NORTEL NETWORKS

Preside Multiservice Data Manager

DPN Provisioning API

Reference Guide

241-6001-204

Preside Multiservice Data Manager

DPN Provisioning API

Reference Guide

Publication: 241-6001-204
Document status: Standard
Document version: 15.1RSUP
Document date: August 2004

Printed in Canada

Publication history

August 2004

15.1RSUP Standard
Commercial availability

Contents

About this document **11**
Who should read this document and why 11
What you need to know 11
How this document is organized 12
What's new in this document 12
Text conventions 12
Related documents 14

Chapter 1
Introducing the DPN Provisioning API **15**
What is the DPN Provisioning API 15
Object classes 17
Data types 17
Sieves and event report types 17
Message types 17

Chapter 2
Object model **23**
About the object model 23
Containment hierarchy 23
Object classes 25
Attributes 27
Actions 28
Name bindings 29

Chapter 3
Using the DPN Provisioning API 31
Code conventions 31
Installing and configuring the DPN Provisioning API 31
Starting a session with the DPN Provisioning API 32
Command Filter parameters 33
The Register script file 34
Terminating DPN Provisioning API access 38
Commands 41
The REGISTER command 41
The DEREGISTER command 43
Actions 45
The GET command 53
The SET command 56
The CREATE command 57
The DELETE command 59
Propagation log files 60
Sample Propagation log file 61

Chapter 4
API model difference tool 67
About the API model difference tool 67
Model files 67
Command line syntax 68
Error messages 70

Appendix A
DPN Provisioning API object definitions 75

Appendix B
Compliance statement 77
API message types 78
API message lines 79
API data types 84
Sieve object support 84

Appendix C
Error messages **87**
Response to undesired events 87
Error messages 88

Index **97**

About this document

The following topics are discussed in this section:

- "Who should read this document and why" (page 11)
- "What you need to know" (page 11)
- "How this document is organized" (page 12)
- "What's new in this document" (page 12)
- "Related documents" (page 14)

Who should read this document and why

This document is for customers who use the DPN Provisioning Application Programming Interface (API) to write custom applications for the Preside Multiservice Data Manager (MDM) workstation.

What you need to know

To use the DPN Provisioning API, you need to know how to log onto the Preside Multiservice Data Manager (MDM) workstation. You must be familiar with the UNIX operating system, the UNIX C-shell, and a UNIX text editor such as vi.

Required reading consists of the 241-6001-200 *Preside MDM Application Programming Interface Primer* for an overview of the Application Programming Interface (API). The DPN Provisioning API follows the API Primer with the constraints listed in "Compliance statement" (page 77).

How this document is organized

241-6001-204 *Preside MDM DPN Provisioning API Reference Guide* contains the following sections:

- "Introducing the DPN Provisioning API" (page 15) describes the managed objects and API messages of the DPN Provisioning API.
- "Object model" (page 23) describes the containment hierarchy and the object classes of the DPN Provisioning API.
- "Using the DPN Provisioning API" (page 31) details how to use the DPN Provisioning API through a set of examples.
- "API model difference tool" (page 67) details how to compare two model files.
- "DPN Provisioning API object definitions" (page 75) is a reference to the file containing information on the DPN Provisioning API object model.
- "Compliance statement" (page 77) details how the DPN Provisioning API complies with the API Primer.
- "Error messages" (page 87) provides a list of error messages for the DPN Provisioning API.

What's new in this document

There are no changes in this document for this release.

Text conventions

This document uses the following text conventions:

- `nonproportional spaced plain type`

  Nonproportional spaced plain type represents system generated text or text that appears on your screen.

- **`nonproportional spaced bold type`**

  Nonproportional spaced bold type represents words that you should type or that you should select on the screen.

- *italics*

  Statements that appear in italics in a procedure explain the results of a particular step and appear immediately following the step.

  Words that appear in italics in text are for naming.

- `[optional_parameter]`

  Words in square brackets represent optional parameters. The command can be entered with or without the words in the square brackets.

- `<general_term>`

  Words in angle brackets represent variables which are to be replaced with specific values.

- UPPERCASE,lowercase

  In Preside Multiservice Data Manager (MDM), uppercase and lowercase letters that appear in UNIX commands and parameters must be matched exactly. The system matches upper and lowercase characters differently.

- |

  This symbol separates items from which you may select one; for example, ON|OFF indicates that you may specify ON or OFF. If you do not make a choice, a default of ON is assumed.

- ...

  Three dots in a command indicate that the parameter may be repeated more than once in succession.

The term absolute pathname refers to the full specification of a path starting from the root directory. Absolute pathnames always begin with the slash (/) symbol. A relative pathname takes the current directory as its starting point, and starts with any alphanumeric character (other than /).

Related documents

See the following documents for related information:

- 241-6001-012 *Preside MDM Configuration Management for DPN User Guide*
- 241-6001-200 *Preside MDM Application Programming Interface Primer*
- 241-6001-304 *Preside MDM Configuration Management for DPN Administration*
- 241-6001-310 *Preside MDM Server Reference Guide*

Chapter 1
Introducing the DPN Provisioning API

This section introduces the main functions of the DPN Provisioning API. This section contains the following information:

- "What is the DPN Provisioning API" (page 15)

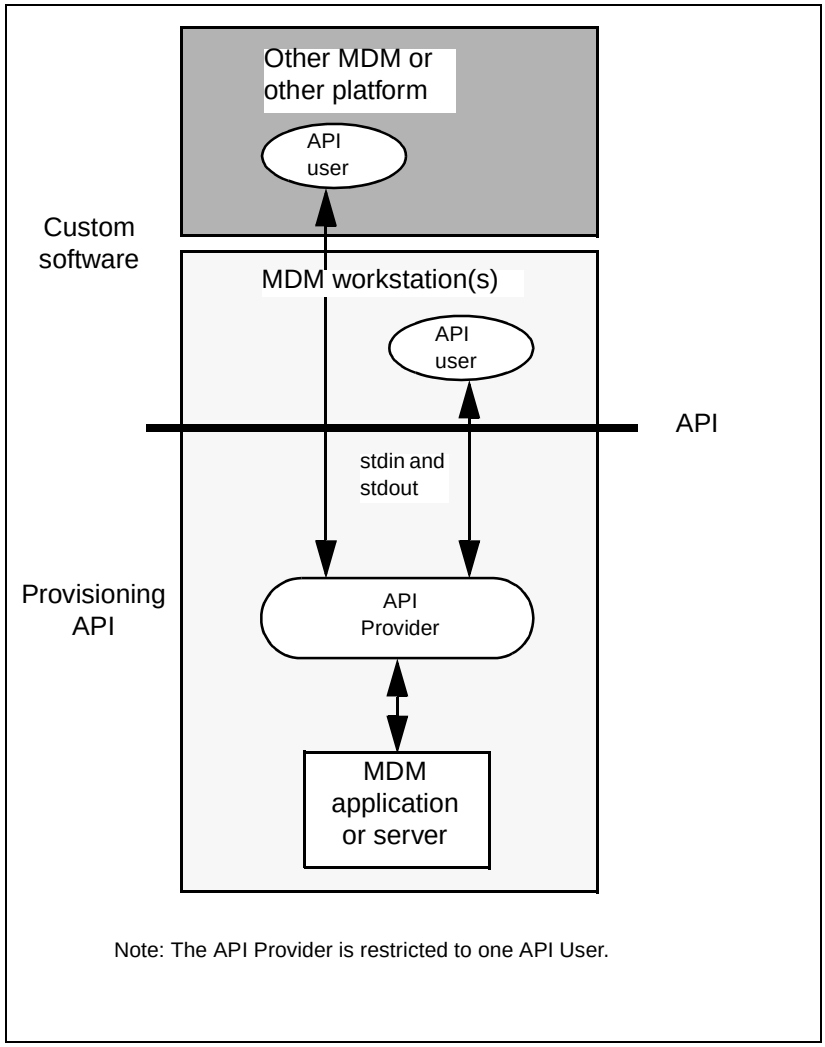
What is the DPN Provisioning API

The DPN Provisioning Application Programming Interface (API) is an ASCII interface that provides access to Preside Multiservice Data Manager (MDM) Configuration Component Provisioning information. Customer-written applications access Component Provisioning through the DPN Provisioning API. It allows you to create, view, and modify service data. The API responds with one or more response messages, error messages, or both.

The DPN Provisioning API reads service requests from standard input (stdin) and writes responses to standard output (stdout). Piping between the user and the DPN Provisioning API must be established when the DPN Provisioning API is initiated. This piping is external to the DPN Provisioning API. The communication between the customer application and the network is shown in the figure "MDM workstation custom programming environment" (page 16). Possible methods of establishing piping are explained in 241-6001-200 *Preside MDM Application Programming Interface Primer*.

The API Provider is the software that provides access to MDM data. The API User is the script, program, or human user that communicates with the API Provider over the interface. The API is defined in terms of the messages passed between the API User and the API Provider.

Figure 1
MDM workstation custom programming environment



In a session between an API Provider and an API User, the API User issues requests and receives responses and/or errors from those requests. In addition, the API Provider sends a version message during initialization and an end message during termination.

Object classes

An online file called /opt/MagellanNMS/lib/api/provapi.model is provided. It contains the object and attribute names as they appear in the DPN Provisioning API model. See “Object model” (page 23), for the format descriptions of the object classes.

Data types

The DPN Provisioning API supports the following data types:

- NI (Node ID)
- I (Integer)
- S (String)
- SS (Sequence of Strings)
- FS (Formatted String)
- H (Hexadecimal)
- B (Boolean)
- SI (Sequence of Integers)
- RI (Range of Integers)
- SB (Sequence of Booleans)

For detailed information on the data types, see the 241-6001-200 *Preside MDM Application Programming Interface Primer*.

Sieves and event report types

The DPN Provisioning API does not support sieves or event reports.

Message types

In a session between the API Provider and the API User, the API User issues requests and receives responses (and where appropriate, error messages) from those requests.

The API Provider may return an error message to indicate some failure in the processing of a request. For error messages with a severity of information or warning, the DPN Provisioning API completes processing, sends a successful response message, and then sends an end-of-response message. For error

messages with a severity of error, the DPN Provisioning API does not process the request, sends the error message(s), and then sends an end-of-response message. For error messages with a severity of FATAL_ERROR, the DPN Provisioning API terminates with an end message.

> ***Note:*** For the SYNTAX_ERROR error message, the DPN Provisioning API stops processing and purges the request.

The end-of-response message indicates that no more responses are forthcoming from the API Provider and that the request processing is complete.

Request message format

The general format of the request message is:

```
_cmd: <command keyword>
[_inv_id: <invoke ID>]
<request data line>
<request data line>
...
<request data line>
<blank line>
```

Response message format

The general format of the response message is:

```
[_inv_id: <invoke ID>]
<request data line>
<request data line>
...
<request data line>
<blank line>
```

Error message format

The format of an error message is:

```
[_inv_id: <invoke ID>]
_error: <API error code>
_attr: errorSeverity E <severity>
_attr: errorCode S <application error code>
_attr: errorApplicationId S <application ID>
_attr: errorFacility S <facility>
_attr: errorText FS <textual error message>
```

```
_attr: errorInformation SS <extra data value>
_attr: errorInformation SS <extra data value>
...
_attr: errorInformation SS <extra data value>
<blank line>
```

For a list of error codes and their description, see “Error messages” (page 87).

For details on the request, response, and error message formats, see the 241-6001-200 *Preside MDM Application Programming Interface Primer*.

