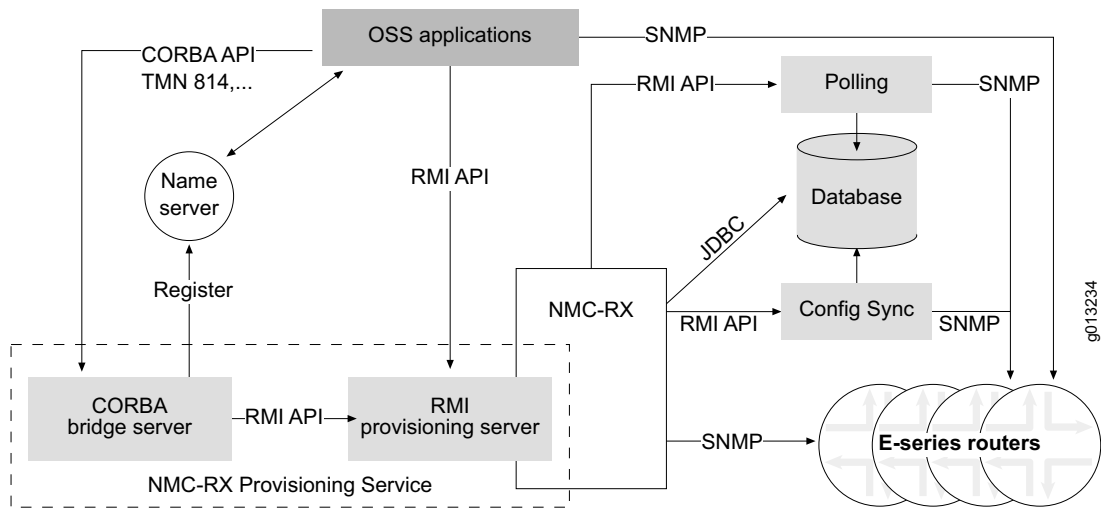
## Overview

---

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.

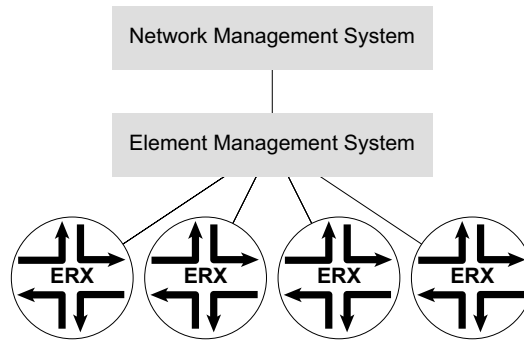
The Juniper Networks implementation of NMC-RX Provisioning Service is based on the TeleManagement Forum (TMF).814, Interface Definition Language (IDL) Solution Set model. The basics of the TMF.814 model consist of specifying an object by its object path and specifying an action to perform on this object.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

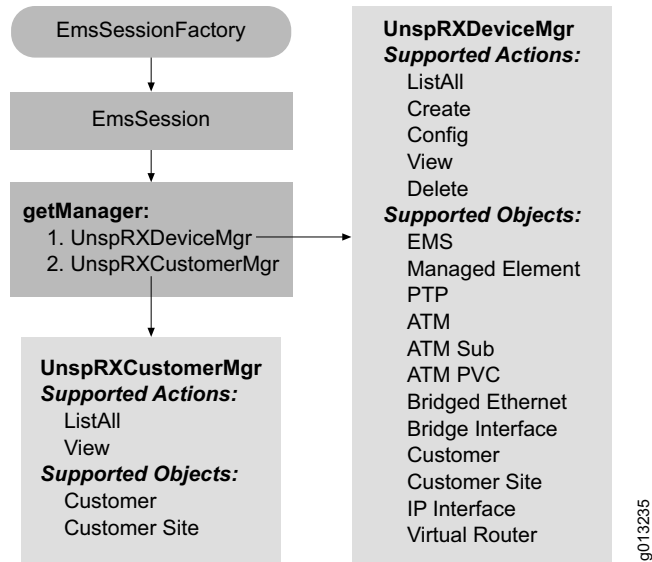


Figure 1-3 NMC-RX Provisioning Service objects

## 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.