VERTIV.

Avocent®

DSView™ Management and Rack Power Manager Software

Installer/User Guide

Software Development Kit

The information contained in this document is subject to change without notice and may not be suitable for all applications. While every precaution has been taken to ensure the accuracy and completeness of this document, Vertiv assumes no responsibility and disclaims all liability for damages resulting from use of this information or for any errors or omissions. Refer to other local practices or building codes as applicable for the correct methods, tools, and materials to be used in performing procedures not specifically described in this document.

Technical Support Site

If you encounter any installation or operational issues with your product, check the pertinent section of this manual to see if the issue can be resolved by following outlined procedures. For additional assistance, visit https://www.VertivCo.com/en-us/support/.

TABLE OF CONTENTS

1 INTRODUCTION

This guide is part of the Avocent® DSView™ Software and Avocent® Rack Power Manager Software Development Kit (SDK). It is intended to be used by engineers to incorporate the DSView or Rack Power Manager software technology into existing applications.

NOTE: All instances of DSView software within this document refer to DSView software version 4.5 or higher.

1.1 Product Overview

The DSView and Rack Power Manager software products enable you to manage remote devices through secure connections. They are browser-based, enterprise management solutions that work across platforms. The DSView and Rack Power Manager software development kits help you to incorporate core services of the DSView or Rack Power Manager software into applications.

The SDK includes the following Application Program Interfaces (APIs):

- GUI Access API - Using the GUI Access API, your application can request browser sessions that display named screens. The named screens allow your application to launch viewer sessions to target devices.
- Web Services API - Using the Web Services API, your application can access and update information in the DSView software. For example, you can retrieve a list of managed units and add events.
- Samples of APIs

The following figure shows the flow of information between the DSView software, application programs, APIs and units.

Figure 1.1 Flow Chart

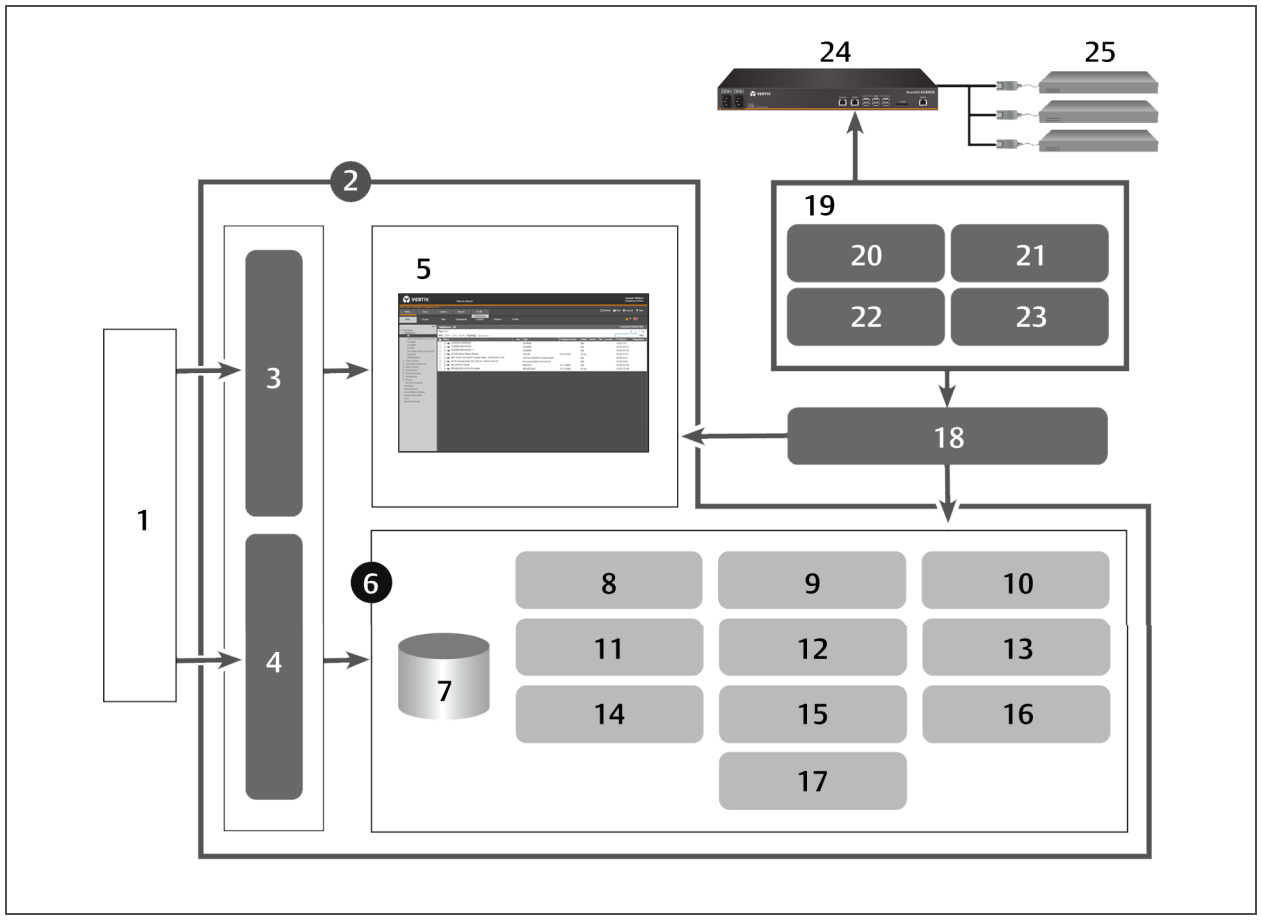

Table 1.1 Flow Chart Component Descriptions

ITEM	DESCRIPTION
1	Third Party - The third party accessing components in the DSView software core
2	DSView Software Core - Components that comprise the DSView software core
3	GUI Access API - Provides the ability to request browser sessions that display named screens
4	Web Services API - Provides the ability to access and update information in the DSView software
5	GUI Console - The console that displays the GUI
6	Core Services - Components that comprise the DSView software core services
7	SQL Database - The SQL database that hosts DSView software data
8	Authentication - The core service that authenticates users requesting access to the DSView software
9	Audit - The core service that audits user and access rights to the DSView software
10	Discovery - The core service that facilitates appliance discovery in the DSView software
11	Authorization - The core service that authorizes
12	Scheduler - The core service that allows users to schedule tasks and actions to take place in the DSView software
13	Firmware Store - The DSView software component that stores the firmware
14	Replication - The DSView software component that allows you to replicate server configuration, settings and characteristics
15	Unit Status - The core service that monitors unit status in the DSView software
16	Licensing - The core service that checks for and validates licensing in the DSView software
17	System Log - The core service that records a log of all system activities in the DSView software
18	Plug-in API - The software element that allows you to control the appliance through the DSView software
19	Plug-in - The module or software element that allows communication between the appliance and the DSView software
20	Navigation - Elements that allow you to navigate the DSView software through the appliance
21	Appliance Logic - Configurations and settings related to the appliance that sends and receives data to the DSView software
22	GUI Content Screens - Screens that are displayed based on user input to the DSView software
23	Module Definition - Provides appliance characteristics and information to the DSView software
24	Appliance - A monitoring appliance that sends and receives data to the DSView software
25	Servers - Servers that are managed by the DSView software

1.2 Platforms

The SDK is supported on the following operating systems:

- Microsoft® Windows® Server 2008, 2008 R2, 2012 and 2012 R2 (64 bit)
- SUSE® Linux® Enterprise Server 11 and OpenSUSE 12.1 (64 bit)
- RedHat® Enterprise Server 6.4, 6.5 and 7 (64 bit)
- Sun™ Solaris SPARC® 10
- CentOS 6.5 and 7 (64 bit)

1.3 DSView™ Software Development Edition

The DSView™ Software Development Edition (SDE) contains the DSView software core and WSDL files that you can import in order to use the SDK APIs.

NOTE: The SDE does not allow spokes.

Unlike the Enterprise DSView™ software, the software development edition:

- Is launched from the command line, not as a service
- Displays specialized debug messages to the command line console
- Allows both HTTP and HTTPS protocols to enable sniffer tracing of API calls
- Enables the legal installation of the SDK on multiple development stations via a separate install license key site license

NOTE: Before you run the SDE, obtain an SDE license key from Vertiv.

To start the SDE:

1. Change to the directory containing the start-up script:
   - If you are starting the SDE in Microsoft® Windows®, enter avocent\dssdk\server\bin\.
   - If you are starting the SDE in Linux®, enter avocent/dssdk/server/bin/.
2. Run the script:
   - For Microsoft Windows, enter dssdk.bat.
   - For Linux, enter dssdk.sh.
3. If there is a port-related error, change the port number to an unused port number between 1 and 65535, inclusive, in the following file:
   - If you are using a Microsoft Windows system, the file is avocent\dssdk\server\bin\ports.properties.
   - If you are using a Linux system, the file is avocent/dssdk/server/bin/ports.properties.

NOTE: The TCP monitor that lets you examine SOAP requests and responses uses port 8080.

4. After the message “DSView successfully started” is displayed, you can minimize the screen.

To run the SDE:

1. Enter http://localhost:8080 (or https://localhost:8080 for a secure connection) in the Address field of a web browser.
2. Use the Server Configuration wizard to configure your DSView software administrator account.
3. Log into the SDE from the User Login screen using the credentials specified in the Server Configuration wizard.
4. Add appliances and target devices.

1.4 Debugging

Debug messages can be helpful when testing an application's SDK API calls. The SDK ships with the debug mechanism disabled; however, the debug mechanism is enabled and configured by running the SDE from the command line with parameters for specific SDK APIs.

The debug output includes, but is not limited to, the display of interface parameters, return results and lifespan information. An output handler displays the SDK API debug information on the command line console. This debug information is independent of the standard DSView software trace debug messages.

To run the SDE with debugging enabled:

1. From the command prompt, change to the directory containing the dssdk startup script.
   - If you are using a Microsoft® Windows® system, change the directory to avocent\dssdk\server\bin.
   - If you are using a Linux® system, change the directory to avocent/dssdk/server/bin.
2. Enter the dssdk script name using the following syntax:

   dssdk "-Dds3.loggers=logger1[[,logger2]...]" "-Dds3.level=level"

NOTE: In the syntax, "logger1,logger2..." signifies a comma-separated list of logger names that can be enabled and "level" signifies the level of detail and type of information displayed for debug logging. See the following table for a list of the levels in descending order. Both parameters are required in the syntax.

3. To enter a logger that displays log messages relating to the GUI access API, enter dsview.sdk.webservice.

   -or-

   Enter dsview.sdk.webservice for a logger that displays log messages relating to the Web Services API.

   -or-

   Enter dsview.sdk.nmmapi for a logger that displays log messages relating to the plug-in API.

Table 1.2 Debug Levels

LEVEL NAME	DESCRIPTION
OFF	Turns off the logging of messages. **NOTE: Debug messages relating to the web server (Jetty) starting or shutting down always appear on the console.**
SEVERE	Serious failures - events of considerable importance that prevent normal program execution. This level is usually for end users and system administrators.
WARNING	Potential problems; this level is usually for end users or system managers.
INFO	Significant informational messages; this level is usually for end users and system administrators.
CONFIG	Static configuration messages; these can assist in debugging problems associated with particular configurations. For example, the CONFIG message might include the CPU type, graphics depth and GUI look-and-feel.
FINE	Moderately detailed tracing information such as minor (recoverable) failures, potential performance problems or information for developers who do not have a specialized interest in the specific subsystem. The exact meaning varies among subsystems. The FINE debug level is not as detailed and returns fewer bugs than FINER and FINEST.
FINER	Fairly detailed tracing message. By default, logging calls for entering, returning or throwing an exception are traced at this level. The exact meaning varies among subsystems. The FINER debug level is more detailed than FINE but not as detailed as FINEST. It returns more bugs than FINE but fewer bugs than FINEST.
FINEST	Highly detailed tracing message. The exact meaning varies among subsystems. The FINEST debug level is the most detailed and returns more bugs than FINE and FINER.
ALL	Enables the logging of all messages.

1.4.1 SDE start command

You can start the SDE with logging enabled for the GUI Access API and the Web Services API. The logging levels include INFO, WARNING and SEVERE.

An example SDE start command with logging enabled is as follows.

```
dssdk "-Dds3.loggers=dsview.sdk.guiaccess,dsview.sdk.webservice" "-Dds3.level=INFO"
```

1.5 DSView™ Software Development Kit Directories

The libs directory contains WSDL (Web Service Description Language) and XSD (XML Schema Definition Language) files that define the Web Services interface to the DSView software. Additional files are provided for services and their common schemas and directories/files for client-side stubs.

NOTE: The sample WSDL files in the libs directory are for reference only. While the SDK is running, retrieve a dynamically generated WSDL file for your Web Services client by visiting the appropriate URL. The URL for each service is noted in Web Services API on page 19.

The samples directory contains sample programs, instructions and script files for each of the interfaces supported by the SDK.