For the DPN Provisioning API, the supported request messages are:

- REGISTER
  The REGISTER message establishes Network Control System (NCS) and/or UNIX access.
- DEREGISTER
  The DEREGISTER message terminates NCS and/or UNIX access.
- CREATE
  The CREATE message is supported for all objects subordinate to the uploaded object. The superior object ID and reference object ID CREATE message lines are not supported. The CREATE message cannot create a Packet Module (PM). To create a PM, refer to the upload action in “Using the DPN Provisioning API” (page 31).
- DELETE
  The DELETE message is supported for all subordinates of the object for which the upload action was issued. Only a scope of ALL is supported. Filtering is not supported.
- GET
  The GET message is supported on the uploaded object and all of its subordinates.
- SET
  The SET message is supported on the uploaded object and all of its subordinates (if the upload action request was in update mode). Scoping and filtering are not supported. Only the REP modification type is supported. Only a scope of BASE is supported. Filtering is not supported.
- ACTION

— upload
The upload action message is supported for a PM component or a lower level component.

— download
The download action message is supported for the component in the most recent upload action.

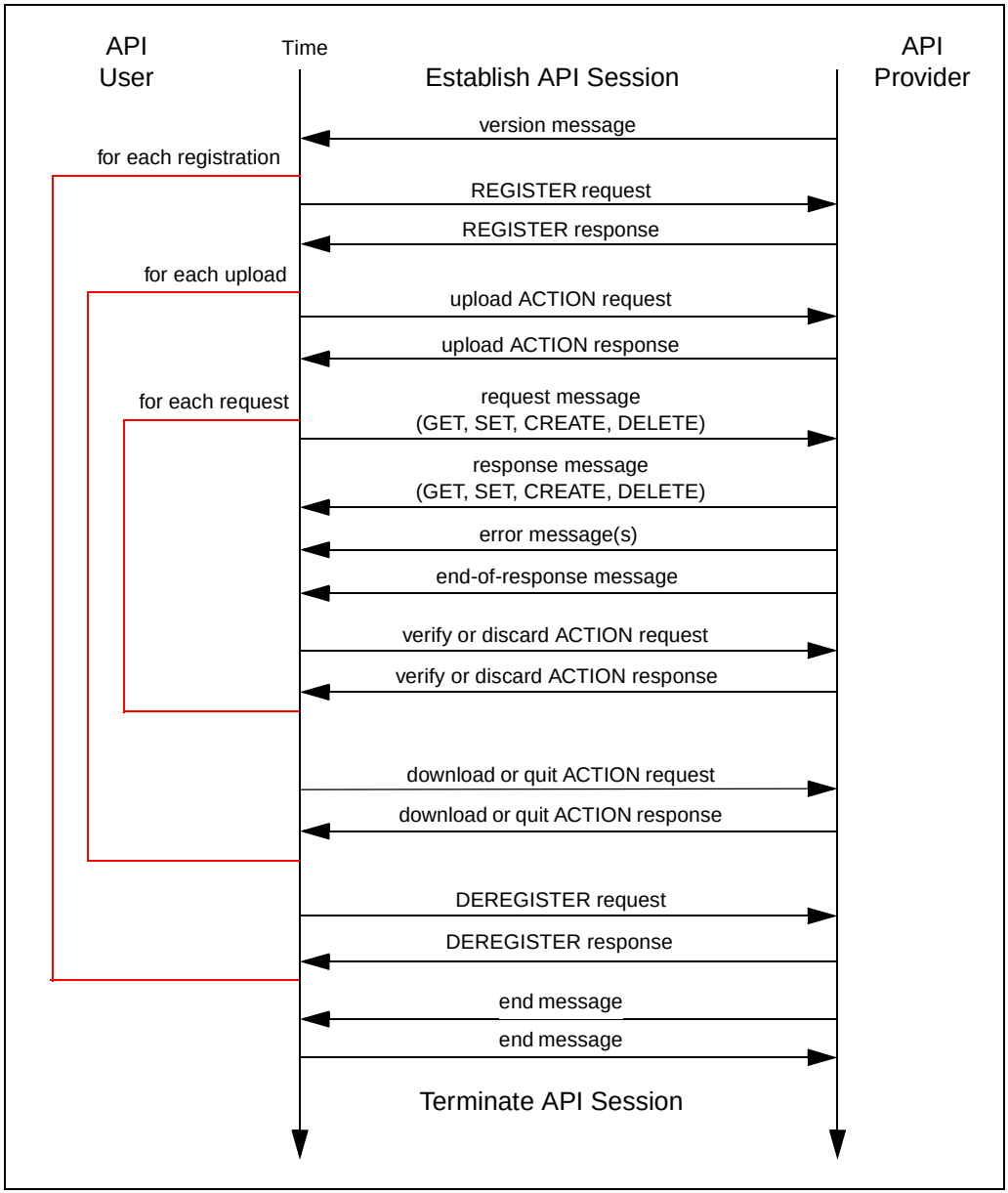
— verify
The verify action message is supported for the uploaded object and all of its subordinates.

— discard
The discard action message is supported for the uploaded object and all of its subordinates.

— quit
The quit action message is supported for the component in the most recent upload action.

The figure “Time line diagram” (page 21) shows an overview of the session events. It represents the normal sequence of messages between the API Provider and the API User. For more details on the message types, see the 241-6001-200 *Preside MDM Application Programming Interface Primer*.

Figure 2
Time line diagram



Chapter 2
Object model

This section describes the object model, and contains the following information:

- "About the object model" (page 23)
- "Containment hierarchy" (page 23)
- "Object classes" (page 25)

About the object model

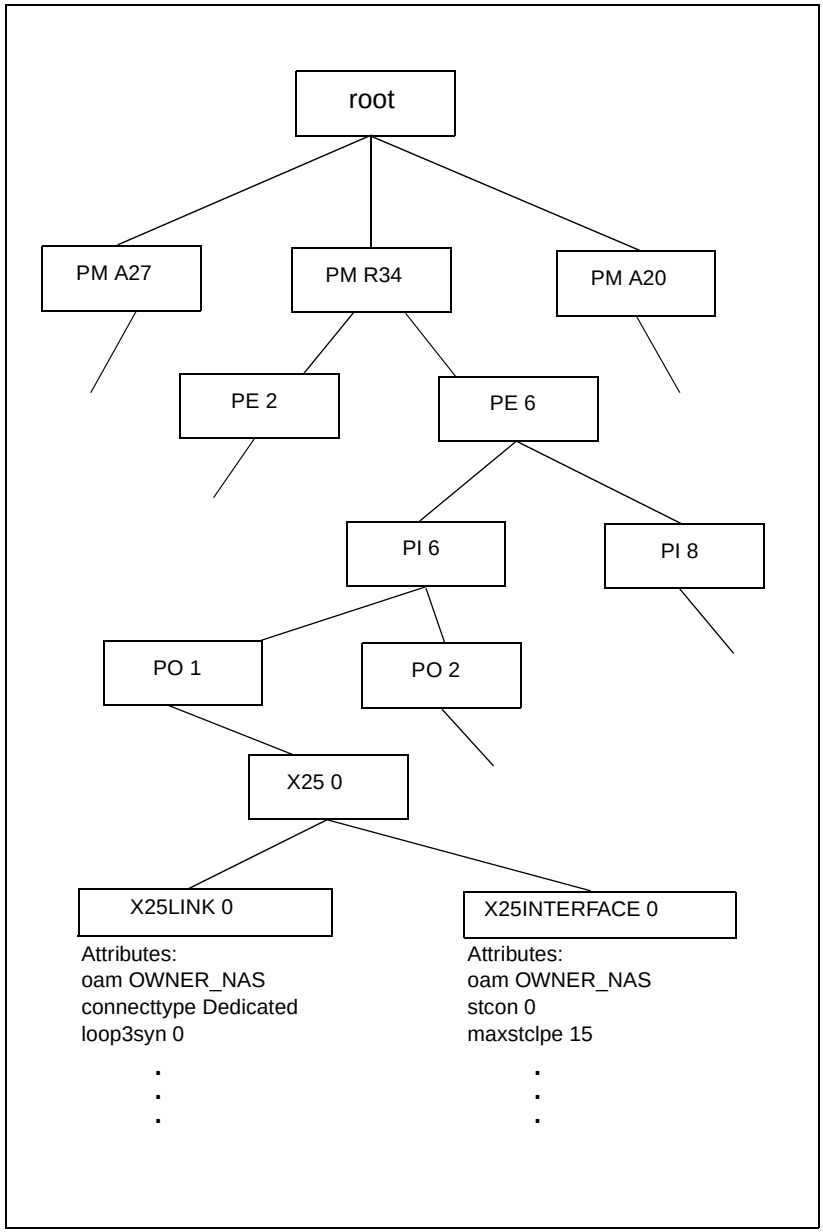
The DPN Provisioning API object model definition is not included in this document because it is too large and dynamic.

An online file containing information on the Provisioning object model called /opt/MagellanNMS/lib/api/provapi.model is provided. This file contains object and attribute names as they appear in the DPN Provisioning API model.

Containment hierarchy

The figure "Sample service data hierarchy" (page 24) shows sample service data components organized into a containment hierarchy, starting at the Packet Module (PM) and continuing to the service envelope object with its attributes (fields). The API User can use the DPN Provisioning API to view the service data for a PM (or lower-level components), edit the service data, and download the edited service data back to a PM. The commands that enable the API User to do this are explained in "Using the DPN Provisioning API" (page 31).

Figure 3
Sample service data hierarchy



An explanation of the components in the service data hierarchy shown in the figure “Sample service data hierarchy” (page 24) is detailed in the following paragraphs.

Service data component
Each service data component has a category name and a key value. For example, PM A27, the category name is Packet Module (PM), and the key value is A27.

Attributes
These are the attributes (fields) of the service data components to which they belong. For example, some service data fields of the X.25 LINK Envelope are linkproc, idleprobe, and framewindowsize.

Component name
This is the sequence of all the category names and key values of all components along the path from the top of the network hierarchy down to and including the component being named in the figure “Sample service data hierarchy” (page 24) (for example, the component name for PI is defined as PM R34 PE 6 PI 6).

Subordinates
These are components that are contained within other components one level higher in the hierarchy. For example, the X.25 LINK and X.25 INTERFACE envelopes are subordinates of the X.25 service component.

Object classes

The DPN Provisioning API object classes and their attributes are described in the /opt/MagellanNMS/lib/api/provapi.model file. The format of this file is described in the following sections.

Supported object classes are specified using the following template:

OBJECT CLASS: **<object class>**

MUST CONTAIN: **{<attributes, components, and allowed requests that objects of this class must have>}**

MAY CONTAIN: **{<attributes, components, and allowed requests that objects of this class may have>}**

* **UI prompt:** **"<UI prompt>"**

where:

object class
is a string name assigned to the defined object class. The value specified is used in the DPN Provisioning API in component IDs where a category name is required.

attributes, components, and allowed requests that objects of this class must have
is a list of the mandatory attributes and components for the objects of this class. Each attribute is indicated as GET if the attribute is readable and SET if the attribute can be modified.

attributes, components, and allowed requests that objects of this class may have
is a list of the optional attributes and components for the objects of this class. Each attribute is indicated as GET if the attribute is readable and SET if the attribute can be modified.

UI prompt
is the label of the field in Provisioning User Interface (UI).

Example

```
OBJECT CLASS: ITI
MUST CONTAIN: {
                ITILINK             CREATE DELETE GET
MAY CONTAIN : {
                ITIUSERPROFILE      CREATE DELETE GET
                ITIAUTODIRECT       CREATE DELETE GET
                ITINUIANDTRANSLATIONOPTIONS
                                    CREATE DELETE GET
                ITIVIDEOTEXAPPLICATION
                                    CREATE DELETE GET
                ITILDNA             CREATE DELETE GET

* UI prompt     : "ITI"
```

Attributes

Supported attributes are specified using the following template:

ATTRIBUTE TYPE: **<attribute name>**

WITH ATTRIBUTE SYNTAX: **<attribute value type>**

LEGAL VALUES: **<legal values for the defined attribute>**

DEFAULT: **<default values for the defined attribute>**

<MULTI-VALUE>

* **UI prompt:** **"<UI prompt>"**

where:

attribute name
is the character string name assigned to this attribute. The value specified is used in the DPN Provisioning API in the _attr:, _attr_id:, and _mod: lines.

attribute value type
specifies the data type for values of this attribute. See “Data types” (page 17) for the list of valid data types.

legal values for the defined attribute
if present, indicates the legal values for the attribute. If not present, any values allowed by the syntax are valid.

default values for the defined attribute
if present, indicates the default value for the attribute. If not present, there is no default value.

UI prompt
is the label of the field in Provisioning User Interface (UI).

MULTI-VALUE
if present, indicates that the attribute can have more than one value. If not present, the attribute can have only one value.

Example

```
ATTRITBUTE TYPE:          inputspeed
WITH ATTRIBUTE SYNTAX:    S, I
LEGAL VALUES :            {75,110,'Rauto'150,200,300,
                          600,1200,2400,4800,9600,
                          'Auto','Ext',14400,19200,
                          28800,38400}
* UI prompt               :  "Input speed"
```

Actions

Supported actions are specified using the following template:

ACTION TYPE: **<action type>**

WITH ATTRIBUTES: **{<attributes that actions of this type may have>}**

RESPONSE ATTRIBUTES: **{<attributes that responses to this type of action may have>}**

where:

action type
is string name assigned to the defined action. The value specified is used in the DPN Provisioning API in the _action_type: line.

attributes that actions of this type may have
is a list of the attributes that may be sent as part of the action request information for these types of actions.

attributes that responses to this type of action may have
is a list of the attributes that may be sent as part of the action response information for these types of actions.

Example

```
ACTION TYPE: upload
WITH ATTRIBUTES:      {
                          accessmode          ,
                          location            ,
                          namsid              ,
                          sdversion           ,
                          view
                      }
RESPONSE ATTRIBUTES:  {
                         mcfname              ,
                         sdversion
                      }
ACTION TYPE:download
```

Name bindings

A name binding is specified by indicating the superior and subordinate object classes and the attribute used for naming the subordinate objects.

Supported name bindings are specified using the following template:

OBJECT CLASS: **<subordinate object class>**

IS NAMED BY: **<superior object class>**

WITH ATTRIBUTE: **<attribute used for naming>**

* **Mandatory child:** **YES or NO**

* **Allowed instants:** **integer value**

where:

superior object class
is the superior object class.

subordinate object class
is the subordinate object class.

Note: When the subordinate object class is subordinate to the root of the tree, the IS NAMED BY line is absent.

attribute used for naming
is the attribute that names the subordinate object.

* **Mandatory child**
If YES, the created superior object class must have an instance of this OBJECT CLASS. If NO, no instance of this OBJECT CLASS is required.

* **Allowed instants**
is the maximum number of instances of this OBJECT CLASS for this particular relationship.

Example

```
OBJECT CLASS:          ITILINK
IS NAMED BY:           ITI
WITH ATTRIBUTES:       ITILINK
* Mandatory child :    YES
* Allowed instants:    1
```

Chapter 3
Using the DPN Provisioning API

This section explains how to install and use the DPN Provisioning API. This section contains the following information:

- "Code conventions" (page 31)
- "Installing and configuring the DPN Provisioning API" (page 31)
- "Starting a session with the DPN Provisioning API" (page 32)
- "Terminating DPN Provisioning API access" (page 38)
- "Commands" (page 41)
- "Propagation log files" (page 60)

Code conventions

There are two code conventions used in this document.

- \
  A backslash (\ ) indicates that the line of code is continued on the next line space.
- * or #
  A message line that starts with an asterisk (*) or an octothorpe (#) is treated as a comment.

Installing and configuring the DPN Provisioning API

The DPN Provisioning API software requires no additional installation or configuration procedures.

Starting a session with the DPN Provisioning API

When using the DPN Provisioning API, you actually work with the process called the API Provider (provapi). To start a session with the DPN Provisioning API, you invoke the API Provider by entering the following:

```
/opt/MagellanNMS/bin/provapi [-h]
```

or

```
/opt/MagellanNMS/bin/provapi [-width <width>]
   [-fasealog <file>]\
   [-nouserid] [-trace] [-echo]\
   [-server <server>]
```

where:

- `-h`
  displays help on the format of the provapi command.
- `-width <width>`
  is the maximum length of the lines generated by the API Provider. The default (and minimum) length is 72 characters; the maximum value is 100000. When specifying a line length greater than 80, be careful of any limitations that might exist in the communication path between the API Provider and the API User application.
- `-fasealog <file>`
  redirects stdout and stderr output from fa (Field Access) and sea (Envelope Access) to the specified <file>. The default file /dev/null discards the output. This option is useful for diagnostic purposes.
- `-nouserid`
  when specified, makes optional the _user_id: and _password: lines in the REGISTER command.

CAUTION
To ensure workstation security, do not use the -nouserid option if the API Provider is to be used as a network server.

- `-trace`
  writes debugging information (Protocol Data Units (PDU) as they are sent and received, and tokens as they are parsed) to stderr.
- `-echo`
  echoes the input.
- `-server <server>`
  replaces the default server /opt/MagellanNMS/bin/fa with the specified executable file <server> for debugging purposes.

If initialization fails, then one or more error messages are output, followed by an end message, and the termination of the API Provider.

If initialization succeeds, the response is similar to the following:

```
_version: x.y Provisioning API Copyright Nortel Networks
1994
```

where x and y are the major and minor version numbers.

Note: The response is on one line on the UNIX screen.

Command Filter parameters

For more information on the Provisioning Command Filter API, see 241-6001-209 *Preside MDM Provisioning Command Filter API for DPN*.

One or more of the following options are used to invoke the Provisioning Command Filter API with the Provisioning API.

```
/opt/MagellanNMS/bin/provapi <provisioning API
   parameters>
   [-filter_appl "<UNIX command>" |
   -filter_serv <node/IP address> <port>]
   [-filter_trace] [-filter_width <width>]
```

where:

- `<provisioning API parameters>`
  One or more of the options described in the previous section: “Starting a session with the DPN Provisioning API” (page 32).

- `-filter_appl`
  An option used to specify a UNIX command that is to be the customer application for command filtering. The Provisioning Command Filter API Provider sets up stdin and stdout for the customer application.
- `<UNIX command>`
  The UNIX Bourne shell command for invoking the customer application that is to filter the provisioning commands.
- `-filter_serv`
  An option used to specify a node name or IP address and port of a server that is to be the customer application for command filtering. The Provisioning Command Filter API Provider makes a stream socket call to the server.
- `<node/IP address>`
  Either the mnemonic or the IP address for a workstation running a Provisioning Command Filter server.
- `<port>`
  The port of the Provisioning Command Filter server.
- `-filter_trace`
  Turns on Provisioning Command Filter API tracing. Trace output is to stderr.
- `-filter_width`
  An option used to specify the maximum length of lines output by the Provisioning Command Filter API Provider. The default is 72.
- `<width>`
  A number between 72 and 100000.

The Register script file

The Register script file is the preferred method of initiating the API Provider and running an API command file, because the passwords are not part of the command file. The Register script establishes the necessary authorization for provisioning activities. If the connection fails, a message is displayed on the screen. If service data on a Packet Module (PM) is accessed, then the destination mnemonic, capability ID, and password must be specified. If only

an NMS disk is accessed for service data, these are not necessary. To use the DPN Provisioning API with this method, create an API command file and enter the following:

```
/opt/MagellanNMS/bin/register [-v] [-h]
```

where:

- `-h`
  is Help.
- `-v`
  is the verbose mode. It directs responses for each command to stdout. If this option is not specified, the command displays what was uploaded, downloaded, and any error messages.

The following prompts are displayed; (enter Return after each line of information):

```
API command file = (enter the API command filename)
NCS access? [y/n] (enter y or n)
```

If Network Control System (NCS) access is requested, the API User is queried for the following (enter Return after each line of information):

```
Destination mnemonic = (enter the destination mnemonic)
Capability id = (enter the capability ID)
Capability password = (enter the password)
```

The above information establishes the necessary authorization for DPN Provisioning activities.

Example 1 and Example 2 both use the api.cmd.file command file as shown below.

```
_cmd: ACTION
_inv_id: 2
_obj_class: pm
_obj_id: compId NI PM R34
_action_type: upload
_attr: accessmode S UPDATE
_attr: location SS DISK
_attr: view SS bundle 4998
_attr: namsid I 4034
```

```
<blank line>
_cmd: SET
_inv_id: 3
_obj_class: itilink
_obj_id: compId NI PE 5 PI 5 PO 1 ITI 0 ITILINK 0
_mod: REP inputspeed I 4800
_mod: REP outputspeed I 4800
<blank line>
_cmd: ACTION
_inv_id: 4
_obj_class: pm
_obj_id: compId NI PM R34
_action_type: download
_attr: location SS DISK
_attr: view SS key CHRISW
_attr: mode S INCREMENTAL
_attr: check S INCREMENTAL
_attr: namsid I 4034
<blank line>
*_end:
*
```

Example 1

To use the Register script file without the *-v* option, enter the following:

```
/opt/MagellanNMS/bin/register
```

Respond to the prompts that appear as shown in the following example:

```
API command file = api.cmd.file
UNIX userid = provapi
UNIX password = <invisible_password>
NCS access? [y/n] y
Destination mnemonic = LAB0
Capability id = IWSA
Capability password = <invisible_password>
```

If the request message is successful, the response message is:

```
NOTE: MCF MC.4998.4034.0 uploaded.
<blank line>
WARNING: CONMAN_SDA 0: A "Data Collection" "ICON_
```

```
Mnemonic" is required under the "Console_Manager"
component.
<blank line>
NOTE: MCF MC.CHRISW01.4034.0 downloaded.
```

Example 2
To use the Register script file with the -v option, enter the following:

```
/opt/MagellanNMS/bin/register -v
```

Respond to the prompts that appear as shown in the following example:

```
API command file = api.cmd.file
NCS access? [y/n] n
```

If the request message is successful, the following response message is received followed by the responses to the commands in the api.cmd.file command file:

```
_version: 8.0 Provisioning API
Copyright Nortel Networks 1994
<blank line>
_inv_id: 1
<blank line>
_inv_id: 1
_end_resp: REGISTER
_time: 1994 02 11 14 31 36
<blank line>
_inv_id: 2
_obj_class: pm
_obj_id: compId NI PM R34
_attr: mcfname S MC.4998.4034.0
_attr: sdversion I 7
<blank line>
_inv_id: 2
_end_resp: ACTION
_time: 1994 02 11 14 32 08
<blank line>
_inv_id: 3
_obj_class: itilink
_obj_id: compId NI PE 5 PI 5 PO 1 ITI 0 ITILINK 0
<blank line>
```

```
_inv_id: 3
_end_resp: SET
_time: 1994 02 11 14 32 17
<blank line>
_inv_id: 4
_error: APPLICATION_ERROR
_attr: errorSeverity E Warning
_attr: errorCode S "ES_7A12"
_attr: errorApplicationId S "PRO_FA"
_attr: errorFacility S "SEMANTIC"
_attr: errorText FS "WARNING: CONMAN_SDA 0: A \"Data\
Collection\" \"ICON_Mnemonic\" is required under the\
\"Console_Manager\" component."
_attr: errorInformation SS CONMAN_SDA 0
<blank line>
_inv_id: 4
_obj_class: pm
_obj_id: compId NI PM R34
_attr: mcfname S MC.CHRISWO2.4034.0
<blank line>
_inv_id: 4
_end_resp: ACTION
_time: 1994 02 11 14 32 59
<blank line>
_end:
```

Terminating DPN Provisioning API access

To terminate the DPN Provisioning API access, use either an end-of-file (control-d) or issue the following message:

```
_end:
<blank line>
```

The response message is:

```
_end: USER_REQUEST
<blank line>
Exiting API Provider with 0 status.
```

The API User may also terminate the session by closing the stdin, that is, the API Provider receives an End of File (EOF). For example, entering control-D from an interactive session or the end of a file redirected to stdin (for instance, /opt/MagellanNMS/bin/provapi < filename>).