The server directory contains licensing, binary, source, data and log files for the SDK.

2 GUI ACCESS API

The GUI Access API provides a means for third party applications, tools, systems and web browsers to access the DSView and Rack Power Manager software using named screens. Named screens are URLs that are processed by the DSView™ or Rack Power Manager software system to provide the requested functionality. These named screens are submitted to the DSView or Rack Power Manager software using HTTPS and generally contain parameters that are specific for the screen.

The application to be integrated can reside on the same system as the DSView or Rack Power Manager software or on another server. The calling application must provide proper credentials to the GUI Access API.

This chapter describes the named screen requests and responses. It also describes the sample applications.

Syntax conventions

This chapter uses the following syntax conventions.

- Bold type indicates user-supplied values or values to be entered.
- Square brackets ([ ]) enclose optional values.
- A vertical bar (|) separates choices.
- When typing commands, omit brackets, braces and vertical bars.
- Use an ampersand (&) between parameters.

2.1 API Operating Environment

Protocols

The API operating production environment uses HTTPS internet protocol with the Enterprise DSView software.

Web browsers

The web browser must have cookies enabled for all named screens in the DSView software. The following are the supported web browsers:

NOTE: Unless noted otherwise, both 32-bit and 64-bit browsers are supported.

- Microsoft® Internet Explorer® 9, 10 or 11
- Mozilla® Firefox® version 45.0 ESR
- Google® Chrome™ version 50

For the most current list of supported browsers, see the latest product release notes.

Many of the named screen requests return responses that are intended to be directed to a browser. Some of these responses require the browser to be configured appropriately. The GUI Access API Session Types table in Launching a Session on page 13 lists the recommended settings for:

- Browser screen width
- Browser screen height
- Show/hide the browser toolbar

Applications

Your applications can reside on the same system as the DSView™ software server or on another server. Supported languages include, but are not limited to:

- HTML/JavaScript
- Visual C++® (only on Microsoft Windows® systems)
- Visual Basic® (only on Microsoft Windows systems)

Sessions and session reuse

A user session is required for accessing named screens. A user session can be created for each named screen requested by passing in the user credentials (username and password). Alternatively, a user session can be reused by passing a session ID or cookie.

Certificates

By default, the DSView software system uses self-signed certificates for secure transactions. The self-signed certificate can be replaced with a certificate signed by a trusted third party. HTTPS sessions, established for use with the GUI Access API, need to accept the certificate. If the HTTPS session is established using a browser, the browser can present a Security Alert/Warning dialog box.

2.2 HTTP Requests for Named Screens

Named screens are submitted to the DSView software using HTTPS and generally contain parameters that are specific for the screen. The DSView™ software server processes the request and returns an HTTP response. These responses are generally displayed in a browser.

NOTE: Authentication differs from authorization. For an authenticated user, the DSView software server authorizes information access based on configured access rights.

2.2.1 URL format

An example named screen URL format is as follows.

http[s]://dsviewServer[:dsviewPort]/namedPage?[authentication_parameters] [target_parameters]

2.2.2 Parameters

The following table lists the parameters that comprise the URL.

Table 2.1 Named Screen URL Parameters

PARAMETER	DESCRIPTION
HTTP or HTTPS	(Required) When debugging and testing with the SDE, you can specify HTTP; however, when running the Enterprise DSView software, you must specify HTTPS.
dsviewServer	(Required) Hostname of the DSView software server to which the request is directed.
dsviewPort	(Optional) Port number to which the request is directed; the default is 8080.
namedPage	(Required) Full pathname of the requested named screen. Valid values are: • /dsview/avct/core/ createSession.page • /dsview/avct/core/ destroySession • /dsview/avct/browser/ launchSession
authentication_ parameters	(Optional) Credentials or session ID to be used for authentication and authorization.
target_ parameters	(Optional) Name or address of the target device.

2.3 Authentication Parameters

All requests to the GUI Access API require authentication. Authentication information for a named screen can be specified in the following ways:

- Passing user credentials (username and password) with the request
- Passing a user session ID with the request
- Using the session ID specified in a cookie

2.3.1 HTTP request with user credentials

The following user credentials can be used as parameters for the request:

- _authUsername - Username for authentication and authorization of the named screen.
- _authPassword - Password for authentication and authorization of the named screen.
- _authZonename - Zone name for authentication and authorization of the named screen. The user must belong to this zone. This parameter is optional; if not provided, the top level zone is used by default.

These parameters are evaluated. If they are valid, a user session is created. The HTTP response returns the user session ID in the HTTP response header and indicates a cookie should be created with the user session ID.

If the named screen encounters any error condition after authentication, the HTTP response does not provide a user session ID.

URL format

An example URL with user credentials is as follows.

https://[dsviewServer]:[dsviewPort]/[namedPage]?_authUsername=user&_authPassword=pwd

An example URL with user credentials and zones is as follows.

https://[dsviewServer]:[dsviewPort]/[namedPage]?_authUsername=user&_authPassword=pwd&authZonename=zone

2.3.2 HTTP request without user credentials

As an alternative to providing credentials, the HTTP request can be redirected to the DSView software login screen and prompted for credentials. The _authNeeded parameter redirects the request to the DSView software login screen before proceeding to the named screen.

If valid credentials are entered in the login screen, a user session is created. The HTTP response returns the user session ID in the HTTP response header and indicates a cookie is being created with the user session ID. The DSView software navigates to the named screen. Any value can be passed with this parameter.

URL format

An example URL without user credentials is as follows.

https://[dsviewServer]:[dsviewPort]/[namedPage]?_authNeeded=1

2.3.3 User session ID

All named screens provide support for passing a user session ID as a parameter.

The _authSession parameter identifies the user session to be used for authentication and authorization of the named screen.

The parameter is evaluated. If it identifies an active session, the named screen continues processing. The HTTP response returns the user session ID in the response header and indicates a cookie is being created with the user session ID.

If the named screen encounters an error condition after session ID verification, the HTTP response does not include the session ID.

URL format

An example URL with user session ID is as follows.

https://[dsviewServer]:[dsviewPort]/[namedPage]?_authSession=48BFACCB16986926

2.3.4 Cookies

Cookies can be used to retrieve user session information.

The DSView software server first checks if credentials (username and password) or a session ID are provided with a named screen request. If they are not, the server checks for a cookie and uses it, if found.

If the user session information from the cookie identifies an active session, the named screen continues processing. The HTTP response returns the user session ID as an attribute in the response header.

If the named screen encounters an error condition after session ID verification, the HTTP response does not include the session ID.

2.3.5 Specifying multiple methods

When user credentials and a user session ID are specified in a named screen request, the user session ID is checked to see if it is still active and for the specified user credentials. If so, the named screen continues processing using the user session ID. If not, a new user session is created using the supplied user credentials.

2.4 HTTP Response Header

After processing each named screen request, the DSView™ software server returns an HTTP response that provides information about the results.

Only a few of the requests have responses that must be programmatically evaluated to determine the results of the request. Many requests expect that the HTTP response is directed to a browser for presentation. If the response is directed to a browser, no additional browser settings are required and any errors are presented to the user.

Attributes returned in an HTTP response header

The following attributes are returned in the HTTP response header.

Table 2.2 HTTP Response Header Attributes

ATTRIBUTE	DESCRIPTION
dsview-error	This attribute is always returned and it indicates the error status of the request. The following numbers define specific errors: • 0 - OK (Success) • 1 - Bad Credentials • 2 - System Error • 3 - License Limit Reached • 4 - User Session Expired • 5 - User Account Disabled/Locked • 6 - Prompting Required (see note) • 100 - Operation Requires Authentication • 101 - Not Authorized • 102 - Unit Requested Not Found • 103 - Unit Requested Cannot Perform Action Requested
dsview-session	This attribute is only returned if the dsview-error attribute value is 0.
User session ID	This attribute indicates the user session ID which you can reuse in subsequent requests to avoid using up all user sessions granted by your DSView software license. Always use the user session ID returned by the most recent request.

NOTE: The GUI Access API does not provide the ability to process multi-step authentication. When the user being authenticated belongs to an external authentication service such as RSA SecurID, and the external authentication service requires additional prompting in order to authenticate the user, such as when an RSA user account is in New Pin Mode or Next Token Mode, authentication fails for that user with a dsview-error status 6. If this situation occurs, the user needs to login to the DSView software through the login screen from a DSView software client and complete the necessary steps to disable the mode that requires additional prompting for the user account.

2.5 Creating a User Session

The Create Session screen is used to create a user session and/or verify an existing user session. The following list describes the screen's actions when a user session ID and/or user credentials are specified:

- If a user session ID is specified and no user credentials are specified, the screen verifies that the user session is still active.
- If a user session ID and user credentials are specified, the screen verifies that the user session is still active and matches the passed credentials. If not, it attempts to create a new user session.
- If only user credentials are specified, the screen attempts to create a new user session.
- If a user session ID is returned, the session inactivity timeout is renewed, which allows the caller of this screen to use this user session ID in a subsequent screen request.

NOTE: Generally, this screen should be called before any of the other named screens. This verifies the current user session is active (creating a new one if necessary) and resets the inactivity timer, thereby reducing the chance that a subsequent named screen request fails due to user session expiration.

2.5.1 URL format

An example Create Session screen URL format is as follows.

http[s]://dsviewServer[:dsviewPort]/dsview/avct/core/createSession.page?[authentication_parameters]

2.5.2 Parameters

The following table lists the parameters that comprise the URL.

Table 2.3 Create Session Screen Parameters

PARAMETER	DESCRIPTION
HTTP or HTTPS	(Required) When debugging and testing with the SDE, you can specify HTTP; however, when running the Enterprise DSView software, you must specify HTTPS.
dsviewServer	(Required) Hostname of the DSView software server to which the request is directed.
dsviewPort	(Optional) Port number to which the request is directed; the default is 8080.
authentication_ parameters	(Optional) User session ID or credentials.
target_ parameters	(Optional) Name or address of the target device.

2.5.3 Return

The HTTP response header contains information that can be processed programmatically, allowing the receiver to get/validate a user session for subsequent named screen requests. If the response is directed to a browser, no additional browser settings are required.

When a URL contains a named screen request to create a user session, it supplies a user session ID and credentials. The user session is verified to see if it is still active and matches the passed credentials. If not, a new user session is created.

An example URL containing a named screen request is as follows.

https://localhost:8080/dsview/avct/core/createSession.page? _authSession=48BFACC&_
authUsername=myUser&_authPassword=myPassword
&authZonename=myZone

2.6 Destroying a User Session

The Destroy Session screen is used to terminate a user session. When a user session is terminated, one licensed user session is returned to the pool of available sessions. If a user session ID is not specified, the session ID stored in a cookie is used, if available.

2.6.1 URL format

An example of a Destroy Session screen URL is as follows.

http[s]://dsviewServer[:dsviewPort]/dsview/avct/core/destroySession.page?[authentication_parameters]

2.6.2 Parameters

The following table lists the parameters that comprise the URL.

Table 2.4 Destroy Session Screen Parameters

PARAMETER	DESCRIPTION
HTTP or HTTPS	(Required) When debugging and testing with the SDE, you can specify HTTP; however, when running the Enterprise DSView software, you must specify HTTPS.
dsviewServer	(Required) Hostname of the DSView software server to which the request is directed.
dsviewPort	(Optional) Port number to which the request is directed; the default is 8080.
authentication_parameters	(Required) User session ID or credentials.

2.6.3 Return

The HTTP response header contains information that can be processed programmatically, allowing the receiver to review the results. If the response is directed to a browser, no additional browser settings are required.

An example of a URL that contains a named screen request to terminate a session is as follows.

https://localhost:8080/dsview/avct/core/destroySession.page?_authSession=1c9si8eud3h4d

NOTE: The session ID is 1c9si8eud3h4d.

2.7 Launching a Session

The Launch Session screen is used to launch a DSView software session to a viewer or a screen within the DSView™ software. You must launch a user session with appropriate access rights. The Establish Viewer Session access right is required when launching a software session to a KVM, serial or serial-over-LAN viewer. The view right is required for all other sessions.

2.7.1 Target parameter

The target parameter can be specified as the name (targetName) or address (targetAddress) of the unit. The target name is used to match the unit name field in the DSView software database and the target address is used to match the unit address field.

A search is conducted for a simple match. If a match is not found, a search is conducted based on the address returned on a DNS lookup with the target address. If a match is not found, a search is conducted based on the name returned on a DNS lookup with the target address. If both a name and address are specified, the target name is used first. If no match is found, the target address is used.

2.7.2 URL format

An example Launch Session screen URL format is as follows.

http[s]://**dsviewServer**[:**dsviewPort**] /dsview/avct/core/launchSession.page?**session_type& target**& [**authentication_parameters**]

2.7.3 Parameters

The following table lists the parameters that comprise the URL.

Table 2.5 Launch Session Screen Parameters

PARAMETER	DESCRIPTION
HTTP or HTTPS	(Required) When debugging and testing with the SDE, you can specify **http**; however, when running the Enterprise DSView software, you must specify **https**.
dsviewServer	(Required) Hostname of the DSView software server to which the request is directed.
dsviewPort	(Optional) Port number to which the request is directed; the default is 8080.
sessionType	Indicates the session type; see the GUI Access API Session Types table in Launching a Session on page 13 , which also lists the optimal browser settings for width, height and whether the toolbar should be displayed.
targetName	(Required) Name (targetName=) or address (targetAddress=) of the unit.
authentication_ parameters	(Optional) User session ID or credentials to be used for authentication and authorization. See Authentication Parameters on page 9.

Table 2.6 GUI Access API Session Types

NAME	DESCRIPTION	BROWSER		
		WIDTH	HEIGHT	TOOLBAR
coreMain	Displays the primary DSView software interface. No target parameter is required.	Default	Default	Default
coreOverview	Launches the Unit Overview screen for the specified target.	Default	Default	Default
avctKVM	Launches a KVM session to the specified target.	400	300	False
avctSerial	Launches a serial session to the specified target.	400	300	False
avctTelnet	Launches a Telnet session to the specified target.	400	300	False
avctBrowser	Launches a browser session to the specified target.	Default	Default	Default
avctSerialOverLan	Launches a Serial over LAN session to the specified target.	400	300	False
avctTerminalServices	Launches a Terminal Services session to the specified target.	400	300	False
avctVNC	Launches a VNC session to the specified target.	400	300	False

2.7.4 Return

The HTTP response is expected to be directed to a browser. If any errors are encountered, an error screen is returned in HTML format. See HTTP Response Header on page 11 for more information.

An example of a URL named screen request to launch a KVM session to the unit named myTarget is as follows.

https://localhost:8080/dsview/avct/core/launchSession.page?sessionType=avctKVM&targetName=myTarget

2.8 GUI Access API Sample Applications

The term sample application means a complete application, including any project file. Each sample application demonstrates the GUI Access API using a programming language and demonstrates creating named screen requests to the DSView™ software server.

Sample applications are included with the SDK in the samples directory.

Samples are provided for the following languages:

- HTML/JavaScript
- Microsoft Visual C++ development system (only on Microsoft Windows systems)
- Microsoft Visual Basic development system (only on Microsoft Windows systems)

Most programming languages can send named screen requests to the DSView™ software server. Slight variations can exist between one implementation of the GUI Access API sample application and another, due to differences in the programming environment.

The GUI Access API sample applications provide limited data input validation. When validation detects errors, the sample applications usually display a diagnostic error message and, if appropriate, prompt again for input. When validation does not detect errors, even when they are present, you can see how the DSView software server responds and then make adjustments to your applications.

Figure 2.1 GUI Access API Sample Application Screen

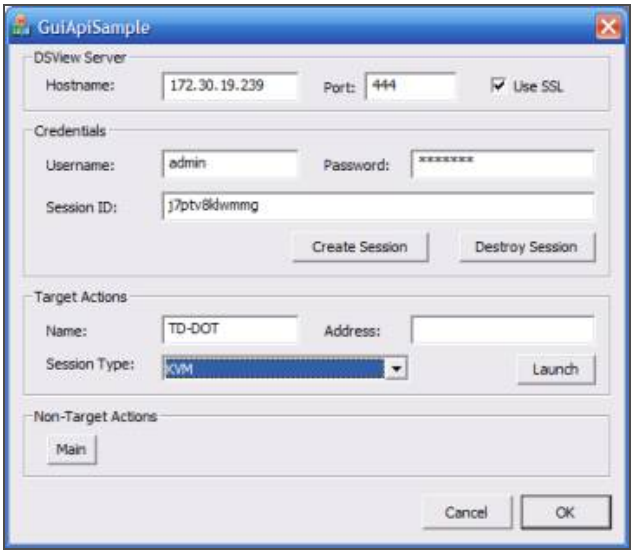

Table 2.7 Fields in GUI Access API Sample Application Screens

FIELD NAME	DESCRIPTION
Hostname	Hostname of the DSView software; the default is localhost.
Port	Port number; the default is 8080.
Use SSL	If checked, Secure Socket Layer (SSL or HTTPS), not HTTP, is used when communicating with the DSView software server.
Username	Username credential that is used for named screen requests.
Password	Password credential that is used for named screen requests.
Session ID	Session ID that is used for named screen requests. Initially, this field is blank. This field is not present in the HTML/JavaScript sample because it is not needed. HTML-based applications use standard web application methods (cookies) for storing session information. Only applications interfacing with the DSView software server need to store the session ID for later use.
Name	Target device name.
Address	Target device address.
Session Type	Type of target session. This drop-down list contains all available session types. See the GUI Access API Session Types table in Launching a Session on page 13 for session type descriptions. Launching the main screen does not require a target device; therefore, it is initiated with the Non-Target Actions - Main button instead of a Session Type selection.

2.8.1 Application screen buttons

The following table describes the buttons in the GUI Access API sample application screens.

For each action that is initiated by clicking a button:

- The sample application validates the fields. If any fields are invalid, an error message is displayed.
- The sample application sends a named screen (HTTP) request to the DSView software server.
- The DSView software server evaluates the request, attempts to perform the operation and then returns the results in an HTTP response.
- The sample application evaluates the HTTP response. If the response indicates a failure, the sample application displays an error message.
- If the response indicates a failure, the application displays an error message.

Table 2.8 Buttons in GUI Access API Sample Screens

NAME	ACTION
Create Session	Initiates creation of a user session. If the operation is successful, the newly created session ID is displayed in the Session ID field and in a message. This button is not present in the HTML/JavaScript sample application because it is not needed. HTML-based applications use standard web application methods (cookies) for storing session information. Only applications interfacing with the DSView software server need to store the session ID for later use.
Destroy Session	Initiates termination of the session specified in the Session ID field. If the operation is successful, the Session ID field is cleared and a success message is displayed.
Launch	Initiates launching of the session type selected in the Session Type field. If the operation is successful, the sample application forwards the HTTP response to a viewer (browser) or a screen within the DSView software. If the named screen launches a viewer, the browser can initially display a splash screen, then update the screen with the viewer session screen or an error message.
Non-Target Actions - Main	Launches the main screen of the DSView software using the GUI Access API.

2.8.2 Using HTML/JavaScript

The following procedure provides steps to run an HTML or JavaScript sample application.

To run the HTML/JavaScript sample application:

1. Run the SDE or the Enterprise DSView™ software.
2. Run a web browser.
3. From the File menu, click *Open* or *Open File*.
4. Enter the following applicable sample application filename:
   - For Microsoft Windows, enter: avocent\dssdk\samples\GUIAccess\JavaScript\GuiApiSample.html
   - For Linux, enter: avocent/dssdk/samples/GUIAccess/JavaScriptGuiApiSample.html
5. Click *OK*.

2.8.3 Using the Microsoft Visual C++ development system

The following procedure provides steps to run a Microsoft Visual C++ sample application.

To run the Microsoft Visual C++ sample application:

1. Run the SDE or the Enterprise DSView software.
2. In the Microsoft Visual Studio® .NET 2003 development system, open the *GuiApiSample.vcproj* project file.
3. From the Build menu, click *Build Solution*.
4. From the Debug menu, click *Start*.

2.8.4 Using the Microsoft Visual Basic development system

The following procedure provides steps to run a Microsoft Visual Basic sample application.

To run a Microsoft Visual Basic sample application:

1. Run the SDE or the Enterprise DSView software.

2. In Microsoft Visual Studio .NET 2003 development system, open the *GuiApiSample.vbproj* project file.
3. From the Build menu, click *Build Solution*.
4. From the Debug menu, click *Start*.

3 WEB SERVICES API

The Web Services API allows applications to interact with other applications by using open standards such as Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP). This interaction enables your applications, tools and systems to manage information in the DSView™ or Rack Power Manager software and provides support for accessing information, performing some unit control and adding events to the event log.

3.1 Syntax Conventions

This documentation uses the following syntax conventions:

- Bold type indicates user-supplied values.
- Square brackets ([ ]) enclose optional values.

3.2 Web Operating Environment

3.2.1 Protocols

The web operating environment uses the following protocols:

- SOAP version 1.1 - messaging protocol
- HTTPS Internet protocol - used in a production environment with the Enterprise DSView software
- HTTP or HTTPS Internet protocol - used in a development environment with the SDE

3.2.2 Applications

Your applications can reside on the same system as the DSView software server or on another server. Supported languages include, but are not limited to:

- Java Axis 1.2
- C++ using Axis C++ 1.4 (only on Microsoft Windows systems)

3.2.3 Implementation

The implementation of the Web Services API is based on Axis (Apache Axis 1.2), an open-source SOAP server and client.

3.3 Definition Files

EXtensible Markup Language (XML) is a platform-independent markup language that describes data contents, not format. Two XML-based languages are Web Service Description Language and XML Schema Definition Language. The Web Service API includes the following XML files:

- WSDL files define the web services, port types, messages and service bindings
- XSD files define schemas for SOAP fault codes, common headers and messages, components and structures

Web Services API definition files are located in the libs directory.

3.4 SOAP Message Structure

The following table contains the syntax and sequence of the SOAP message (envelope) structure. The same structure applies to requests and responses. In actual applications, italicized text is replaced by application-specific, site-specific or user-specific values.

Table 3.1 SOAP Message Syntax and Sequence

NAME	DESCRIPTION
<soapenv:Envelope>	Opening tag for a standard SOAP envelope.
<soapenv:Header>	Opening tag for a standard SOAP header.
<sessionHeader>	Opening tag for a DSView software session header. Generally, the header on an initial request passes the username and password for user authentication and authorization. In all responses that report success, the header includes the session ID for session reuse on subsequent calls.
<sessionId>*sessionID*</sessionId>	Unique user session ID created by the DSView software. The user session ID, which is used for authentication and authorization, can be: • Created by supplying user credentials (username and password) in a request. • Created by an authenticate request. • Destroyed by a logout request. Each Web Services API session exclusively uses one session license from the pool of available sessions.
<username>*username*</username>	Username component of credentials for a DSView software user account. This is required in all request headers if a valid session ID is not provided. If a username is provided in the request header, it is also returned in the response header.
<password>*password*</password>	Password component of credentials for a DSView software user account. This is required in all request headers if a valid session ID is not provided. If a password is provided in the request header, it is also returned in the response header.
<zonename/>	Name of the zone that the user and service account belong to. The user and service account must belong to the same zone. If no zone name is provided, the top level zone name is used by default.
<servicename/>	Name of the service account.
</sessionHeader>	Closing tag for a DSView software session header.
</soapenv:Header>	Closing tag for a standard SOAP header.
<soapenv:Body>	Opening tag for a standard SOAP body.
<*someMethod*>*Body of request or response* <*someMethodAttributes*></*someMethodAttributes*> </*someMethod*>	In a request, this section contains any parameters or attributes needed to perform the request. Parameter data values are user-supplied. In a response, this section contains any returned data.
</soapenv:Body>	Opening tag for a standard SOAP body.
</soapenv:Envelope>	Closing tag for a standard SOAP envelope.

3.4.1 Example SOAP request

An example of a SOAP request for all unit details for the unit at IP address 192.168.1.1 is as follows.

```
POST /dsview/services/UnitServiceApi?wsdl HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.2
Host: localhost:8081
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: "urn:units:webservices:server:dsview:avocent:com#getUnitDetail"
Content-Length: 1099
<?xml version="1.0" encoding="UTF-8"?>
 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
 <ns1:sessionHeader
 soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
 soapenv:mustUnderstand="0"
 xmlns:ns1="urn:common:webservices:server:dsview:avocent:com">
 <ns1:credentials>
 <ns1:sessionId>3fn5lc16l9s47</ns1:sessionId>
 <ns1:username>admin</ns1:username>
 <ns1:password>password</ns1:password>
 </ns1:credentials>
 </ns1:sessionHeader>
 </soapenv:Header>
 <soapenv:Body>
 <getUnitDetail
 xmlns="urn:schema:units:webservices:server:dsview:avocent:com">
 <getUnitDetailRequest xsi:type="ns2:GetUnitDetailRequestType"
 xmlns:ns2="urn:schema:units:webservices:server:dsview:avocent:com">
 <unitId>
 <ns3:address
 xmlns:ns3="urn:common:webservices:server:dsview:avocent:com">
 192.168.1.1</ns3:address>
 </unitId>
 <showFields>
 <ns4:fields
 xmlns:ns4="urn:common:webservices:server:dsview:avocent:com">
 </ns4:fields>
 </showFields>
 </getUnitDetailRequest>
 </getUnitDetail>
 </soapenv:Body>
 </soapenv:Envelope>
```

3.4.2 Example SOAP response

An example of a SOAP response to a SOAP request for all unit details for the unit at IP address 192.168.1.1 is as follows.

```
HTTP/1.1 200 OK
Date: Wed, 06 Jul 2005 18:41:36 GMT
Server: Jetty/4.2.12 (Windows XP/5.1 x86 java/1.5.0_03)
Set-Cookie: JSESSIONID=tgr8f2752ier;path=/dsview
Content-Type: text/xml; charset=utf-8
<?xml version="1.0" encoding="utf-8"?>
 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
 <ns1:sessionHeader
 soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
 soapenv:mustUnderstand="0"
 xmlns:ns1="urn:common:webservices:server:dsview:avocent:com">
 <ns1:credentials>
 <ns1:sessionId>3fn5lc16l9s47</ns1:sessionId>
 <ns1:username>admin</ns1:username>
 <ns1:password>password</ns1:password>
 </ns1:credentials>
 </ns1:sessionHeader>
 </soapenv:Header>
 <soapenv:Body>
 <getUnitDetailResponse
 xmlns="urn:schema:units:webservices:server:dsview:avocent:com">
 <ns2:version
 xmlns:ns2="urn:common:webservices:server:dsview:avocent:com">1.0
 </ns2:version>
 <return>
 <unitId>9217</unitId>
 <name>target</name>
 <address>192.168.1.1</address>
 </return>
 </getUnitDetailResponse>
 </soapenv:Body>
 </soapenv:Envelope>
```

3.5 Session Management Service

Operations in the Session Management Service allow your application to control the lifespan of user sessions established with the Web Services API.