Note: EOF discards any partial service request.

The API Provider may need to terminate the session on its own. To do so, it outputs an end record (usually preceded by one or more error records). The general form of the end_record is:

```
_end: <termination code> <termination code data>
<blank line>
```

The <termination code> is one of the following:

- USER_REQUEST
  The customer application has requested termination by executing the _end: message request. This is the normal termination response.
- FATAL_ERROR
  One of the unrecoverable errors listed in the table “Fatal errors” (page 39) occurred.

Table 1
Fatal errors

Error	Description
PDU failure	The DPN Provisioning API was unable to send/receive a Protocol Data Units (PDU) to/from Field Access (PDUs are the specially formatted Interprocess Communications (IPC) messages exchanged by Provisioning applications). This is either a software problem or there is insufficient space in the /tmp file.
Lost connection	There is a software or other problem in Field Access, Envelope Access, File Access, or IPC.
Lost session	Field Access had to abort the session for some reason, or there is a software, or other problem in the Field Access, Envelope Access, or File Access.
(Sheet 1 of 2)	

Table 1 (Continued)
Fatal errors

Error	Description
Out of memory	An attempt to allocate some memory has failed.
Startup failure	Make sure that the directories /opt/MagellanNMS/bin/fa and /opt/MagellanNMS/bin/sea exist. Make sure that mnsd is running and that pfas is running and accessible through mnsd. Check the fa and sea log file (assuming the -fasealog option was specified on the provapi command line) for Field Access and Envelope Access log messages that might indicate the nature of the problem (for example, activation file errors). ***Note:*** This error also contains the API version number. This is used for errors occurring before the version message can be output. The version number is included on these errors since the version message telling the client the version of the API has not been output. The client cannot process the error record without knowing the version and therefore, the format.

(Sheet 2 of 2)

Example
In this example, the DPN Provisioning API was unable to start up Field Access. For /opt/MagellanNMS/bin/provapi -v, the error message is:

```
_error: INIT_ERROR 8.0
_attr: errorSeverity E Fatal_Error
_attr: errorCode S "A0012"
_attr: errorApplication "PRO_API"
_attr: errorFacility S ""
_attr: errorText FS "FATAL ERROR:
      Invalid option: -v"
_attr: errorInformation SS -v
<blank line>
_end: FATAL_ERROR
```

Commands

Each command has a request message and a response, and/or error message(s). See 241-6001-200 *Preside MDM Application Programming Interface Primer*, for more details on commands. The DPN Provisioning API supports the following commands:

- REGISTER
- DEREGISTER
- ACTION
  - upload
  - download
  - verify
  - discard
  - quit
- GET
- SET
- CREATE
- DELETE

Note: The GET, SET, CREATE, and DELETE commands can only be issued after the upload ACTION command has been issued, and before the corresponding quit or download ACTION command has been issued.

The REGISTER command

The REGISTER request must be the first request message. It identifies an API User and determines the privileges for the use of the Provisioning API. A REGISTER request results in a single, REGISTER response message, followed by an end-of-response message.

A REGISTER command establishes NCS access and authorization when the service data is on a Packet Module (PM). The API User must issue a DEREGISTER request message before issuing another REGISTER request

message (see “The DEREGISTER command” (page 43)). The DEREGISTER request message is not required before terminating an API session.

If service data on a Packet Module (PM) is accessed, the destinationmnemonic, capabilityid attributes, and password must be specified. If only an NMS disk is accessed for service data, this validation is not necessary.

> ***Note:*** Registering and deregistering have no effect on the presence of managed objects in the API Provider’s object model, but they affect the API User’s ability to access objects.

Using the REGISTER command

The format of the REGISTER request message is:

```
_cmd: REGISTER
[_inv_id: <invoke ID>]
[_user_id: <UNIX user ID>]
[_password: <UNIX user ID password>]
[_attr: destinationmnemonic S <destination mnemonic>]
[_attr: capabilityid S <capability ID]
[_attr: password S <password>]
<blank line>
```

The format of the REGISTER response message is:

```
[_inv_id: <invoke ID>]
<blank line>
[_inv_id: <invoke ID>]
_end_resp: REGISTER
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

When you invoke the API Provider from a UNIX window on a Preside Multiservice Data Manager (MDM) workstation or from a program running on an MDM workstation, do the following:

- specify the -nouserid option in the provapi command
- omit the _user_id: and _password: lines from the REGISTER commands

Note: If the invocation is from the root userid, you can omit the -nouserid option and specify the _user_id: and _password: lines in the first REGISTER command.

When running the API Provider as a network server (for example, from the inet daemon), you need to invoke it as root without the -nouserid option. Specify the _user_id: and _password: lines in the first REGISTER command in the API session and omit them from subsequent REGISTER commands. The API Provider switches from root to the specified UNIX userid for the rest of the API session.

Example

The following REGISTER request message declares the UNIX User ID provapi and the NCS capability ID *IWSA*. The user and capability passwords are also specified at this time.

```
_cmd: REGISTER
_user_id: user1
_password: secretwd
_attr: destinationmnemonic S LAB0
_attr: capabilityid S IWSA
_attr: password S QWERTY
<blank line>
```

If the REGISTER request message is successful, the response message is:

```
<blank line>
```

followed by an end-of-response message.

```
_end_resp: REGISTER
_time: 1993 08 19 17 14 18
<blank line>
```

The DEREGISTER command

The DEREGISTER command terminates the registration of an API User. A DEREGISTER request message results in a single DEREGISTER response message followed by an end-of-response message.

The DEREGISTER request message discards the authorization established by the last REGISTER request message. It is used after a download action (see “The download action” (page 48)) or a quit action (see “The quit action” (page 52)) in preparation for a new REGISTER request message (see “The REGISTER command” (page 41)).

Note: Registering and deregistering have no effect on the presence of managed objects in the API Provider’s object model, but they affect the API User’s ability to access objects.

Using the DEREGISTER command

The format of the DEREGISTER request message is:

```
_cmd: DEREGISTER
[_inv_id: <invoke ID>]
<blank line>
```

The format of the DEREGISTER response message is:

```
[_inv_id: <invoke ID>]
<blank line>
[_inv_id: <invoke ID>]
_end_resp: DEREGISTER
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

Example

To deregister, issue the following request message:

```
_cmd: DEREGISTER
<blank line>
```

If the DEREGISTER request is successful, the response message is:

```
<blank line>
_end_resp: DEREGISTER
_time: 1993 08 19 17 14 18
<blank line>
```

Actions

The ACTION request identifies one object to perform a defined action. An ACTION request results in one successful ACTION response message with zero or more Informational or Warning error messages, or error message(s) followed by an end-of-response message.

In the API Primer, an ACTION request message is directed at the object and is performed by that particular object. In the DPN Provisioning API, this request performs one of the following functions:

- upload
- download
- verify
- discard
- quit

The upload action

The upload action establishes a session to the Packet Module (PM) or NMS disk and specifies access to a specific view of service data for a specific PM or one of its subcomponents. This action request is required before service data can be examined or modified.

If another service data view is required, the quit (see "The quit action" (page 52)) or download (see "The download action" (page 48)) action request message must be issued. The upload action request message is then reissued with the new parameters.

There are two formats for the upload action request message: one for creating new service data, and one for existing service data.

The format of the upload action request message for creating new service data is:

```
_cmd: ACTION
[_inv_id: <invoke ID>]
[_obj_class: <object class>]
_obj_id: compId NI <distinguished name>
_action_type: upload
```

```
_attr: accessmode S UPDATE
_attr: sdversion I <version>
<blank line>
```

The format of the upload action request message for existing service data is:

```
_cmd: ACTION
[_inv_id: <invoke ID>]
[_obj_class: <object class>]
_obj_id: compId NI <distinguished name>
_action_type: upload
[_attr: accessmode S READ | UPDATE]
[_attr: location SS PM <PM> | DISK]
_attr: view SS bundle <bundle> | committed | active | \
key <key> | datekey <date key>
[_attr: namsid I <NAMS ID>]
<blank line>
```

The format of the upload action response message is:

```
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI PM <distinguished name>
[_attr: mcfname S <MCF name>]
[_attr: activationdate S <YYMMDD>]
_attr: sdversion I <SD version>
<blank line>
[_inv_id: <invoke ID>]
_end_resp: ACTION
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

Note 1: The default value for accessmode attribute is READ.

Note 2: The NAMS ID attribute must not be specified if the view attribute is COMMITTED or ACTIVE.

Note 3: The NAMS ID attribute must be specified if the location attribute is DISK.

Note 4: The COMMITTED or ACTIVE view attribute cannot be specified if the location attribute is DISK.

Note 5: The default location attribute is PM, specified in the _obj_id: line.

Note 6: Names of objects are rooted at the object that gets uploaded. This applies to commands issued after the upload ACTION and before the corresponding quit or download ACTION.

Example
This example shows how to view the service data for PM A27. The sample request message is:

```
_cmd: ACTION
_obj_class: pm
_obj_id: compId NI PM A27
_action_type: upload
_attr: accessmode S UPDATE
_attr: view SS key 91JULY
<blank line>
```

If the upload action request message is successful, the response message is:

```
_obj_class: pm
_obj_id: compId NI PM A27
_attr: mcfname S MC.91JULY03.7027.0
_attr: activationdate S 951103
_attr: sdversion I 7
<blank line>
_end_resp: ACTION
_time: 1993 08 19 17 14 18
<blank line>
```

This response message specifies the name of the Master Configuration File (MCF), the activation date, and the service data version.

Example
This example shows how to create service data using the upload command. The sample request message is:

```
_cmd: ACTION
_obj_id: compId NI PM R34
_action_type: upload
```

```
_attr: accessmode S UPDATE
_attr: sdversion I 8
<blank line>
```

If the upload action request message is successful, the response message is:

```
_obj_class: pm
_obj_id: compId NI PM R34
_attr: sdversion I 8
<blank line>
_end_resp: ACTION
_time: 1995 07 11 16 42 15
<blank line>
```

You can now provision the PM using CREATE and SET requests. Provisioning creates a new MCF, which you can download.

The name of the resulting MCF is determined at download time and is based on the download ACTION request attributes.

The download action

The download action downloads service data to a PM or to the NMS disk and then terminates the session that was opened using the upload action request message (see “The upload action” (page 45)).

The format of the download action request message is:

```
_cmd: ACTION
[_inv_id: <invoke ID>]
[_obj_class: <object class>]
_obj_id: compId NI PM <PM>
_action_type: download
[_attr: location SS PM <PM> | DISK]
_attr: view SS bundle <bundle> | key <key> | datekey\
<date key>
[_attr: activatable S YES | NO]
[_attr: activationdate S <YYMMDD>]
[_attr: mode S COMPLETE | INCREMENTAL]
[_attr: check S COMPLETE | INCREMENTAL | MINIMAL]
[_attr: namsid I <NAMS ID>]
<blank line>
```

The format of the download action response message is:

```
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI PM <PM>
_attr: mcfname S <MCF name>
<blank line>
[_inv_id: <invoke ID>]
_end_resp: ACTION
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

Note 1: The default value for the mode attribute is INCREMENTAL.

Note 2: The value of the mode attribute cannot be INCREMENTAL if there were no service data changes since the most recent upload action.

Note 3: The default value for the check attribute is INCREMENTAL.

Note 4: The default value for the view attribute is the value from the most recent upload action. This value may be unsuitable for a download action (for example, ACTIVE, COMMITTED, or bundle view attribute values).

Note 5: The default value for the NAMS ID attribute is the value from the most recent upload action. If that upload action created a whole new MCF rather than making changes to the existing service data, then the NAMS ID attribute must be specified on the download action.

Note 6: The default value for the location attribute is the value from the most recent upload action.

Note 7: The default value for the activatable attribute is YES.