A session is initiated by supplying user credentials (username, password and optional service name and zone name). However, the session begins only after the DSView™ software server authenticates the user credentials (the Web Services API requires that all requests be authenticated) and grants the session exclusive use of one of a limited number of licenses for user sessions.

The Web Services API does not support multi-step authentication. For example, if you belong to the RSA SecurID external authentication service and New Pin Mode is configured, additional prompting at the login is required and the authentication from the Web Services API fails. You must log in to the DSView software and change the account settings so that multi-step authentication is not required.

The DSView software server then includes your credentials and a newly-created session ID in its response to the application. To conserve licenses, your application can reuse this session ID in subsequent requests. If your application attempts to reuse a session ID after the session has timed out, the DSView software server creates a new session with the passed credentials and then processes the request.

NOTE: After a user is authenticated, the DSView software server authorizes information access based on configured access rights.

The session ends when either of the following conditions occur:

- The idle time for the session equals the session timeout value configured in the DSView™ software. This is the default behavior.
- Your application explicitly terminates the session (logs out) when the session is no longer needed. This is the better practice.

3.5.1 Service accounts

In the DSView software, you can configure user accounts or service accounts that never expire. This is useful for applications and systems that use a fixed or dedicated account to access the DSView software using the Web Services API.

Both the username and servicename must represent valid users in the DSView software. The service account must be an internal user account while the impersonated user can be either an internal or external account.

A service account cannot be used to log in as a regular user through either the Web Services API or the DSView software. The username and password fields of the SessionHeader for the request must not contain the credentials for a service account. Certificate log-in is not supported.

To log in as a service account and impersonate another user:

Enter the following values in the SessionHeader:

- servicename field - the name of a service account
- password field - the password for the service account
- username field - the name of the user
- zone name field - the name of the zone that the user and service account belong to. The user and service account must belong to the same zone. If no zone name is provided, the top level zone name is used by default.

The session rights are the intersection of the rights assigned to the service account and the user account. The DSView software logs the session activity for the user.

NOTE: The GUI Access API does not support impersonated user sessions. However, you can use impersonated user sessions with the GUI Access API if a session is first established with the Web Services API. Use the session ID returned from the impersonated user session to make requests to the GUI Access API.

To log in as a service account without impersonating another user:

Enter the following values in the SessionHeader:

- servicename field - the name of a service account
- password field - the password for the service account

Do not include the username field in the request. The session rights are the rights assigned to the service account.

To log in as a regular user not using a service account:

Enter the following values in the SessionHeader:

- password field - the password for the user account
- username field - the name of a non-service account
- zone name field - the name of the zone that the user belongs to. If no zone name is provided, the top level zone name is used by default.

Do not include the servicename field in the request. The session rights are the rights assigned to the user.

3.5.2 Definition file

The filename for the DSView™ software session management service definition file is dsviewService-SessionMgmt.wsdl

Visit https://dsviewServer[:dsviewPort]/dsview/services/ EventServiceApi?wsdl to retrieve the WSDL dynamically generated for you.

3.5.3 Operations

The following table describes Session Management Service operations.

Table 3.2 Web Services API Session Management Service Operations

NAME	DESCRIPTION
getVersion	Requests the current version of the Session Management Service from the DSView software server. The format of the returned string is: majorVersion.minorVersion. A change in minorVersion indicates the service is modified but maintains backward compatibility with existing client implementations at the same majorVersion level. A change in majorVersion indicates the service is modified and cannot maintain backward compatibility.
authenticate	Requests the DSView software server to authenticate the user credentials (username and password) or session ID provided in the session header of a SOAP envelope and if successful: • Log in the user associated with the request. • Grant exclusive use of one user session license. • Start the session. Respond to the request with the user credentials and a newly created session ID in the sessionHeader. The DSView software server compares the provided credentials with user accounts in its internal database. This operation is redundant and provided for completeness. The DSView software server validates user credentials and session IDs provided in the session header of all requests. Generally, if you explicitly request authentication, it should be done before any other Web Services API requests.
logout	Requests the DSView software server: • Log out the current user associated with the request. • Relinquish the user session license. • Terminate the active user session for the Web Services API. • Retire the session ID from further use.

3.6 Event Service

Operations in the Event Service allow your application to add events to the DSView software event log.

3.6.1 Definition file

The filename for the DSView software event service definition file is dsviewService-Event.wsdl.

Visit http[s]://dsviewServer[:dsviewPort]/dsview/services/UnitServiceApi?wsdl to retrieve the WSDL dynamically generated for you.

3.6.2 Operations

The following table describes the Event Service operations.

Table 3.3 Web Services API Event Service Operations

NAME	DESCRIPTION
getVersion	Requests the current version of the Event Service from the DSView software server. The format of the returned string is: majorVersion.minorVersion. A change in minorVersion indicates the service is modified but maintains backward compatibility with existing client implementations at the same majorVersion level. A change in majorVersion indicates the service is modified and cannot maintain backward compatibility. The following lists the DSView software version and the corresponding API returns: • DSView 4 returns API version 1.1 • DSView 4.5.0.247 returns API version 1.2 • DSView 4.5 SP5 returns API version 2.0
addEvent	Requests the DSView software server log an event in its event log. The event is based on the information provided in the body of a SOAP envelope. This supplements event-related functionality in the DSView software.
getEvent	Requests the event details for Access Control, Appliance, Authentication, Data Logging, External, IPMI, Modem, Session, SSH Passthrough, Status, System, Tasks, Units, Users and Virtualization. The input parameters are: • Severities (Mandatory) - When entering multiple severities, separate them with a comma. • Categories (Optional) - When entering multiple categories, separate them with a comma. • dateRange (Optional) - Enter the date and time in UTC format. • State (Optional) - Enter either **New** or **Acknowledged**. • unitOnly (Optional) - Enter either **True** or **False**.
DoAcknowledgeEvent	Sets the event status to acknowledged. The input parameters are: • EventOid (Mandatory) - When entering multiple OIDs, separate them with a comma.
SetDCPEventSettings	Updates the event settings for the DCP server information. The input parameters are: • trapEnabled (Mandatory) - Enter **true** if it is enabled and **false** if it is disabled. If you enter false, the DSView and Rack Power Mount software does not enable the trap function and does not push event data to the DCP. • trapInterval (Mandatory) - Enter a minute value between one and 60. • Severities (Mandatory) - When entering multiple severities, separate them with a comma. • Categories (Optional) - When entering multiple categories, separate them with a comma. • Hostname: Enter or update the DCP IP address. • apiPort (Mandatory) - Enter the DCP web service port number which is always fixed at 8443. • authPort (Mandatory) - Enter the AMP authentication port number which is always fixed at 8092. • Account (Mandatory) - Enter the account name for the DCP user who has administrative rights. • Password (Mandatory) - Enter the password for the account. • apiVersion (Mandatory) - Enter the version number of the DCP integration web service APIs which is fixed at 1.

3.7 User Service

Operations in the User Service allow a third-party client to manage DSView™ software users.

3.7.1 Definition file

The filename for the DSView software user service definition file is dsviewService-User-wsdl.

Visit http[s]://dsviewServer[:dsviewPort]/dsview/services/UserServiceApi?wsdl to retrieve the WSDL dynamically generated for you.

3.7.2 Operations

The following table describes the User Service operations. The operations addUserToUserGroup, removeUserFromUserGroup and removeUserFromAllUserGroup return one of the following integers:

- 1 = Operation successful.
- 2 = Operation failed; user provided in request was invalid.
- 3 = Operation failed; user group data provided in request was invalid.
- 4 = Operation failed; logged in user did not have adequate rights.
- 5 = Operation failed; the group to be removed contained the last DSView software user administrator in the system.
- 6 = Invalid unit name/OID.
- 7 = No such unit from the database.
- 8 = Invalid right.
- 9 = Invalid unit group name/OID.
- 10 = Invalid authentication or preemption level.
- 11 = Username already exists.

NAME	DESCRIPTION
getVersion	Determines the current version of the User Service.
addUserToUserGroups	Adds a user in the DSView software to internal user group(s).
removeUserFromUserGroups	Removes a user in the DSView software from internal user group(s).
removeUserFromAllUserGroups	Removes a user in the DSView software from all internal user groups this user is assigned to.
getUserGroupsAssignedToUser	Determines which internal user groups a user in the DSView software is assigned to.

addLocalUser	Creates a new local user in the DSView software.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• UserName - Username to be added into the DSView software
	• UserPass - Password
	• authServerName - Authorization server in which the user is added. The authServerName value for DSView is 'DSView Internal'.
	• User Group Name (optional) – The Group Name in which the user must be added. The Group ID for built-in user groups are:
	  • 1 = DSView software administrators
	  • 2 = User administrators
	  • 3 = Appliance administrators
	  • 4 = Auditors
	  • 5 = Users
	• User Group Type (optional)– Group Type is the parent node where group name resides and includes the following:
	  • User Group Type 1 - To add user under built-in group
	  • User Group Type 2 - To add user under user defined group

deleteLocalUser	Deletes a local user from the DSView software.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• Username - The username to be deleted from the DSView software

modifyLocalUserDetail	Modifies local user details such as preemption level, password, address, email addresses, contact numbers and notes.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• Preemption Level - The preemption level of the user which ranges from 1 to 4
	• Password - User password
	• Home Address - User's home address
	• Business Address - User's business address
	• Business Phone - User's business phone
	• Home Phone - User's home phone

NAME	DESCRIPTION
	• Mobile Phone - User's mobile phone • Mobile Business Phone - User's mobile business phone • Email Addresses - Primary and additional email addresses • Notes - Notes about the local user

queryAllLocalUsers	Finds all internal users from the DSView software.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• Search string (optional) – Keyword to search for all the users that contain this search string

addRightsToUser	Adds rights to a local user on specific units/unit groups.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• Username - Name of the user that rights are being set for
	• List of rights - The list of available rights
	• Right Name - Right to be set. The codes for the rights include:
	  • 0 = View unit information
	  • 1 = Reboot appliance and disconnect sessions
	  • 2 = Flash upgrade appliance
	  • 3 = Configure unit settings
	  • 4 = Configure appliance local user accounts
	  • 5 = Establish viewer sessions
	  • 6 = Control target device power
	  • 7 = Establish virtual media sessions
	  • 8 = Establish reserved virtual media sessions
	  • 9 = View data logging
	• List of Units - The list of available units to assign user rights to and the following information for each unit:
	  • Unit ID - Unit ID
	  • Unit Name (optional) – Unit Name
	• List of Unit Groups - The list of available user groups to assign user rights and the following information for each group:
	  • Unit Group Id - Group id
	  • Unit Group Name (optional) – Group Name

deleteRightsFromUser	Deletes rights of a local user on specific units/unit groups.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• Username - The name of the user that rights are being removed from
	• List of rights - The list of available rights to be removed

NAME	DESCRIPTION

	• Right Name - Right to be set. The codes for the rights include:
	  • 0 = View unit information
	  • 1 = Reboot appliance and disconnect sessions
	  • 2 = Flash upgrade appliance
	  • 3 = Configure unit settings
	  • 4 = Configure appliance local user accounts
	  • 5 = Establish viewer sessions
	  • 6 = Control target device power
	  • 7 = Establish virtual media sessions
	  • 8 = Establish reserved virtual media sessions
	  • 9 = View data logging
	• List of Units - The list of available units to remove user rights from and the following information for each unit:
	  • Unit Id - Unit id
	  • Unit Name (optional) – Unit Name
	• List of Unit Groups - The list of available unit groups to remove rights from and the following information for each group:
	  • Unit Group Id - Group id
	  • Unit Group Name (optional) – Group Name

modifyRightsToUsers	Modifies (adds/deletes) the rights of a local user on specific units.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• Username - The name of the user with rights being modified
	• Right Name - Right to be set. The codes for the rights include:
	  • 0 = View unit information
	  • 1 = Reboot appliance and disconnect sessions
	  • 2 = Flash upgrade appliance
	  • 3 = Configure unit settings
	  • 4 = Configure appliance local user accounts
	  • 5 = Establish viewer sessions
	  • 6 = Control target device power
	  • 7 = Establish virtual media sessions
	  • 8 = Establish reserved virtual media sessions
	  • 9 = View data logging
	• List of rights to be modified and the following information:
	  • Right Name - Right to be set.
	• List of Units - The available units to modify user rights and the following information for each unit:
	  • Unit ID - Unit ID
	  • Unit Name (optional) – Unit name
	• List of Unit Groups - The available unit groups to modify user rights

queryRightsToUser	Lists the rights of a local user on specific units/Units Groups. The input parameters are:

NAME	DESCRIPTION

	• Administrative credentials which includes the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• Username - Username being queried
	• Unit Name - Unit name being queried

getUserAndAuthServers	Returns all of the User Oids (useroid), User Names (username), Authentication Service Oids (authServiceOid) and Authentication Service Names (authServerName) of every user (excluding virtual users) within the DSView 4 database.
	The input parameters are:
	• Administrator credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password

getUserGroupAndAuthServers	Returns all of the User Group Oids (userOid), User Group Names (userName), Authentication Server Oids (authServerOid) and Authentication Server Names (authServerName) within the DSView 4 database.
	The input parameters are:
	• Administrator credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password

getUserEffectiveRights	Returns all the effective rights on each unit for the specified user/user group.
	The input parameters are:
	• Administrator credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• Target User or User Group including the following:
	  • Authentication Server Name

3.8 LDAP User Service

Operations in the LDAP User Service allow a third-party client to manage external LDAP users.

3.8.1 Definition file

The filename for the DSView™ software LDAP user service definition file is dsviewService-LdapUser.wsdl.

Visit https://dsviewServer[:dsviewPort]/dsview/services/LdapUserServiceApi?wsdl to retrieve the WSDL dynamically generated.

3.8.2 Operations

One of the following operations from the LDAP user service is returned:

- 1 = Operation success
- 2 = Invalid username
- 3 = Invalid user group
- 4 = Insufficient rights for operation
- 5 = Admin only

- 6 = Invalid unit name/OID
- 7 = No such unit
- 8 = Invalid right
- 9 = Invalid unit group name/OID
- 10 = Invalid authentication or preemption level
- 11 = Username already exists

NAME	DESCRIPTION

getExternalAuthServices	Lists all the external LDAP authentication services.
	The input parameters are:
	• Administrative credentials Username including the following:
	  • DSView software admin username password
	  • DSView software admin user password

getExternalLDAPUsers	Lists all the external LDAP users belonging to one LDAP server.
	The input parameters are:
	• Administrative credentials Username including the following:
	  • DSView software admin username password
	  • DSView software admin user password
	• LDAP Server - LDAP server name

addLDAPUsers	Adds an LDAP user into DSView management software.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• Username - Username to be added into the DSView software
	• LDAP Server - LDAP Server name in which the user is added
	• User Group Name - The group name in which the user must be added
	• User Group Type (optional) – Group type is a parent node where group name resides and includes the following:
	  • User Group Type 1 - To add a user under a built-in group
	  • User Group Type 2 - To add a user under a user-defined group
	• Preemption Level - The preemption level of the user which ranges from 1 to 4

deleteLDAPUsers	Deletes LDAP users from DSView management software.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• LDAP Server - LDAP server name in which the user is to be deleted
	• Username - Username to be deleted into the DSView software

modifyLDAPUsers	Modifies the DSView software group of an LDAP user in the DSView software.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• LDAP Server - LDAP server name
	• Username - Username to be modified in the DSView software
	• New Name - The new name for the LDAP user
	• Full Name - The full name of the LDAP user

getLDAPUsers	Lists all the LDAP users added in the DSView software.

NAME	DESCRIPTION

	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• LDAP Server - LDAP server name
	• Search String - Keyword search for all the users that contain this search string
	• Group Name - Group name
	• Group Type - Group type

addRightsToLDAPUser	Sets rights to a user in units/unit groups.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• LDAP Server - LDAP server name
	• Username - LDAP username that rights are set for
	• List of Units - Units on which the rights are set
	• List of Unit Groups - Unit groups that rights are set for

removeRightsFromLDAPUser	Removes user rights in units/unit groups.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• LDAP Server - LDAP server name
	• Username - LDAP username to remove rights
	• List of Units –The available units with user rights to be removed
	• List of Unit Groups - The available unit groups with user rights to be removed

NAME	DESCRIPTION

modifyRightsForLDAPUser	Modifies rights of a user in units.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• LDAP Server - LDAP server name
	• Username - LDAP username to modify rights
	• List of rights to be added
	• List of rights to be removed
	• List of Units - Units with user rights to be modified
	• List of Unit Groups - Unit Groups with user rights to be modified

getRightsFromLDAPUser	Lists all the rights of a LDAP user in units.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• LDAP Server - LDAP server name
	• Username - LDAP username to view rights
	• List of Units - Units to review the user rights
	• List of Unit Groups - Unit groups to review the user rights

3.9 Unit Group Service

This web service provides operations to third-party clients to manage unit groups in DSView™ software.

3.9.1 Definition file

The filename for the DSView software unit group service definition file is dsviewService-UnitGroup.wsdl.

Visit https://dsviewServer[:dsviewPort]/dsview/services/UnitGroupServiceApi?wsdl to retrieve the WSDL dynamically generated for you.

3.9.2 Operations

The Unit Group service supports the following operations:

getlUnitGroupList

Gets the detailed information for all unit groups in DSView, including root level unit groups

Input Parameters:

- None.

Returns:

- A list of UnitGroupDetail objects which contain the details of each unit group. Refer to the UnitGroupDetail data type.

Error Messages:

- Insufficient Access Rights – The user does not have the rights to view unit groups.
- System Error – There is an error accessing the database.

getUnitGroup

Gets information for a unit group in DSView

Input Parameters:

- unitGroupId: The unit group ID of the unit group. This field cannot be empty or invalid.

Returns:

- A UnitGroupDetail object which contains the details of the unit group. Refer to UnitGroupDetail data type.

Error Messages:

- Invalid Unit Group ID – The unit group ID is empty or invalid.
- Insufficient Access Rights – The user does not have the rights to view unit groups.
- System Error – There is an error accessing the database.

addUnitGroup

Adds a unit group to DSView. Duplicate unit groups are not allowed to be added to the database

Input Parameters:

- unitGroupDetail: The details of the unit group to add. Refer to the UnitGroupDetail data type. You must specify the following fields:
  - groupName: The name of the unit group to be added. This field cannot be empty.
  - parentGroupId : The Id of the parent unit group under which the unit group will be added. This field cannot be empty or invalid.

Returns:

- A UnitGroupDetail object which contains the details of the unit group.These details include the Id for the new unit group and the type of unit group. Refer to the UnitGroupDetail data type. An event will be logged in the DSView Event Log.

Error Messages:

- Invalid Parent Unit Group ID – The parent unit group ID is empty or invalid.
- Invalid Unit Group Name – The unit group name is empty.
- Duplicate Unit Group – The unit group already exists in the database.
- Insufficient Access Rights – The user does not have the rights to edit unit groups.
- System Error – There is an error accessing the database

addUnitsToUnitGroup

Adds one or more units to an existing unit group in DSView. Any duplicate units will be ignored and not added to the database

Input Parameters:

- unitGroupId: The unit group ID of the unit group. This field cannot be empty or invalid.
- listUnitDetail: A list of UnitGroupDetail objects which contains the details of each unit to be added to the unit group. If there is at least one unit Id that is empty or invalid, there will be an error to allow the user to include the correct unit Id and send the request again. If a unit already exists in the unit group, the unit will not be added to the unit group. Refer to the UnitDetail data type.

Returns:

- A value of 1 if the operation is successful. An event will be logged in the DSView Event Log.

Error Messages:

- Invalid Unit Group ID – The unit group ID is empty or invalid.
- Invalid Unit Id – A unit ID is empty or invalid.
- Insufficient Access Rights – The user does not have the rights to edit unit groups.
- System Error – There is an error accessing the database.

getUnitsFromUnitGroup

Gets the list of units that are members of an existing unit group in DSView

Input Parameters:

- unitGroupId: The unit group ID of the unit group. This field cannot be empty or invalid. Refer to the UnitGroupDetail data type.

Returns:

- A list of UnitGroupDetail objects which contains the details of each unit that is a member of the unit group. Refer to the Unit Detail data type.

Error Messages:

- Invalid Unit Group ID – The unit group ID is empty or invalid.
- Insufficient Access Rights – The user does not have the rights to view unit groups.
- System Error – There is an error accessing the database.

deleteUnitsfromUnitGroup

Deletes one or more units from an existing unit group in DSView

Input Parameters:

- unitGroupId: The unit group ID of the unit group. This field cannot be empty or invalid.
- listUnitDetail: A list of UnitGroupDetail objects which contains the details of each unit to be deleted from the unit group. Refer to the UnitDetail data type. You must specify the following fields in the UnitDetail data type.
- unitId: The unit Id of the unit to be deleted from unit group. If the unit Id is empty or invalid, the unit will not be removed from the unit group.

Returns:

- A value of 1 if the operation is successful. An event will be logged in the DSView Event Log.

Error Messages:

- Invalid Unit Group ID – The unit group ID is empty or invalid.
- Invalid Unit Id – A unit ID in the list of units to be deleted is empty or invalid.
- Invalid Unit Group To Delete – The Global Root, Personal Root and Unassigned unit groups are not allowed to be deleted.
- Insufficient Access Rights – The user does not have the rights to edit unit groups.
- System Error – There is an error accessing the database.

renameUnitGroup

Renames a unit group in DSView

Input Parameters:

- unitGroupDetail: The details of the unit group to rename. Refer to the UnitGroupDetail data type. You must specify the following fields:
- groupId: The unit group Id. This field cannot be empty or invalid. Refer to the UnitGroupDetail data type.
- groupName: The new unit group name. This field cannot be empty.

Returns:

- A UnitGroupDetail object which contains the details of the renamed unit group. Refer to the UnitGroupDetail data type. An event will be logged in the DSView Event Log.

Error Messages:

- Invalid Unit Group ID – The unit group ID is empty or invalid.
- Invalid Unit Group Name – The unit group name is empty or invalid.
- Insufficient Access Rights – The user does not have the rights to edit unit groups.
- System Error – There is an error accessing the database.

deleteUnitGroup

Deletes a unit group from DSView

Input Parameters:

- unitGroupId: The unit group ID of the unit group. This field cannot be empty or invalid.

Returns:

- A value of 1 if the operation is successful. An event will be logged in the DSView Event Log.

Error Messages:

- Invalid Unit Group ID – The unit group ID is empty or invalid.
- Insufficient Access Rights – The user does not have the rights to edit unit groups.
- System Error – There is an error accessing the database.

3.10 Unit Group Service Data Definitions

3.10.1 Data types

Data types are generally used for input or parameters in certain Web Services API requests. The following table lists the data types and the Unit Group Service operations that use them.

UnitGroupDetail

- groupId – A unique group ID assigned when the unit group is added to DSView. Refer to the addUnitGroup, getUnitGroup and getlUnitGroupList, getUnitsFromUnitGroup, and renameUnitGroup API methods.
- groupName – The unit group name.
- parentGroupId – A unique group ID for the parent of a unit group.
- parentGroupName – The parent unit group name.
- type – The type of unit group (e.g., 1=Personal or 2=Global)

UnitIdType

- oid – A unique unit ID assigned when the unit is added to DSView. Refer to the getUnitsFromUnitGroup, addUnitsToUnitGroup, deleteUnitsFromUnitGroup API methods.
- name – The unit name.
- address – The IP address associated with the unit if applies.

UnitDetail

- unitId – A unique unit ID assigned when the unit is added to DSView. Refer to the getUnitsFromUnitGroup, addUnitsToUnitGroup, deleteUnitsFromUnitGroup API methods.
- name – The unit name.
- address – The IP address associated with the unit.
- type – The type of unit.
- custom0 – The first custom field.
- custom1 – The second custom field.
- custom2 – The second custom field.
- notes – A description for the unit.

3.11 User Group Service

This web service provides operations to third-party clients to manage LDAP user groups in DSView™ software and set the rights on them.

3.11.1 Definition file

The filename for the DSView software user group service definition file is dsviewService-UserGroup.wsdl.

Visit https://dsviewServer[:dsviewPort]/dsview/services/UserGroupsServiceApi?wsdl to retrieve the WSDL dynamically generated for you.