Example

To download service data for PM R34, issue the following request message:

```
_cmd: ACTION
_obj_class: pm
_obj_id: compId NI PM R34
_action_type: download
```

```
_attr: view SS KEY 91AUG
_attr: mode S COMPLETE
<blank line>
```

If the download action request message is successful, the response message is:

```
_obj_class: pm
_obj_id: compId NI PM R34
_attr: mcfname S MC.91AUG02.4034.0
<blank line>
_end_resp: ACTION
_time: 1993 09 05 17 14 18
<blank line>
```

The response message gives the MCF file name, the date, and time the download action occurred.

The verify action

The verify action performs semantic checks on components (and their subcomponents) that were modified. The verify action is performed separately from a download action. Before the semantic check is executed, the designated component is fully expanded. This can result in significant overhead if the component is at a high level (for example, a PM) and has not already been expanded.

The format of the verify action request message is:

```
_cmd: ACTION
[_inv_id: <invoke ID>]
[_obj_class: <object class>]
_obj_id: compId NI <distinguished name>
_action_type: verify
<blank line>
```

The format of the verify action response message is:

```
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI <distinguished name>
<blank line>
[_inv_id: <invoke ID>]
```

```
_end_resp: ACTION
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

To verify service data from PE 7, issue the following request message:

```
_cmd: ACTION
_obj_class: pe
_obj_id: compId NI PM R34 PE 7
_action_type: verify
<blank line>
```

If the verify message is successful, the response message is:

```
_obj_class: pe
_obj_id: compId NI PM R34 PE 7
<blank line>
_end_resp: ACTION
_time: 1993 08 19 17 14 18
<blank line>
```

The discard action

The discard action abandons any changes made to a component (and any subcomponents) since the last upload action request.

The format of the discard action request message is:

```
_cmd: ACTION
[_inv_id: <invoke ID]
[_obj_class: <object class>]
_obj_id: compId NI <distinguished name>
_action_type: discard
<blank line>
```

The format of the discard action response message is:

```
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI <distinguished name>
<blank line>
[_inv_id: <invoke ID>]
_end_resp: ACTION
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

To discard changes to service data for PE 7, issue the following request message:

```
_cmd: ACTION
_obj_class: pe
_obj_id: compId NI PM R34 PE 7
_action_type: discard
<blank line>
```

If the discard action request message is successful, the response message is:

```
_obj_class: pe
_obj_id: compId NI PM R34 PE 7
<blank line>
_end_resp: ACTION
_time: 1993 08 19 17 14 18
<blank line>
```

The quit action

The quit action terminates access to a specific service data view. Service data changes since the most recent upload action request are lost.

The format of the quit action request message is:

```
_cmd: ACTION
[_inv_id: <invoke ID>]
[_obj_class: <object class>]
_obj_id: compId NI PM <PM>
_action_type: quit
<blank line>
```

The format of the quit action response message is:

```
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI PM <PM>
<blank line>
[_inv_id: <invoke ID>]
_end_resp: ACTION
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

To close the service data view on PM A27, issue the following request message:

```
_cmd: ACTION
_obj_class: PM
_obj_id: compId NI PM A27
_action_type: quit
<blank line>
```

If the quit action request is successful, the response message is:

```
_obj_class: PM
_obj_id: compId NI PM A27
<blank line>
_end_resp: ACTION
_time: 1993 08 19 17 14 18
<blank line>
```

The GET command

The GET command retrieves values of attributes from managed objects. A GET request message results in one of the following:

- successful response message for each selected object and zero or more Informational or Warning severity error messages
- error message(s)

followed by an end-of-response message.

The GET request message retrieves service data. The amount of service data returned may be tailored using the _scope: and _attr: lines. If the distinguished name is blank in the _obj_id: line, the base object is the upload component. The _attr_id: line identifies the attribute(s) to be returned. If ALL is specified as an attribute, all attributes are returned. If no _attr_id: line is entered, only the names of components are returned. More than one _attr_id: line may be entered.

The options for the _scope: line are:

- BASE
  returns the attributes for the component named by the _obj_id: line. This is the default value if the _scope: line is not used.

- NEXT
  returns the names of the objects at the next lower level and their attributes. A separate response message is returned for each object.
- ALL
  returns the names of the objects for the entire subhierarchy and their attributes. A separate response message is returned for each object.

  Note: Filtering is not supported.

Using the GET command

The format of the GET request message is:

```
_cmd: GET
[_inv_id: <invoke ID>]
[_obj_class: <object class>]
_obj_id: compId NI <distinguished name>
[_scope: BASE | NEXT | ALL]
[_attr_id: <attribute name> | ALL]
<blank line>
```

The format of the GET response message is:

```
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI <distinguished name>
_attr: <attribute name> <attribute type> <attribute
value>
_attr: <attribute name> <attribute type> <attribute
value>
...
_attr: <attribute name> <attribute type> <attribute
value>
<blank line>
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI <distinguished name>
_attr: <attribute name> <attribute type> <attribute
value>
_attr: <attribute name> <attribute type> <attribute
value>
...
_attr: <attribute name> <attribute type> <attribute
```

```
value>
<blank line>
...
...
...
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI <distinguished name>
_attr: <attribute name> <attribute type> <attribute
value>
_attr: <attribute name> <attribute type> <attribute
value>
...
_attr: <attribute name> <attribute type> <attribute
value>
<blank line>
[_inv_id: <invoke ID>]
_end_resp: GET
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

To obtain all the service data for PE 5 PI 5 PO 1 ITI 0 and itilink 0, issue the following request:

```
_cmd: GET
_obj_id: compId NI pe 5 pi 5 po 1 iti 0 itilink 0
_attr_id: ALL
<blank line>
```

If the GET request message is successful, the response message is:

```
_obj_class: itilink
_obj_id: compId NI PE 5 PI 5 PO 1 ITI 0 ITILINK 0
_attr: parity S “Auto”
_attr: characterlength S “7 bits”
_attr: inputspeed I 1200
_attr: outputspeed I 1200
_attr: modem I 113
_attr: stopbitclass S “1 over 110 bpc”
_attr: portmode S “DTE”
_attr: busyoutallowed I 0
_attr: callretries S “Unlimited”
```

```
_attr: forwardeverycharacter I 0
_attr: idletimerinterval S “50 msec”
_attr: cts I 0
_attr: rts I 0
_attr: stimer S “Infinity”
_attr: rtimer S “Infinity”
_attr: maximumassemblytimer S “Infinity”
_attr: callsetuptimer I 1
<blank line>
_end_resp: GET
_time: 1993 08 19 17 14 18
<blank line>
```

The SET command

The SET command modifies values of attributes of managed objects. A SET request message results in either one successful SET response message and zero or more Informational or Warning error messages, or error message(s) followed by an end-of-response message.

The SET command changes the values of the specified attributes of the indicated service data component. If the distinguished name of the _obj_id: line is blank, the base object is the upload component.

Note: Only BASE scope is supported. Filtering is not supported.

Using the SET command

The format of the SET request message is:

```
_cmd: SET
[_inv_id: <invoke ID>]
[_obj_class: <object class>]
_obj_id: compId NI <distinguished name>
[_scope: BASE]
_mod: REP <attribute name> <attribute type> <attribute
value>
_mod: REP <attribute name> <attribute type> <attribute
value>
...
_mod: REP <attribute name> <attribute type> <attribute
value>
<blank line>
```

The format of the SET response message is:

```
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI <distinguished name>
<blank line>
[_inv_id: <invoke ID>]
_end_resp: SET
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

To set the input and output speed to 1200, issue the following request:

```
_cmd: SET
_obj_class: itilink
_obj_id: compId NI pe 7 pi 7 po 1 iti 0 ITILINK 0
_mod: REP inputspeed I 1200
_mod: REP outputspeed I 1200
<blank line>
```

If the SET request message is successful, the response message is:

```
_obj_class: itilink
_obj_id: compId NI PE 7 PI 7 PO 1 ITI 0 ITILINK 0
<blank line>
_end_resp: SET
_time: 1993 08 19 17 14 18
<blank line>
```

The CREATE command

The CREATE command forms a new managed object. A CREATE request results in either one successful CREATE response message and zero or more Informational or Warning error messages, or error message(s) followed by an end-of-response message. The CREATE command builds a service data component. The attribute lines provide explicit values; all other attributes are set to their default values.

The CREATE request automatically creates all mandatory subcomponents that satisfy one of the following conditions:

- Unprovisionable subcomponents. For example, the Process Type envelope component for the X25 Service.
- Fields under the envelope component.

  Note: The _ref_obj_id: and _sup_obj_id: lines are not supported.

Using the CREATE command

The format of the CREATE request message is:

```
_cmd: CREATE
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI <distinguished name>
[_attr: <attribute name> <attribute type>\
<attribute value>]
[_attr: <attribute name> <attribute type>\
<attribute value>]
...
[_attr: <attribute name> <attribute type>\
<attribute value>]
<blank line>
```

The format of the CREATE response message is:

```
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI <distinguished name>
<blank line>
[_inv_id: <invoke ID>]
_end_resp: CREATE
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

Example

To create an ITI service data component PE 7 PI 7 PO 6 ITI 0 ITIDNACUG X30213400410610, issue the following request:

```
_cmd: CREATE
_obj_class: itidnacug
_obj_id: compId NI PE 7 PI 7 PO 1 ITI 0 ITILDNA
X30213400410610 ITIDNACUG X3021340041061
_attr: allowedpacketsize SS 128 256 512
_attr: extendedaccountclass I 0
```

If the CREATE request is successful, the response message is:

```
_obj_class: itidnacug
_obj_id: compId NI PE 7 PI 7 PO 1 ITI 0 ITILDNA
X30213400410610 ITIDNACUG X3021340041061
<blank line>
_end_resp: CREATE
_time: 1993 08 19 17 14 18
<blank line>
```

The DELETE command

The DELETE command deletes an object and all its subordinates recursively. A DELETE request message results in either one successful DELETE response message and zero or more Informational or Warning error messages, or error message(s) followed by an end-of-response message.

Using the DELETE command

The format of the DELETE request message is:

```
_cmd: DELETE
[_inv_id: <invoke ID>]
[_obj_class: <object class>]
_obj_id: compId NI <distinguished name>
<blank line>
```

The format of the DELETE response message is:

```
[_inv_id: <invoke ID>]
_obj_class: <object class>
_obj_id: compId NI <distinguished name>
<blank line>
[_inv_id: <invoke ID>]
_end_resp: DELETE
_time: <year> <month> <day> <hour> <minute> <second>
<blank line>
```

Example

To delete the ITI service data component PE 7 PI 7 PO 1, issue the following request:

```
_cmd: DELETE
_obj_class: iti
_obj_id: compId NI PE 7 PI 7 PO 1 ITI 0
<blank line>
```

If the DELETE request is successful, the response message is:

```
_obj_class: iti
_obj_id: compId NI PE 7 PI 7 PO 1 ITI 0
<blank line>
_end_resp: DELETE
_time: 1993 08 19 17 14 18
<blank line>
```

Propagation log files

Propagation log files are created through Component Provisioning and contain changes that have been made to the service data. Using the propagate command, this file allows the API User to edit the service data in a bundle, or apply the same changes to multiple bundles. This is useful in software migration, where service data changes to many bundles are often required. These log files may be edited but do not need to be if the propagate or REGISTER commands are used.

> ***Note:*** These log files can be used as a starting template for developing customized applications. They capture the series of commands issued in Component Provisioning User Interface (UI), but may contain extra or redundant API commands that would not be present in a well-written API application.

The Propagation log file can be used through

- the Register script file
- the Propagate script file,
- direct API Provider invocation

The same Propagation log file is used as an example for these three methods.