3.11.2 Operations

One of the following operations from the User Group service is returned:

- 1 = Operation success
- 2 = Invalid unit/unit group
- 3 = Invalid user group
- 4 = Insufficient rights
- 5 = Invalid right
- 6 = Right does not exist
- 7 = LDAP authentication server does not exist
- 8 = Invalid Role ID
- 9 = User group does not exist
- 10 = Invalid preemption level
- 11 = User group already exists

Table 3.4 User Group Service Options

NAME	DESCRIPTION
listExternalAuthService	Lists the external authentication services
getUserGroupsFromLDAP	Gets the User Group list from LDAP
addUserGroup	Adds an LDAP user group into the DSView software
deleteUserGroup	Deletes LDAP groups from the DSView software
modifyUserGroup	Modifies the User Group Name, Preemption Level and Role of a given LDAP group
queryUserGroup	Lists all the LDAP groups added into the DSView software
addUserGroupRights	Sets the rights to an LDAP user group on specific units
deleteUserGroupRights	Removes the rights of an LDAP user group on specific units
modifyUserGroupsRights	Modifies the LDAP user group rights on given units
getListUserGroupsRights	Lists all the rights of an LDAP user group on specific units

3.12 Remote Session Service

This web service provides a feature to open a video viewer session (KVM/Serial) remotely through which a third-party client can access the target devices.

3.12.1 Definition file

The filename for the DSView™ software remote session service definition file is dsviewService-RemoteSession.wsdl.

Visit https://dsviewServer[:dsviewPort]/dsview/services/ RemoteSessionServiceApi?wsdl to retrieve the WSDL dynamically generated for you.

3.12.2 Operations

One of the following operations from the Remote Session Service is returned:

- 001 = Operation is success

- 002 = Input data missing
- 003 = Remote system does not exist
- 004 = Remote system is closed
- 005 = User does not have authentication
- 006 = Operation not allowed for this remote unit
- 007 = Invalid input
- 008 = Remote session is used, but client does not want to preempt
- 009 = Remote unit does not match session type

Table 3.5 Remote Session Service Options

NAME	DESCRIPTION

connVMSession	Opens viewer for KVM/Serial session.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• Session Type - KVM/Serial session
	• Target Name - Target device name

terminateVMSession	Terminates a KVM/Serial session.
	The input parameters are:
	• Administrative credentials which include the following:
	  • Username - DSView software admin username
	  • Password - DSView software admin user password
	• OID - Appliance OID of the target device to be connected
	• Name - Appliance name of the target device to be connected
	• Address - Appliance address of the target device to be connected
	• Target Name - Target device name to which a session needs to be opened

3.13 Unit Service

Operations in the Unit Service allow your application to manage units (target devices and managed appliances) using the DSView software.

3.13.1 Definition File

The filename for the DSView™ software unit service definition file is dsviewService-Unit.wsdl.

Visit http[s]://dsviewServer[:dsviewPort]/dsview/services/UnitServiceApi?wsdl to retrieve the WSDL dynamically generated for you.

3.13.2 Operations

One of the following operations from the TDPowerOperation is returned:

- 1 = Operation success
- 2 = Operation failed; User provided request is invalid
- 3 = Remote system does not exist

- 4 = User does not have a right

One of the following operations from the getRequestStatus is returned:

- 0 = A request was received and processed
- 1 = ID for the request is not created
- 2 = Unit was not found
- 3 = No path found to contact unit
- 4 = No owner (plug-in) found to execute the request
- 5 = Request Id is invalid
- 6 = Request completed
- 7 = Invalid operation type
- 8 = Invalid request
- 9 = Owner does not support the operations interface
- 10 = Request completion could not be verified
- 11 = No more room in the request queue
- 12= Request received and is in process

Table 3.6 Web Services API Unit Service Operations

NAME	DESCRIPTION
getVersion	Requests the current version of the Unit Service from the DSView software server. The format of the returned string is: majorVersion.minorVersion. A change in minorVersion indicates the service is modified but maintains backward compatibility with existing client implementations at the same *majorVersion* level. A change in majorVersion indicates the service is modified and cannot maintain backward compatibility.
getUnitList	Requests the DSView software server retrieve a list of units that match a filter. You can identify a set of attributes to be returned for each unit, along with the set of attributes to be searched (if any) for a match by the filter. The valid data types in this request are: • Unit types • Unit attributes • Unit filter attributes • Unit actions
getUnitDetail	Requests the DSView software server retrieve a set of attributes for a specific unit. The valid data types in this request are: • Unit types • Unit attributes • Unit actions
doAction	Requests the DSView software server perform the requested action on the specified unit. The valid data type in this request is unit actions.
performTDPowerOperation	Requests the DSView software server power off and power on the specified units. The input parameters are: • Username - DSView software admin username • Password - DSView software admin password • Names - Device name to be power cycled • IP address (optional) – IP address of the device to be power cycled
getRequestStatus	Requests DSView software server to get the status of all the devices that have been powered off and powered on. Request ID - requests device IDs from the devices returned as a result of the performTDPowerOperation
getEffectiveRights	Show the user and user group effective rights for a unit or unit group when the target information is entered. The input parameters are: • Target (Mandatory) - Enter one of the following values: Unit Name, Unit ID, Unit IP Address, Unit Group Name or Unit Group ID • Type (Optional) - Enter **user** or **usergroup**; if you leave the field blank the web service displays both user and user group

Table 3.6 Web Services API Unit Service Operations (continued)

NAME	DESCRIPTION
GerPowerStatus	Returns the current power status for the outlet. The input parameters are: • OID (Optional) - Enter the object identifier • Name (Optional) - Enter the name of the PDU or outlet
DoAction	Performs the tool action for the device. The following appliance and PDU commands are supported for this release: • Reboot (appliance) • Shutdown (appliance) • Power Operation (PDU and outlets) The input parameters are: • OID - Enter the object identifier • Name (Optional) - Enter the name of the PDU or outlet • IP address (Optional) - Enter the appliance or PDU IP address • Command (Optional) - Enter one of the following commands: • acvtWallPowerOff/avctWallPoweron/avctWallPowerCycle • avctIPMIPowerON/avctIPMIPowerOff/avctIPMIPowerGracefulOff/ avctIPMIPowerCycle • avctIPMIReset/avctIPMIIdentityOn/avctIPMIIdentityOff/avctReboot/avctShutdown
getUnitDetailForDCP	Returns all detailed information for a particular unit including supported power operations, reboot and shutdown details. The input parameters are: • OID - Enter the object identifier • Name (Optional) - Enter the name of the PDU or outlet • IP address (Optional) - Enter the appliance or PDU IP address • showFields (Optional)

3.13.3 Additional unit service operations

addAppliances

Discovers appliances on the network by IP address and adds them to DSView. The functionality of this API is similar to the Add Unit Wizard feature and uses the default settings (e.g., Merge Target Devices with the same name, Enable Secure Mode, etc) specified in *System - Global Properties - Wizard Defaults - Add Unit* Wizard page of the DSView Web UI to enroll any discovered appliances in the DSView system.

NOTE: Before executing this API method, make sure that the Enable Secure Mode configuration property is setup correctly and the appliance is not in secure mode. If the appliance is in secure mode, this API method will not be able to add the appliance to DSView.

NOTE: Since the discovery of appliances on the network takes time, the execution of this API occurs in the background. You can use the unique timestamp and call the getAddApplianceStatus API method to determine whether the discovery and enrollment of appliances is complete and get the list of appliance units that are added to DSView.

NOTE: Multiple API calls to this method using the same IP address range is not recommended.

Input Parameters:

- discoverSettings: The configuration settings that are used during the discovery of appliances on the network. Refer to the DiscoverSettings data type.

Returns:

- Returns a unique timestamp indicating the time when this operation has started on the background.

Error Messages:

- Invalid IP Address Type – The IP address type is not valid.
- Unsupported IPV6 Address Type – The IPV6 address type is currently not being supported.
- Invalid IP Address Search Type – The IP address search type is not valid.
- Invalid IPV4 From Address – The IPV4 From address is not valid.
- Invalid IPV4 To Address – The IPV4 To address is not valid.
- Invalid IPV4 Range Address – The IPV4 Range address is not valid.
- Invalid IPV4 Address List – An address in the IPV4 Address list is not valid.
- Invalid Plugin ID – A plugin ID in the list of plugin IDs is not valid.
- Insufficient Access Rights – The user does not have the rights to edit units.
- System Error – There is an error accessing the database.

getAddApplianceStatus

Gets the status of the execution of the addAppliances API method when discovering appliances on the network to add them to DSView.

Input Parameters:

- timestamp: The timestamp associated with the call to the addAppliances API method.

Returns:

- An AddApplianceStatus object which contains the status of the execution of the addAppliances API method. Refer to the AddApplianceStatus data type.

Error Messages:

- Add Appliance Status Not Found – The status for adding appliances is not found. It is possible that the timestamp does not match the one returned by the addAppliances API method.
- Insufficient Access Rights – The user does not have the rights to edit units.
- System Error – There is an error accessing the database.

renameUnit

Renames an existing unit DSView.

Input Parameters:

- unitId: The unique unit identifier for the unit to delete. Refer to the UnitIdType data type.
- unitNewName: The new name to assign to the unit. This value cannot be empty.

Returns:

- A UnitDetail object which contains the details of the updated unit which includes the unitId, name, address and type fields. Refer to the UnitDetail data type. An event will be logged in the DSView Event Log.

Error Messages:

- Invalid Unit ID – The unit ID is empty or invalid.
- Invalid Unit Name – The new unit name is empty.
- Insufficient Access Rights – The user does not have the rights to edit units.
- System Error – There is an error accessing the database.

deleteUnit

Deletes an existing unit along with any target devices connected to it from DSView. If the unit is an appliance and the global setting to prevent target devices to be deleted is enabled, then the appliance will be deleted and the target devices connected to the appliance will remain in the DSView system.

Input Parameters:

- unitId: The unique unit identifier for the unit to delete. Refer to the UnitIdType data type.

NOTE: Both the unit OID and unit name in the unitId input parameter must be provided. The unit name must be an exact match to an existing unit name in DSView with the same unit OID that was provided as input parameter.

Returns:

- A value of 1 if the operation is successful. An event will be logged in the DSView Event Log.

Error Messages:

- Invalid Unit ID – The unit ID is empty or invalid.
- Invalid Unit Name – The unit name is empty or does not match a unit in the database.
- Insufficient Access Rights – The user does not have the rights to edit units.
- System Error – There is an error accessing the database.

getUnitConnectionList

Gets a list of connections (e.g., KVM, Serial, Power) from a unit.

Input Parameters:

- unitId: A unique ID for a unit. Refer to the UnitIdType data type.

Returns:

- A list of UnitConnectionDetail objects which contains the detailed information for the list of unit connections. Refer to the UnitConnectionDetail data type.

Error Messages:

- Invalid Unit ID – The unit ID is empty or invalid.
- Insufficient Access Rights – The user does not have the rights to view units.
- System Error – There is an error accessing the database.

mergeUnitConnections

Merges connections (e.g., KVM, Serial, Power) from two or more units into a single unit in DSView. This functionality is similar to the Merge Target Devices feature in the unit overview page of the DSView Web UI.

Input Parameters:

- fromUnitIdList: A list of UnitIdType objects which contains the list of IDs for units that have connections to merge. Refer to the UnitIdType data type.
- toUnitId: A unique ID for the unit that will have all merged connections. The name of this unit will be unchanged after the merge operation has been completed. Refer to the UnitIdType data type.

Returns:

- A value of 1 if the operation is successful. An event will be logged in the DSView Event Log.

Error Messages:

- Invalid Target Unit ID – The merge unit connections operation is supported only by target device units.
- Invalid From Unit ID – The unit ID for one of the units in the From unit list is empty or invalid. The units in the From unit list must be target devices.
- Invalid To Unit ID – The unit ID for the To unit is empty or invalid.
- From and To Units Must be Different – The unit ID for a unit in the From unit list cannot be the same as the unit ID for the To unit parameter.
- Insufficient Access Rights – The user does not have the rights to edit units.
- System Error – There is an error accessing the database.

UnmergeUnitConnections

Removes connections (e.g., KVM, Serial, Power) from a previously merged unit in DSView. This functionality is similar to removing connections from the Target Device Connections page of the DSView Web UI.

NOTE: The execution of this method may trigger a resync of the units whose connections have been removed has been executed and therefore will take some time to complete.

Input Parameters:

- fromMergedUnitId: A unique ID for a previously merged unit. Refer to the UnitIdType data type.
- listUnitConnections: A list of UnitConnectionDetail objects which contains the list of connections to remove from the merged unit. Refer to the UnitConnectionDetail data type and getUnitConnectionList API method.

Returns:

- A value of 1 if the operation is successful.

Error Messages:

- Invalid Target Unit ID – The unmerge unit connections operation is supported only by target device units.

- Invalid From Unit ID – The unit ID for the From unit to unmerge is empty or invalid.
- Invalid Connection ID – The connection ID for one of the connections in the list of unit connections to remove is invalid.
- Invalid Unit ID in Unit Connection to Remove – The unit ID for the target device to unmerge must match the target side unit in a unit connection to remove.
- Insufficient Access Rights – The user does not have the rights to edit units.
- System Error – There is an error accessing the database.

updateUnitTopology

Updates the topology for a unit (e.g., an appliance) in DSView. The functionality of this API is similar to the Appliance Resync feature and uses the default settings specified in *System - Global Properties - Wizard Defaults - Resync* Wizard page of the DSView Web UI.

NOTE: The completion of this API method will depend on the number of PDU, cascade, and target connections to resync from the unit.

When the Auto Topology and Unit Status polling features are enabled, the resynchronization of the unit topology will occur automatically on the background.

Input Parameters:

- unitId: The unique unit identifier for the unit to update topology. Refer to the UnitIdType data type.

Returns:

- A value of 1 if the operation has completed successfully.

Error Messages:

- Invalid Unit ID – The unit ID is empty or invalid.
- Unable to Discover Unit – Unable to discover unit with given unit ID on the network.
- Insufficient Access Rights – The user does not have the rights to edit units.
- System Error – There is an error accessing the database.

getPluginList

Gets the list of plugins in DSView software. The plugin information is the same information displayed on the System->Plug-ins page of the DSView Web UI.

Input Parameters:

- None.

Returns:

- A list of PluginDetail objects that contains the details of each plugin in DSView. Refer to the PluginDetail data type.

Error Messages:

- None.

3.13.4 Unit Service data definitions

Data types

Data types are generally used for input or parameters in certain Web Services API requests. The following table lists the data types and the Unit Service operations that use them.

Table 3.7 Web Services API Data Types

DATA TYPE	UNIT SERVICE OPERATIONS THAT USE THIS DATA TYPE
Unit Types	getUnitList() getUnitDetail()
Unit Attributes	getUnitList() getUnitDetail()
Unit Filter Attributes	getUnitList()
Unit Actions	getUnitList() getUnitDetail() doAction()

Unit types

The DSView™ software has well-defined unit types for managed appliances, cascade devices and power devices. In addition, as a user of the DSView software, you can define unit types for target devices. By default, the unit type for target devices is blank (empty). The user-defined unit types are strings and can be used in Web Service API requests in the same manner as any of the well-defined unit types.

The SDE recognizes only the Avocent DSR1021, CCM1650 and DSI5100 managed appliances.

The following table lists the unit types and models. See the Avocent® DSView™ 4.5 Management Software Release Notes for a current list of supported unit types and models.

Table 3.8 Web Services API Unit Types

DSVIEW SOFTWARE UNIT TYPES	MODELS
ACS advanced console server	ACS6000
Avocent® AutoView	AV2108 and AV2216
Avocent® MergePoint Unity™	1016, 1016DAC, 104E, 108E, 108EDAC, 2016, 2016DAC, 2032, 2032DAC, 4032, 4032DAC, 8032 and 8032DAC
Avocent® Power Management Distribution Unit (PM PDU)	PM1000, PM2000 and PM3000
Avocent® SPC power control devices	N/A
Avocent® Universal Management Gateway	2000, 4000 and 6000
Liebert® Rack PDUs	MPH2™ and MPX™ (with RPC-2000 cards)
Liebert® GXT4™ UPS devices	N/A
Liebert® PDUs	MPH™ and MPX™ (with RPC-1000 cards)
APC	AP78xx, AP79xx, AP8881, AP8661 and AP8941
Blades	Dell DRAC MC, Dell M1000e, FSC BX600, Generic Blade Chassis, HP BladeSystem, HP BladeSystem c-Class, HP BladeSystem p-Class, IBM BladeCenter, IBM BladeCenter H, IBM BladeCenter HT and IBM BladeCenter T
Citrix XenServer	6.5 Standard
Dell Appliances	Dell 2161DS-2, Dell 2321DS, Dell 4161DS and Dell 8321DS
Dell Remote Console Switch	1082DS, 2162DS and 4322DS
FW Digital KVM Switch	FW-D1008NP, FW-D2016NP, FW-D2032NP, FW-D4016NP and FW-D8032NP
Fujitsu KVM s4 Appliances	KVM s4-0812, KVM s4-1622 and KVM s4-3242
HP Console Switch G2	Server Console Switch G2 0x2x16, Server Console Switch G2 0x2x32, IP Console Switch G2 1x1Ex8, IP Console Switch G2 2x1Ex16 and IP Console Switch G2 4x1Ex32
HP G3 KVM Console Switch	0x1x8 G3 KVM Console Switch and 0x2x16 G3 KVM Console Switch
IBM Console Switch	GCM16/32 product family
KVM Switch	1x1x8 switch, 2x1x16 switch and 8x1x32 switch
Lenovo GCM	16/32 console switch product family
Lenovo LCM	8/16 switch product family
Microsoft Hyper-V2 on Windows 2008 R2	N/A

Table 3.8 Web Services API Unit Types (continued)

DSVIEW SOFTWARE UNIT TYPES	MODELS
Microsoft Hyper- V3 on Windows 2012 and 2012 R2	N/A
ServerTech Sentry Switched CDU	CW-8H1*, CW-8H2*, CW-16V1*, CW-16V2*, CW-24V2*, CW-24V3*, CW-32VD1*, CW-32VD2*, CW-48V5Z454-A1P, CW-24VY-L30M, CWG-24V4Z423A9/QR, CW-8H1A413,CW-24V4K425A9, STV-4501C,STV-6502M and STV-4501C **NOTE: Models with an asterisk (*) must be attached to a managed appliance such as the Avocent® MergePoint Unity™ KVM over IP and serial console switch.**
VMware	5.5, 5.5u2 and 6.0 **NOTE: If using VMware 6.0, the VI Client link under the Virtual Center no longer points to a location where the VMWare Client can be downloaded. VMWare has removed this link, so customers must manually download the VMWare Client executable from VMWare or ESXi.**

Unit attributes

The following table describes the unit attributes that can be retrieved from DSView™ software.

Table 3.9 Web Services API Unit Attributes

UNIT ATTRIBUTE	DESCRIPTION
avctUnitId	Unique ID that represents a specific unit in the DSView software. This attribute is always returned.
avctUnitName	Unit name.
avctUnitType	Unit type.
avctUnitAddress	Unit address.
avctUnitNotes	Notes associated with the unit.
avctUnitCustom0	Custom field 0 associated with the unit.
avctUnitCustom1	Custom field 1 associated with the unit.
avctUnitCustom2	Custom field 2 associated with the unit.
avctUnitSupportedActions	Actions supported by the unit.
avctUnitSessions	Comma separated list of sessions available on the unit.

Unit filter attributes

The following table describes the unit attributes that can be searched (used as a filter) for getUnitList requests.

Table 3.10 Web Services API Unit Filter Attributes

UNIT FILTER ATTRIBUTE	DESCRIPTION
avctUnitID	Unit ID
avctUnitName	Unit name
avctUnitType	Unit type
avctUnitAddress	Unit address
avctUnitNotes	Notes associated with the unit
avctUnitCustom0	Custom field 0 associated with the unit
avctUnitCustom1	Custom field 1 associated with the unit
avctUnitCustom2	Custom field 2 associated with the unit

Unit Actions

The following table describes the unit actions that are supported by the DSView software and can be returned as supported actions of a unit.

Table 3.11 Web Services API Unit Actions

UNIT ACTION	DESCRIPTION
avctWallPowerOff	Powers off each of the SPC sockets connected to the unit
avctWallPowerOn	Powers on each of the SPC sockets connected to the unit
avctWallPowerCycle	Power cycles each of the SPC sockets connected to the unit
avctIPMIPowerOn	Issues the power on command to each of the IPMI connections to the unit
avctIPMIPowerOff	Issues the power off command to each of the IPMI connections to the unit
avctIPMIPowerGracefulOff	Issues the graceful power off command to each of the IPMI connections to the unit
avctIPMIPowerCycle	Issues the power cycle command to each of the IPMI connections to the unit
avctIPMIReset	Issues the reset command to each of the IPMI connections to the unit
avctIPMIIdentityOn	Issues the identity on command to each of the IPMI connections to the unit
avctIPMIIdentityOff	Issues the identity off command to each of the IPMI connections to the unit and disables the Identity feature

3.13.5 Unit Connection Data Types

The following describes the data types associated with unit connections.

UnitConnectionDetail

- connectionId – A unique ID for a unit connection. Refer to the getUnitConnectionList API method.
- connectionType – The unit connection type (e.g., 1, 2, 3, etc).
- connectionTypeName– The name of the unit connection type (e.g., KVM, Serial, Power)
- sourceSideUnitId – A unique ID for the source side unit of the unit connection.
- sourceSideUnitName – The name of the source side unit of the unit connection.
- sourceSideUnitPort – The port on the source side of the unit connection.
- targetSideUnitId – A unique ID for the target side unit of the unit connection.

- targetSideUnitName – The name of the target side unit of the unit connection.
- connection Path – The connection path from the source side unit to the target side unit.

NOTE: The Target Side Unit Port is not needed since it is always set to -1.

3.13.6 Unit Discovery Data Types

The following information describes the data types associated with the discovery of units.

DiscoverSettings

- ipAddressSettings – The IP Address settings to find appliances on the network are as follows:
  - ipAddressType – The type of address to use. Valid values are IPV4 and IPV6. These values are case insensitive (e.g., ipv4 and ipv6 are also valid values).
  - ipAddressSearchType – This indicates whether to search IP addresses by range or by specifying a list of IP addresses. Valid values are SearchByRange or SearchByIP. These values are case insensitive (e.g., searchbyrange and searchbyip are also valid values).
  - ipV4FromAddress – When the IP Address Type is IPV4 and the IP Address Search Type is SearchByRange, you must specify a valid IPV4 From Address.
  - ipV4ToAddress – When the IP AddressType is IPV4 and the IP Address Search Type is SearchByRange, you must specify a valid IPV4 To Address.
  - ipV4AddressList – When the IP Address Type is IPV4 and the IP Address Search Type is SearchByIP, you must specify a comma separated list of valid IPV4 addresses.
  - ipV6AddressList – When the IP Address Type is IPV6 and the IP Address Search Type is SearchByIP, you must specify a comma separated list of valid IPV6 addresses.
  - ipV6NetworkPrefix – When the IP Address Type is IPV6 and the IP Address Search Type is SearchByRange, you must specify a valid IPV6 Network Prefix.
- snmpV1Settings – The SNMP settings to find appliances on the network are as follows:
  - snmpReadCommunity – The Read Community for reading SNMP values. The default value is public.
  - snmpWriteCommunity – The Write Community for updating SNMP values. The default value is private.
- portSettings – The port settings to find appliances on the network are as follows:
  - httpPort – The HTTP Port to use during search of appliances on the network. The default value is 80.
  - httpsPort – The HTTPS Port to use during search of appliances on the network. The default value is 443.
- pluginSetttings – The plugin settings to find appliances on the network are as follows:
  - pluginIds – A comma separated list of unique plugin IDs to limit the number of plugins to use during the discovery of appliances on the network. If this list is not specified, all active plugins will be used during the appliance discovery. Refer to the getPluginList API method to get the list of plugins from DSView.

AddApplianceStatus

- discoverStatus – The status of the discovery of appliances on the network.
  - Not Started
  - Started

    - In Progress
    - Completed
    - No New Appliances Found
- enrollmentStatus – The status of the enrollment of appliances in the DSView system.
    - Not Started
    - Started
    - In Progress
    - Completed
    - Partially Completed - A list of appliances that failed to enroll is displayed.
    - Failed – A list of appliances that failed to enroll is displayed.
- enrolledApplianceList – The list of UnitDetail objects for any appliance units that were added to DSView. This list will be populated when the enrollment status has changed to Completed.

PluginDetail

- pluginId – A unique plugin ID. Refer to the getPluginList API method.
- pluginName – The plugin name.
- pluginDescription – The plugin description.
- pluginVersion – The version of the plugin.
- pluginVendor – The plugin vendor.
- overallStatus – The overall status of the plugin.
    - Mixed – A mixed administrative and operational status.
    - Same – The same administrative and operational status
- administrativeStatus – The administrative status of the plugin.
    - Unknown
    - Replication Needed
    - Active
    - Disabled
    - Mixed
    - Not Installed
    - DSView Server Not Responding
    - License Required
- operationalStatus – The operational status of the plugin.
    - Unknown
    - Inactive
    - Active
    - Initializing
    - Shutting Down
    - Upgrading
    - Mixed

3.14 SOAP Faults

All service requests can potentially cause an error condition. The service attempts to handle error conditions where possible and continue processing the request. Error conditions that prevent the service from completing successfully return a SOAP fault.

3.14.1 SOAP fault fields

The following fields appear in the DSView software:

- faultcode - A field that contains fault code values. The following table defines the error status of each fault code.
- faultstring - A field that contains a brief description of the problem or error.
- faultactor - The URI of the service causing the fault.
- detail - A field in the DSView software that is unused.