Sample Propagation log file

For the following examples, use the api.cmd.file command file shown below with the comment lines (remove the asterisks).

```
** Thu Feb 3 14:11:51 1994
*_cmd: REGISTER
*_inv_id: 1
*_user_id: chrisw
*_password: ********
*
_cmd: ACTION
_inv_id: 2
_obj_class: pm
_obj_id: compId NI PM R34
_action_type: upload
_attr: accessmode S UPDATE
_attr: location SS DISK
_attr: view SS bundle 4998
_attr: namsid I 4034
<blank line>
_cmd: SET
_inv_id: 3
_obj_class: itilink
_obj_id: compId NI PE 5 PI 5 PO 1 ITI 0 ITILINK 0
_mod: REP inputspeed I 4800
_mod: REP outputspeed I 4800
<blank line>
_cmd: ACTION
_inv_id: 4
_obj_class: pm
_obj_id: compId NI PM R34
_action_type: download
_attr: location SS DISK
_attr: view SS key CHRISW
_attr: mode S INCREMENTAL
_attr: check S INCREMENTAL
_attr: activatable S YES
_attr: namsid I 4034
<blank line>
*_end:
*
```

Use of the Propagation log file through the Register script file

To use this Propagation log file through the Register script file, refer to the following steps.

1. Edit the upload and download criteria as required.
2. Change the comment lines of the file into commands by deleting the leading asterisks.

Then enter:

```
/opt/MagellanNMS/bin/register
```

and respond to the prompts in the following example:

```
API command file = prop.log
NCS access? [y/n] n
```

A typical response is:

```
NOTE: MCF MC.4998.4034.0 uploaded.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
LINKSETUPTIMER:1 was out of range and has been reset
to the specified value.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
DIALINTIMER:30 was out of range and has been reset
to the specified value.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
INPUTSPEED:1200 has been added.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
OUTPUTSPEED:1200 has been added.

WARNING: CONMAN_SDA 0: A "Data Collection"
"ICON_Mnemonic" is required under the "Console_
Manager" component.

NOTE: MCF MC.CHRISW05.4034.0 downloaded.
```

Use of the Propagation log file through the Propagate script file

To use this Propagation log file through the Propagate script file, enter:

```
/opt/MagellanNMS/bin/propagate
```

and respond to the prompts in the following example:

```
Propagation log file = prop.log
Upload and download mode [(u)ser specified/(k)eyed/
(d)ated; default: (u)] = u
Source upload bundle = 4998
Target download bundle = tstapi30
```

A typical response is:

```
NOTE: MCF MC.4998.4034.0 uploaded.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0\
LINKSETUPTIMER:1 was out of range and has been reset
to the specified value.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
DIALINTIMER:30 was out of range and has been reset
to the specified value.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
INPUTSPEED:1200 has been added.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
OUTPUTSPEED:1200 has been added.

WARNING: CONMAN_SDA 0: A "Data Collection"
"ICON_Mnemonic" is required under the "Console_
Manager" component.

NOTE: MCF MC.TSTAPI30.4034.0 downloaded.
```

Respond to the prompts in the following example:

```
Exit [(x)] or
Upload and download mode [(u)ser specified/
(k)eyed/(d)ated; default: (u)] = u
Source upload bundle = 4998
Target download bundle = tstapi40
```

A typical response is:

```
NOTE: MCF MC.4998.4034.0 uploaded.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
LINKSETUPTIMER:1 was out of range and has been reset
to the specified value.
```

```
NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
DIALINTIMER:30 was out of range and has been reset
to the specified value.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
INPUTSPEED:1200 has been added.

NOTE: The field PE 5 PI 5 PO 1 ITI 0 ITILINK 0
OUTPUTSPEED:1200 has been added.

WARNING: CONMAN_SDA 0: A "Data Collection"
"ICON_Mnemonic" is required under the "Console_
Manager" component.

NOTE: MCF MC.TSTAPI40.4034.0 downloaded.
```

Respond to the prompts in the following example:

```
Exit [(x)] or
Upload and download mode [(u)ser specified/
(k)eyed/(d)ated; default: (u)] = x
```

Use of the Propagation log file through API Provider direct invocation

To invoke this Propagation log file, refer to the following steps.

1. Change the REGISTER comment at the top of the file into a command by deleting the leading asterisk (*).
2. Insert the proper user identifier, password, and if applicable, the destination mnemonic, capability identifier, and capability password.
3. Edit the upload and download criteria as required.
4. Change the comment lines of the file into a commands by deleting the leading asterisks.

Invoke the API Provider by redirecting standard input to the edited Propagation log file. For example, direct a file containing API commands into the API Provider as shown below:

```
/opt/MagellanNMS/bin/provapi -echo <last.prop.editted
```

A typical response follows. The _version line is on one line on the UNIX screen.

```
_version: 8.0 Provisioning API Copyright Nortel
Networks 1994
<blank line>
** Thu Feb 3 14:11:51 1994
_cmd: REGISTER
_inv_id: 1
_user_id: user1
_password: secretwd
<blank line>
_inv_id: 1
<blank line>
_inv_id: 1
_end_resp: REGISTER
_time: 1994 02 11 14 44 08
<blank line>
_cmd: ACTION
_inv_id: 2
_obj_class: pm
_obj_id: compId NI PM R34
_action_type: upload
_attr: accessmode S UPDATE
_attr: location SS DISK
_attr: view SS bundle 4998
_attr: namsid I 4034
<blank line>
_inv_id: 2
_obj_class: pm
_obj_id: compId NI PM R34
_attr: mcfname S MC.4998.4034.0
_attr: sdversion I 7
<blank line>
_inv_id: 2
_end_resp: ACTION
_time: 1994 02 11 14 44 40
<blank line>
_cmd: SET
_inv_id: 3
_obj_class: itilink
_obj_id: compId NI PE 5 PI 5 PO 1 ITI 0 ITILINK 0
_mod: REP inputspeed I 4800
```

```
_mod: REP outputspeed I 4800
<blank line>
_inv_id: 3
_end_resp: SET
_time: 1994 02 11 14 44 49
<blank line>
_cmd: ACTION
_inv_id: 4
_obj_class: pm
_obj_id: compId NI PM R34
_action_type: download
_attr: location SS DISK
_attr: view SS key CHRISW
_attr: mode S INCREMENTAL
_attr: check S INCREMENTAL
_attr: namsid I 4034
<blank line>
_inv_id: 4
_error: APPLICATION_ERROR
_attr: errorSeverity E Warning
_attr: errorCode S "ES_7A12"
_attr: errorApplicationId S "PRO_FA"
_attr: errorFacility S "SEMANTIC"
_attr: errorText FS "WARNING: CONMAN_SDA 0: A \"Data\
Collection\" \"ICON_Mnemonic\" is required under the\
\"Console_Manager\" component."
_attr: errorInformation SS CONMAN_SDA 0
<blank line>
_inv_id: 4
_obj_class: pm
_obj_id: compId NI PM R34
_attr: mcfname S MC.CHRISWO3.4034.0
<blank line>
_inv_id: 4
_end_resp: ACTION
_time: 1994 02 11 14 45 32
<blank line>
_end:
<blank line>
_end: USER_REQUEST
```

Chapter 4
API model difference tool

This section describes the tool you can use to identify the differences between two Provisioning API model files. This section contains the following:

- "About the API model difference tool" (page 67)
- "Model files" (page 67)
- "Command line syntax" (page 68)
- "Error messages" (page 70)

About the API model difference tool

The API model difference tool is a UNIX line command that compares two Provisioning API model files and reports any differences. By default, this tool reports on all templates, constructs, service data components, and service data attributes in the model files. If you do not need to know everything that has changed, you can restrict the report to templates, constructs, service data components, and service data attributes that you specify.

Model files

The Provisioning API model file for the currently installed release is /opt/MagellanNMS/lib/api/provapi.model.

You need to save versions of this file from previous releases to use them with the difference tool.

Command line syntax

The format for the difference tool command is:

```
/opt/MagellanNMS/bin/dpnapidiff [-h | -help]
                                or
   [-components <components list file>]
   [-attributes <attributes list file>]
   [-templates <templates list file>]
   [-constructs <constructs list file>]
   [-debug] [-trace] <old file> <new file>
```

where:

`-h | -help` displays command line usage.

`-components` specifies a file that lists the component names for which changes are to be reported. The file contains one component name to a line. If you do not use this option, the difference tool reports changes for all components. You can specify more than one file by repeating this option.

`-attributes` specifies a file that lists the attribute names for which changes are to be reported. The file contains one attribute name to a line. If you do not use this option, the difference tool reports changes for all attributes. You can specify more than one file by repeating this option.

`-templates` specifies a file that lists the template names for which changes are to be reported. The file contains one template name to a line. If you do not use this option, the difference tool reports changes for all templates. You can specify more than one file by repeating this option.

`-constructs` specifies a file that lists the construct names for which changes are to be reported. The file contains one construct name to a line. If you do not use this option, the difference tool reports changes for all constructs. You can specify more than one file by repeating this option.

`-debug` writes high level debugging information to stderr.

`-trace` writes detailed low level information to stderr. This option generates a great deal of output.

`old file` specifies the old Provisioning API model file.

`new file` specifies the new Provisioning API model file.

You must always list the old file before the new file. Compressed files (suffixes .Z, .z, or .gz) can be used for either the old file or the new file.

The output from the difference tool is written to stdout. You can use standard redirection to write the output to a file. Informational messages are written to stdout. Error messages are written to stderr and the difference tool terminates.

Example

```
/opt/MagellanNMS/bin/dpnapidiff \
provapi.model.096pai.gz provapi.model.106pbg.gz
```

A sample of the response is:

```
Changed DPN API Data Model file: provapi.model.096pai.gz
                              to: provapi.model.106pbg.gz

Changed: OBJECT CLASS PM
   Changed: MAY CONTAIN
      Deleted: PE_SERVER                CREATE DELETE GET
        Added: MPANL_PREFIX             CREATE DELETE GET
...
Changed: Name Binding SCRPREFIXGROUPPARAMETERS under
         SCRPREFIXGROUP
   Deleted: * Mandatory child: NO
   Deleted: * Allowed instants: “225”

Changed: ATTRIBUTE TYPE p003
   Changed: LEGAL VALUUES
        Added: 1

Changed: ATTRIBUTE TYPE p007
   Changed: LEGAL VALUES
        Added: 16

Changed: ATTRIBUTE TYPE p013p126
   Changed: LEGAL VALUES
        Added: 3

Changed: ATTRIBUTE TYPE MI8_PROFILEGROUPPARAMETERS
   Changed: * UI VALID value: “1..255”
                           to: “1..254”
```

```
Changed: ATTRIBUTE TYPE x121minnatllength
   Changed: LEGAL VALUES
      Deleted: 4..15
        Added: 2..15

Changed: ATTRIBUTE TYPE moduleidentification
   Changed: LEGAL VALUES
        Added: 'Stub-AM'

Changed: ATTRIBUTE TYPE shelftype
   Changed: LEGAL VALUES
        Added: 'MAS_6X_shelf'
        Added: 'MPA_LDM_shelf'
        Added: 'MPA_6X_shelf'

Changed: ATTRIBUTE TYPE loadpetype
   Changed: LEGAL VALUES
        Added: 'MAS_6X_PE486'
        Added: 'MPA_PE486'
...
```

Error messages

The table "Error messages for difference tool" (page 70) lists the error messages that the difference tool can generate. The $<n> items indicate values that are filled in at run time.

Table 2
Error messages for difference tool

Message and Action
`ERROR: Invalid option: $0` The command line parameters are incorrectly specified. Correct and retry.
`ERROR: More than two files specified.` The model files are incorrectly specified. Correct and retry.
`ERROR: Less than two files specified.` The model files are incorrectly specified. Correct and retry.
(Sheet 1 of 4)

Table 2 (Continued)
Error messages for difference tool

Message and Action
`ERROR: Missing filename for -components option.` Correct and retry.
`ERROR: Missing filename for -attributes option.` Correct and retry.
`ERROR: Missing filename for -templates option.` Correct and retry.
`ERROR: Missing filename for -constructs option.` Correct and retry.
`ERROR: Unable to open $0: $1` The indicated model file, components list file, attributes list file, templates list file, or constructs list file ($0) cannot be opened. The UNIX error message ($1) should provide the information to determine the cause of the problem and how to correct it.
`ERROR: Invalid template header line in $0 ($1): $2` The first line in a template does not have the expected format. The filename ($0), line number ($1), and text of the line in error ($2) are provided. Verify that the correct model files are specified on the command line. If there is an obvious error in the model file, you can manually edit the file. However you still need to report the problem to the System Administrator.
`ERROR: Invalid construct line in $0 ($1): $2` A construct line does not have the expected format. The filename ($0), line number ($1), and text of the line in error ($2) are provided. Verify that the correct model files are specified on the command line. If there is an obvious error in the model file, you can manually edit the file. However you still need to report the problem to the System Administrator.
(Sheet 2 of 4)

Table 2 (Continued)
Error messages for difference tool

Message and Action
`ERROR: Missing '}' in $0 ($1): $2` A set of values, started by an opening brace bracket ({) is not terminated by a closing brace bracket (}). The filename ($0), line number ($1), and text of the line in error ($2) are provided. Verify that the correct model files are specified on the command line. If there is an obvious error in the model file, you can manually edit the file. However you still need to report the problem to the System Administrator.
`ERROR: Invalid line in $0 ($1): $2` Indeterminate parsing error. The filename ($0), line number ($1), and text of the line in error ($2) are provided. Verify that the correct model files are specified on the command line. If there is an obvious error in the model file, you can manually edit the file. However you still need to report the problem to the System Administrator.
`ERROR: Missing terminating quote in $0 ($1): $2` The terminating quote for a quoted string is missing, or there are illegal quotes in the line. The filename ($0), line number ($1), and text of the line in error ($2) are provided. Verify that the correct model files are specified on the command line. If there is an obvious error in the model file, you can manually edit the file. However you still need to report the problem to the System Administrator.
(Sheet 3 of 4)

Table 2 (Continued)
Error messages for difference tool

Message and Action
`ERROR: Premature EOF on $0` Verify that the correct model files are specified on the command line. If there is an obvious error in the model file, you can manually edit the file. However you still need to report the problem to the System Administrator.
`ERROR: Insufficient memory` The workstation may be overloaded or misconfigured. Another workstation may be able to run the tool, or the same workstation may be able to run the tool at another time. Contact the System Administrator regarding the configuration of the workstation.
(Sheet 4 of 4)

Appendix A
DPN Provisioning API object definitions

An online file containing information on the DPN Provisioning object model is provided; this file is called /opt/MagellanNMS/lib/api/provapi.model. This file also contains object and attribute names as they appear in the DPN Provisioning API model. For a description of this file, see “Object model” (page 23).

Appendix B
Compliance statement

This appendix indicates how the Feature Specification of a DPN Provisioning API conforms to the API Primer.

This appendix contains the following:

- "API message types" (page 78)
- "API message lines" (page 79)
- "API data types" (page 84)
- "Sieve object support" (page 84)

Each box in the Supported column of the tables is filled with one of the following letters:

- Y
  yes, the item is supported
- N
  no, the item is not supported
- C
  conditional, the item is supported under certain conditions. In this case, the condition is described.
- N/A
  not applicable

API message types

The table “API message types” (page 78) indicates the messages that are supported by the DPN Provisioning API. This means that the message can be generated or received with all the mandatory lines. The Direction column indicates whether the message is sent or received by the API Provider.

Table 3
API message types

Message type	Direction	Supported
GET request	receive	Y
GET response	send	Y
SET request	receive	Y
SET response	send	Y
ACTION request	receive	Y
ACTION response	send	Y
CREATE request	receive	Y
CREATE response	send	Y
DELETE request	receive	Y
DELETE response	send	Y
REGISTER request	receive	Y
REGISTER response	send	Y
DEREGISTER request	receive	Y
DEREGISTER response	send	Y
EVENT-REPORT message	send	N
end-of-response message	send	Y
error message	send	Y
block message	send	N
version message	send	Y

(Sheet 1 of 2)

Table 3 (Continued)
API message types

Message type	Direction	Supported
end message	send	Y
end message	receive	Y
(Sheet 2 of 2)		

API message lines

The table “API message lines” (page 79) indicates the message lines and keywords that are supported by the DPN Provisioning API. It describes overall support for the message line. The table “Optional message lines” (page 82) indicates the optional message lines that are supported by the DPN Provisioning API for each API message type and for each API message line. Message lines that are mandatory are listed in this table, since there may be some conditions and restrictions when used.

Table 4
API message lines

Message line	Keywords	Supported
_action_type	unload, download, verify, discard, quit	Y
_attr		Y
_attr_id	general support ALL	Y Y
_block		N
_capability		N
_cmd	REGISTER, DEREGISTER, GET, SET, CREATE, DELETE, ACTION	Y (see also Table 2)
_echo	ON, OFF	Y
(Sheet 1 of 4)		

Table 4 (Continued)
API message lines

Message line	Keywords	Supported
_end	general support USER_REQUEST FATAL_ERROR SYSTEM_SHUTDOWN LOST_CONNECTION LOST_SESSION	Y Y Y N N N
_end_block		N
_error	general support ACCESS_DENIED APPLICATION_ERROR UNRECOVERABLE_ERROR DUPLICATE_OBJECT INVALID_ACTION_TYPE INVALID_ATTRIBUTE_NAME INVALID_ATTRIBUTE_VALUE INVALID_FILTER INVALID_OBJECT_ID INVALID_OBJECT_TYPE INVALID_SCOPE LOGON_FAILS MISSING_ATTRIBUTE_VALUE MODIFICATION_ERROR NO_SUCH_OBJECT_ID OUT_OF_SEQUENCE	Y N Y Y N N N N N N N N N N N N Y
_event_type		N
(Sheet 2 of 4)		

Table 4 (Continued)
API message lines

Message line	Keywords	Supported
_filter	general support P EQ NE LT GT LE GE LEFT MIDDLE RIGHT IN NOTIN	N
_inv_id		Y
_mod	general support ADD DEL DEF REP	Y N N N Y
_obj_id		Y
_obj_class		Y
_password		Y
_ref_obj_id		N
_scope	general support BASE NEXT ALL	C[1] Y Y Y
_sieve_id		N
_sup_obj_id		N
_time		Y
_trace	ON, OFF	Y
(Sheet 3 of 4)		

Table 4 (Continued)
API message lines

Message line	Keywords	Supported
_user_id		Y
_version		Y
(Sheet 4 of 4)		

1 See the table “Optional message lines” (page 82) for details.

Table 5
Optional message lines

Message type	Message line	Supported
GET request	_inv_id _scope _filter _attr_id	Y Y N Y
GET response	_inv_id _attr	Y Y
SET request	_inv_id _scope _filter _mod	Y C[1] N C[1]
SET response	_inv_id _attr	Y N
ACTION request	_inv_id _scope _filter _attr	Y N N C[2]
ACTION response	_inv_id _attr	Y C[3]
CREATE request	_inv_id _obj_id _sup_obj_id _ref_obj_id _attr	Y Y N N Y
(Sheet 1 of 2)		

Table 5 (Continued)
Optional message lines

Message type	Message line	Supported
CREATE response	_inv_id	Y
DELETE request	_inv_id _scope _filter	Y C[4] N
DELETE response	_inv_id	Y
REGISTER request	_inv_id _user_id _password _attr	Y Y Y Y
REGISTER response	_inv_id _user_id _capability _attr	Y N N N
DEREGISTER request	_inv_id	Y
DEREGISTER response	_inv_id _user_id _capability	Y N N
EVENT-REPORT message	_attr	N/A
end-of-response message	_inv_id	Y
error message	_inv_id _attr	Y Y
(Sheet 2 of 2)		

1 The SET request only supports BASE scope.

2 The SET request only supports REP in the _mod: line. A SET request must include at least one _mod: line.

3 There are _attribute: line(s) in the upload and download actions, but not in the quit, discard, and verify actions.

4 The DELETE request only supports ALL scope.

API data types

The table “API data types” (page 84) indicates the API data types supported by the DPN Provisioning API. In this table, supported means that the base software to support the data type is present, whether or not there are actually any attributes of this type in the current object model.

Table 6
API data types

API data type	Supported
B (Boolean)	Y
I (Integer)	Y
H (Hexadecimal)	Y
D (Date/time)	N
S (String)	Y
FS (Formatted String)	Y
NI (Node Identifier)	Y
LI (Link Identifier)	N
E (Enumerated)	C[1]
SS (Sequence of Strings)	Y
SI (Sequence of Integers)	Y
SB (Sequence of Booleans)	Y
RS (Range of Strings)	N
RI (Range of Integers)	Y
RD (Range of Data)	N

1 The E (enumerated) data type is only in the errorSeverity attribute in the error messages.

Sieve object support

The table “Sieve object attributes” (page 85) indicates the attributes of the sieve object that are supported by the DPN Provisioning API.

Table 7
Sieve object attributes

Attribute	Supported
general support	N
sieveId	N
eventFilter	N
admState	N
annotation	N
eventInfo	N
repOClass	N
repOid	N
repScope	N
repFilter	N
repInfo	N

Appendix C
Error messages

This appendix describes the error messages produced by the DPN Provisioning API. This appendix contains the following:

- "Response to undesired events" (page 87)
- "Error messages" (page 88)

Response to undesired events

When an error occurs, one or more error messages are returned. If the error is not recoverable (by re-issuing a corrected request from the customer application), an _end: line is also returned and the API Provider terminates.

For grammar and syntax errors, only one error message is returned (for the first error), and the rest of the command message is purged.

The following error return codes are returned for errors detected by the Provisioning API:

- INIT_ERROR
  occurs during API Provider initialization. The version number is included on the _error: line. Since the version message is not output (as initialization is not complete), the version number is included so the customer application can determine the format of the error message.
- SYNTAX_ERROR
  occurs when the request data does not conform to the syntax specified for the request.

- NOT_SUPPORTED
  occurs when the specified element, although part of the overall API grammar, is not supported by the DPN Provisioning API.
- OUT_OF_SEQUENCE
  occurs when the request is not valid within the current context (for example, an upload action before a REGISTER request, or a GET or SET request before a successful upload action).
- APPLICATION_ERROR
  indicates that the request was syntactically correct, but semantically incorrect and the Field Access has returned an error Protocol Data Unit (PDU) in response to the request.

If an error PDU is returned from the Field Access application, the error response data consists of the text resulting from the formatting of the error qualifiers in the error PDU.

For lost sessions or connections, the API Provider returns an error message, followed by an end message, and then terminates.

Error messages

The table “Error messages” (page 88) lists the error messages that can be generated by the DPN Provisioning API. The $<n> items indicate values that are filled in at run-time.

Table 8
Error messages

Code	Description and possible solution
A0001	`Unknown session status received: $0` Report this internal error to your system administrator.
A0002	`Unknown transaction status received: $0` Report this internal error to your system administrator.

(Sheet 1 of 9)

Table 8 (Continued)
Error messages

Code	Description and possible solution
A0003	`Fork failure: $0` It was not possible to start up the server (/opt/MagellanNMS/bin/sea by default). Either the UNIX process limit is too low and needs to be set to a higher value, or the workstation is overloaded, and the number of processes that are running needs to be reduced (rescheduling may need to be considered). The UNIX system error message is substituted for $0 and should provide additional information.
A0004	`PDU failure.` Report this internal error to your system administrator.
A0005	`Lost connection.` This error could be caused by a software failure in fa, sea, or pfas, and should therefore be reported to your system administrator. It may also be due to a connectivity problem between the processes that need to be involved in the running of this application. Check with your local UNIX administrator to correct any local network problems, and/or UNIX software problems and retry.
A0007	`Out of memory.` Either there is not enough real memory, and/or swap space and these need to be increased, or the workstation is overloaded and the work load needs to be reduced or rescheduled.
A0008	`Improperly terminated string.` A quoted string does not end with a quote, or it does not end with the same (single or double) quote with which it began. Quoted strings must begin and end with the same kind of quote (single or double).
A0009	`Unknown connection status received: $0` Report this internal error to your system administrator.
A0010	`New transaction on session.` Report this internal error to your system administrator.
A0011	`New session on connection.` Report this internal error to your system administrator.
(Sheet 2 of 9)	

Table 8 (Continued)
Error messages

Code	Description and possible solution
A0012	`Invalid option: $0` Check the entered command line for spelling mistakes. Reread the descriptions of the command line parameters. Re-enter a corrected command line.
A0013	`Unable to start up $0 with args $1 $2 : $3` UNIX was not able to start up the server (value substituted for $0). If the -server option was specified, then the name of the server may have been misspelled or the server may not have the necessary permissions (example, no execute permission). Correct and retry. If the -server option was not specified, then the default server is /opt/MagellanNMS/bin/fa, which means that MDM Provisioning was not installed properly. Either the fa image is missing, or has the wrong permissions. Make sure that MDM Provisioning has been installed properly and retry. The UNIX system error message is substituted for $3 and should provide additional information.
A0014	`Unable to initialize PROP PDO/PDU.` This error message may be the result of a problem associated with the MDM servers, or an internal software error. To isolate the cause, attempt to start the DPN Component Provisioning Tool. If you are able to start the DPN Component Provisioning Tool, there may be an internal software error. Report this internal error to your system administrator. If you are unable to start the DPN Component Provisioning Tool, there may be a problem with the MNSD, HGDS, NCSMGR, or the PFAS server. Check the System Log Display for errors related to these servers, then use the Server Manager Administration tool to verify that the servers are running. Restart the servers as necessary. Next, check the configuration of these servers against the information provided in 241-6001-310 *Preside MDM Server Reference Guide* and ensure that these servers are configured correctly. If this does not correct the problem, report it to your system administrator, and indicate any corrective action you have taken.
A0015	`Missing server name.` The -server option was specified without a server name. Re-enter the command and specify the name of the server for the -server option.

(Sheet 3 of 9)

Table 8 (Continued)
Error messages

Code	Description and possible solution
A0016	`Missing fa/sea log file name.` The -fasealog option was specified without a file name. Re-enter the command and specify the name of the file for the -fasealog option.
A0017	`Missing width.` The -width option was specified without a width. Re-enter the command and specify the width for the -width option.
A0018	`Invalid width: $0` The specified width is outside the range 72 to 100000, or it is non-numeric. Re-enter the command with a numeric width that is within the range.
A0019	`Unable to connect to server.` The server failed shortly after startup. Check the log file and/or console for error messages to determine the cause of the failure.
A0020	`Unable to determine login userid.` Something is wrong with the UNIX environment. Check with the UNIX system administrator to determine the cause and the solution.
A0101	`Missing _action_type: line.` An action request message does not have an _action_type: line. Re-enter the action request message specifying the desired action on an _action_type: line.
A0104	`Invalid invoke ID: $0` Invoke IDs must be numeric digits. Re-enter the request message in an _inv_id: line that has only numeric digits in the invoke ID.
A0107	`Invalid keyword in place of compId keyword: $0` Only the keyword compId is permitted on the _obj_id: line. Correct and re-enter.
A0108	`Invalid compId type: $0` Only the NI type is permitted on the _obj_id: compId line. Correct and re-enter.
A0109	`Missing $0` Some syntactical element is missing from a line. The phrase substituted for $0 specifies the element that is missing. Re-enter with the element filled in.
(Sheet 4 of 9)	

Table 8 (Continued)
Error messages

Code	Description and possible solution
A0110	`Invalid $0: $1` Some syntactical element has not been properly formed. A phrase describing the element is substituted for $0. The actual token in error is substituted for $1.
A0111	`Improperly formed SET type value.` Correct and re-enter.
A0112	`Improperly formed RANGE type value.` Correct and re-enter
A0113	`Unsupported $0: $1` The syntactical element described by the phrase substituted for $0 is not supported by the Provisioning API. The unsupported token is substituted for $1. Correct and re-enter.
A0114	`Improperly formed SEQUENCE of INTEGERS type.` Correct and retry.
A0115	`Improperly formed RANGE of INTEGERS type.` Correct and retry.
A0202	`Invalid component id.` An odd number of tokens was specified for a component ID. A component ID consists of category and key value pairs, and must have an even number of tokens. Correct and re-enter.
A0301	`REGISTER request is not valid in current context.` The REGISTER request is valid only at startup or after a successful DEREGISTER request. Correct and re-enter.
A0302	`Invalid Unix userid/password specification.` The specified UNIX user ID and password specified in the REGISTER request were not validated by UNIX. Use a correct UNIX user ID and password combination. If the API Provider is not used as a network server, then you can specify the -nouserid option and omit the _user_id: and _password: lines from the REGISTER requests.
(Sheet 5 of 9)	

Table 8 (Continued)
Error messages

Code	Description and possible solution
A0303	`Incomplete NCS access specification.` When specifying the NCS access, all three NCS access attributes must be specified. Correct and retry. If NCS access is not desired, then do not specify any of the NCS access attributes.
A0401	`Upload action is not valid in the current context.` The upload action is valid only after a successful REGISTER request, download action, or quit action request. Correct and retry.
A0402	`Unable to extract MCF name from Open Session reply.` Report this internal software error to your system administrator.
A0403	`Unable to extract Service Data Version from Open Session reply.` Report this internal software error to your system administrator.
A0404	`Location attribute is not allowed with sdversion attribute.` When creating all new Service Data, the location of the Service Data is not specified on the upload action request. The location is specified on the download action request. Correct and retry.
A0405	`View attribute is not allowed with sdversion attribute.` When creating all new Service Data, the view is not specified on the upload action request. The view is specified on the download action request. Correct and retry.
A0406	`Update mode must be specified with sdversion attribute.` When creating all new Service Data, there is nothing to read initially, and so read access is disallowed. Correct and retry.
A0407	`ACTIVE view is not compatible with DISK location.` A view is designated ACTIVE only on PMs. This designation is not available for Service Data stored on MDM disk. Correct and retry.
(Sheet 6 of 9)	

Table 8 (Continued)
Error messages

Code	Description and possible solution
A0408	`COMMITTED view is not compatible with DISK location.` A view is designated COMMITTED only on PMs. This designation is not available for Service Data stored on Preside Multiservice Data Manager (MDM) disk. Correct and retry.
A0409	`NAMS ID must be specified with DISK location.` When Service Data is on PM disk, the NAMS ID can default to that of the PM, but with MDM disk, this is not possible. Correct and retry.
A0410	`NAMS ID specification is not compatible with ACTIVE view.` Remove the NAMS ID specification or select a bundle or key view.
A0411	`NAMS ID specification is not compatible with COMMITTED view.` Remove the NAMS ID specification or select a bundle or key view.
A0501	`GET request is not valid in the current context.` GET requests are valid only after a successful upload action request, until a successful download, or quit action request. Correct and retry.
A0502	`Requested object(s) not found.` This is probably caused by naming a Service Data field as if it were an object rather than an attribute. If the component does not really exist, then a different error message and code from fa is presented. Remove the last category and key value pair from the _obj_id: line. If all the attributes are not requested, then put that last category in an _attr_id: line.
A0601	`SET request is not valid in the current context.` SET requests are valid only after a successful upload action request, until a successful download, or quit action request. Correct and retry.
A0602	`Modifications missing from SET request.` At least one modification must be specified in a SET request. Correct and retry.
A0603	`Only BASE scope is supported for the SET request.` Either remove the _scope: line or specify BASE.
(Sheet 7 of 9)	

Table 8 (Continued)
Error messages

Code	Description and possible solution
A0701	`CREATE request is not valid in the current context.` CREATE requests are valid only after a successful upload action request, until a successful download, or quit action request. Correct and retry.
A0801	`DELETE request is not valid in the current context.` DELETE requests are valid only after a successful upload action request, until a successful download, or quit action request. Correct and retry.
A0901	`Download action is not valid in the current context.` Download action requests are valid only after a successful upload action request, until a successful download, or quit action request. Correct and retry.
A0902	`Unable to extract MCF name from Download reply.` Report this internal software error to your system administrator.
A0903	`ACTIVE view cannot be downloaded.` Specify bundle or key view on download action requests.
A0904	`COMMITTED view cannot be downloaded.` Specify bundle or key view on download action requests.
A0905	`Incremental Download not allowed without Service Data changes.` An Incremental download only downloads changed Service Data. If there are no changes, there is nothing to download. Specify a Complete download, use a quit action request instead of the download action request, or make some Service Data changes.
A0906	`Both date key view and activation date attribute may not be specified.` Specify either a date key view or an activation date, but not both.
A0907	`NAMS ID must be specified when downloading newly created PM.` Specify the NAMS ID in the download action request.
A1001	`Quit action is not valid in the current context.` The quit action request is valid only after a successful upload action request and until a successful quit or download action request. Correct and retry.
(Sheet 8 of 9)	

Table 8 (Continued)
Error messages

Code	Description and possible solution
A1101	`Missing object id.` Many API requests operate on a specific Service Data component and that component needs to be explicitly specified in and _obj_id: line. Correct and retry.
A1102	`Unexpected PDU reply code: $0` Report this internal software error to your system administrator.
A1103	`Re-specification of object id rejected.` There is a second _obj_id: line in an API request. Remove it and retry.
A1201	`Discard action request is not valid in the current context.` The discard action request is valid only after a successful upload action request and until a successful quit or download action request. Correct and retry.
A1301	`DEREGISTER request is not valid in the current context.` The DEREGISTER request is valid only after a successful REGISTER request, quit action request, or download action request. Correct and retry.
A1401	`Verify action request is not valid in the current context.` The verify action request is valid only after a successful upload action request and until a successful quit or download action request. Correct and retry.
(Sheet 9 of 9)	

Index

A

ACTION
- discard 20, 51
- download 20, 48
- quit 20, 52
- request 19, 45
- upload 20, 45
- verify 20, 50

action template 28
API 15
- initialization 32
- installation 31
- object model 23

API model difference tool
- command syntax 68–69
- error messages 70–73

API Provider 15
API User 15
application programming interface
- …See API

attributes
- action 28
- component 25
- name binding 29
- object class 25
- template 27

C

command 41
- …See also ACTION
- …See also CREATE
- …See also DELETE
- …See also DEREGISTER
- …See also GET
- …See also REGISTER
- …See also SET

Command Filter parameters 33
communication links 16
component
- name 25
- object class 25
- …See also service data components

component hierarchy 24
Configuration 15
containment hierarchy 23
CREATE
- example 58
- request 19
- request format 58
- response format 58

D

data types 17
DELETE
- example 59
- request 19
- request format 59
- response format 59

DEREGISTER
- example 44

request 19
request format 44
response format 44
difference tool
…See API model difference tool

E

environment 16
errors 87
event report 17

G

GET
example 55
request 19, 53
request format 54
response format 54

H

hierarchy
…See containment hierarchy

I

initialization 32
installation 31

L

log file 60

M

message
error format 18
request format 18
response format 18
types 17

N

name binding template 29

O

object class
in name binding 29
template 25
object model 23

P

Propagation log file 60

R

REGISTER 41
request 19
request format 42
response format 42
Register script file 34
request
message format 18
object class 25
response
message format 18

S

script file 34
service data components 23, 25
attributes 25
component name 25
subordinates 25
service data hierarchy
…See component hierarchy
SET
example 57
request 19
request format 56
response format 57
sieve 17
subordinate components 25

T

template
action 28
attribute 27
name binding 29
object class 25

time line 21

U

UI prompt
attributes 27
object class 26

Preside Multiservice Data Manager

DPN Provisioning API

Reference Guide

Release: R15.1

Publication: 241-6001-204
Document status: Standard
Document version: 15.1RSUP
Document date: August 2004
Printed in Canada

NORTEL NETWORKS