Table 3.12 Web Services API SOAP Fault Codes

FAULT CODE VALUE	DESCRIPTION
BadCredentials	Credentials (username or password) not valid.
SystemError	DSView software system level error. Normally not expected.
LicenseLimitReached	Maximum number of allowed user sessions is reached.
UserSessionExpired	User session (HTTP session) used with the request has expired.
UserAccountDisabledLocked	User account associated with the request is disabled or locked.
UserPasswordExpired	User account associated with the request has an expired password.
OperationRequiresAuthentication	Request requires user credentials or a previously established session. User authentication is required.
NotAuthorized	User account associated with the request is not authorized to perform this request.
UnitRequestedNotFound	Unit associated with the request is not defined in the DSView software system.
UnitRequestedCannotPerformAction	Action requested is not supported by the specified unit.
WebServicesDisabled	Requested service is not operational.
MultipleMatchingUnitsFound	Unit associated with the request identifies multiple units and the request is intended to be applied to a single unit.

NOTE: Additional SOAP faults can also be generated. These are the SOAP-defined faults for SOAP version 1.1. These faults cannot follow the previously provided format.

3.14.2 Definition file

The filename for the SOAP fault definition file is serviceComponent-Common.xsd.

3.15 Java Sample Using Axis

The following procedure runs the sample application Java client, displays SOAP requests and responses and shows the effect in the SDE.

To run the Java client sample application:

1. Run the SDE.
2. Run scripts related to the sample Java client.

3. Change to the directory where the scripts reside:
    - On a Microsoft Windows system, the directory is avocent\dssdk\samples\WebService\Java.
    - On a Linux system, the directory is avocent/dssdk/samples/WebService/Java.

NOTE: A keystore for HTTPS (SSL) holds keys and digital certificates for trusted certificate authorities. See the following procedure for steps to create a keystore.

4. If connecting to the SDE with HTTPS (SSL), create a keystore.

    -or-

    Get the certificate of the DSView software server and save it as dsview.cer in the sample/webservices/java directory.

5. Build the sample Java client by double-clicking the following script:
    - Microsoft Windows: buildClient.bat
    - Linux: buildClient.sh
6. Press a key after the prompt.
7. Run the sample Java client in its own screen by double-clicking the following script:
    - Microsoft Windows: runClient.bat
    - Linux: runClient.sh
8. To run the TCP monitor and see SOAP requests and responses in their own screen, double-click the following script:
    - Microsoft Windows: runTcpmon.bat
    - Linux: runTcpmon.sh

NOTE: By default, the TCP Monitor is listening on port 8081 and forwarding messages to port 8080 (the port number of the DSView software server). SOAP messages can only be viewed if HTTP is used.

9. Select the *XML Format* checkbox in the TCP Monitor screen.
10. In the sample Java Client screen, enter 100 at the Enter Selection prompt.
    a. Enter http://localhost:8080 to change the protocol, server and port number at the Enter URL Base prompt.
    b. Enter the number of the desired menu entry.
    c. Enter the responses to other prompts and read the bracketed DSView software server responses.
11. To verify an added event in the sample Java Client screen in the SDE, click the *Reports* tab and click *Event Log Views - All* in the side navigation bar.

NOTE: The TCP Monitor screen contains the SOAP request that corresponds to your request and the SOAP response from the DSView software server.

To create a keystore:

1. Run the following script:
    - Microsoft Windows: createkeystore.bat
    - Linux: createkeystore.sh
2. Press a key after the prompt.
3. Enter the keystore password when prompted.

NOTE: WebServiceSample.java must be modified to uncomment the lines in the constructor relating to the keystore setup.

3.16 C++ Sample Using Axis

The following procedure runs the sample application C++ client, displays SOAP requests and responses and shows the effect in the SDE.

To run the C++ client sample application:

1. Run the SDE.
2. In the Microsoft Visual Studio® .NET 2003 development system, open the *WebServiceSample.vcproj* in the avocent/dssdk/samples/WebService/CPP directory.
3. From the Build menu, click *Build Solution*.
4. From the Debug menu, click *Start*.
5. In the TCP Monitor screen, select the *XML Format* checkbox.
6. In the sample C++ Client screen:
    a. Enter 100 at the Enter Selection prompt.
    b. Enter http://localhost:8080 to change the protocol, server and port number at the Enter URL Base prompt.
    c. Enter the number of the desired menu entry.
    d. Enter the responses to other prompts and read the bracketed DSView software responses.
7. To verify an added event in the sample C++ client screen in the SDE, click the *Reports* tab and click *Event Log Views - All* in the side navigation bar.

4 PLUG-IN API

The plug-in API enables you to add and use elements in the DSView™ software, such as appliances and attached target devices.

This chapter describes which DSView software features are supported, prohibited and customizable in a plug-in. It also describes the Java interfaces and XML definition files that are used to create a plug-in.

When a supported feature is mentioned in this chapter, you can often find more information about it in the DSView Software Installer/User Guide and online help. The Vertiv web site also contains Technical Bulletins that can be helpful.

A sample plug-in API is available in the samples directory.

Nomenclature

The nomenclature for this document is as follows.

- Generally, this chapter uses the term 'connection' to indicate the successful launching of a session with a target device (for example, a Telnet session to a serial device or a viewer session to a device attached to a KVM appliance).
- When DSView software users log on or click a link, a screen in the DSView software Explorer opens. A content screen appears in the content area of a screen. Both terms are used interchangeably in this chapter.
- If you see references to NMM in general text or code comments, substitute the word plug-in. When NMM or nmm appears in an actual program element, such as a method or attribute, it should be used as specified.

4.1 About Plug-ins

A plug-in contains the following basic components:

- Module definition - Defines the appliance type and provides module configuration information.
- Navigation - Defines navigation trees and nodes that appear in the side navigation bar of the DSView software Explorer screen for the appliance type and its target devices.
- GUI content - Defines screens and links to Java classes that contain model data for the GUI content screens. A content screen is displayed in the content area of a screen. Content screen data can come from the database, the appliance or any source the plug-in supports.

  A content screen can contain three basic GUI types:

  - Property screens - contain read/write property values, which can be displayed in text fields, drop-down lists, number spinners or labels
  - Item tables - contain lists of items (for example, a Units View screen)
  - Wizards - contain a series of screens for performing some function (for example, the Add Appliance wizard)
- Appliance logic - Defines communications with the appliance type.

The plug-in API supports the localization of plug-in resources for navigation trees and nodes and content screens.

4.1.1 DSView software feature support

Adding appliances and target devices with the Add Unit wizard

The new appliance type defined in a plug-in is displayed in the Product list in the Add Unit wizard's Select Appliance Type screen. The plug-in API provides support for discovery of an appliance type with or without an IP address; the plug-in must provide the method for finding the appliance.

The Select Options screen in the wizard contains options specific to an appliance type. A plug-in can use the existing Select Options screen, omit the screen from the Add Unit wizard or replace the existing screen with one or more custom content screens.

During discovery, if two different appliance types are discovered and selected for adding, all Select Options screens are shown for each appliance type.

Resync wizard

If a plug-in supports the resynchronization feature, the Resync wizard can be used. The Resync wizard displays all Select Options screens that are defined for the appliance type.

Units View screens

After an appliance is added to the DSView™ software, it is included in the Units View screens that contain appliances of that type. Similarly, target devices that have been added (either when the appliance was added or subsequently) are included in Units View screens that contain target devices.

The following information or actions are supported for a plug-in in a Units View screen:

- Type - Displayed in the Type field of the content area and as a node in the side navigation bar.
- Icon - A graphic representation of the appliance that appears on the screen.
- Status - Contains the appliance or target device status gathered from the plug-in.
- Target devices - Target devices are displayed when the Topology view is enabled in a Units View screen that contains appliances. Target devices are also displayed when a link in the Target Devices tree in the side navigation bar is selected (All target devices or a specific target device type).
- Action - Target devices can use existing connection types or the plug-in can define a new connection type. See Connection types on page 59 that provide which screens can contain connection types.
- A target device session type must be HTML-based.
- Multiple unit operations - The Operations button/menu can be used to initiate actions on one or more units. The plug-in can define new operations to be supported, or it can use any of the following existing operations:
  - For appliances: Hide Units from View, Reboot, Show Version, Push Names to Appliance, Pull Names from Appliance and Properties.
  - For target devices: Hide Units from View and Properties.
- Access rights - The same rights that are available for the appliance types and target devices that are automatically supported in the DSView software are available for appliance types and target devices defined in a plug-in.

Appliance tools

Clicking on a unit name in a Units View screen opens the Unit Overview screen. The Unit Overview screen for an appliance contains icons and links for supported tools. A plug-in can use some of the existing appliance tools. It can also define new tool buttons. These tools can link to custom content screens or they can initiate an operation on the appliance. If the tool initiates an operation, the result of the operation is displayed in the Operations Results screen.

NOTE: A plug-in can use all the existing appliance tools except Manage Power Devices.

Appliance navigation

The plug-in must define the content of the side navigation bar (navigation trees and nodes it displays) when the appliance is selected in a Units View screen containing appliances. The plug-in can display any of the existing nodes/trees (such as the Properties tree and nodes), as well as custom navigation nodes with custom content screens.

A node is a single entity that displays a single content screen; it is denoted with a screen icon and cannot be expanded. A tree contains an arrow icon that expands when clicked, to display one or more nodes or additional trees/nodes.

An appliance side navigation bar has the following rules:

- The Overview and Access rights nodes are always displayed (assuming the user has viewing access rights).
- A plug-in can add custom nodes under the Properties tree, but they must be added at the end of the tree (after existing nodes). The plug-in can only add nodes under the Properties tree; it cannot add a tree under the Properties tree.
- A plug-in can add trees and nodes after the Access Rights node.

Target device navigation

Unlike an appliance navigation tree, the navigational tree for a target device is the same for all target devices on an appliance. A target device side navigation bar has the following rules:

- A plug-in can add custom nodes under the Properties tree, but they must be at the end of the tree (after existing nodes).
- A plug-in can add trees and nodes after the Access Rights node.

Appliance firmware files

A plug-in can optionally upload firmware files for upgrading the appliances it supports. The header information in the appliance firmware file must follow the Vertiv Flash Upgrade File Format Specification; contact Vertiv for details. See the Avocent® DSView™ 4.5 Management Software Installer/User Guide and online help for the uploading procedure.

Connection types

When a plug-in is added to the DSView software, any new connection types that are defined in the plug-in are added to screens that contain available connection types.

The new connection type also appears in the following screens:

- Units View screens containing appliances - Action field

- Unit Overview screen for a target device - link and icon
- Automatic Name Push Properties screen - checkbox
- Automatic Name Pull Properties screen - listbox
- Auto Topology Update Properties screen - checkbox
- Add Unit Wizard Default Properties screen - listbox (this affects the content of the Add Unit Wizard Select Options screen)
- Add Unit Wizard Select Options screen - listbox
- Resync Wizard Default Properties screen - listbox (this affects the content of the Resync Wizard Select Options screen)
- Resync Wizard Select Options screen - listbox
- Add Target Device Connection Wizard Select Connection Type screen - listbox

System tasks

A plug-in can contain support for executing the following tasks:

- Upgrade the appliance firmware
- Control power of target devices

You cannot use system tasks to configure SNMP trap settings or migrate units.

Event logs

A plug-in can use the existing events defined in the DSView™ software. The plug-in can also define new events. New events must use existing severity levels and categories. These new events are added to the lists of events that can be enabled or disabled to trigger an email notification and/or post to the event log.

4.2 Integration with the DSView Software

After you have created a plug-in using the DSView SDK plug-in API, complete the following sequence of procedures.

To use a plug-in in the DSView software system:

1. Obtain a plug-in license from Vertiv, if necessary.

   The DSView SDE allows plug-ins to operate without licensing. However, the DSView software Enterprise Edition requires a plug-in license for each plug-in that supports a non-Vertiv product. If a license is needed, it must be obtained from Vertiv and installed on the DSView software system before adding a plug-in. See the Vertiv web site or contact your Vertiv representative about to obtain licenses.

2. Add the plug-in to the DSView software hub server.
3. Add the plug-in to each of the spoke servers.
4. Initiate a replication operation on all servers.
5. Add appliances to the DSView software.

The DSView Software Installer/User Guide and online help describe the procedures for installing licenses, adding plug-ins, initiating replication and adding appliances.

4.3 Implementation Guidelines

Use system resources in a manner that does not have a detrimental effect on the performance of the DSView software or other plug-ins. Similarly, the DSView software performance should not be degraded due to interactions with a plug-in. All interface implementations must be thread-safe, perform quickly and use pooling where appropriate to improve performance and resource use.

Only the files and directory structure defined for the plug-in JAR file are allowed.

All interfaces that are supported by the plug-in must be available when the implementation class is constructed.

4.3.1 Multi-addressed DSView™ software servers

At times, units that are managed by the DSView software must be configured with the address of a DSView software server, for example, SNMP trap destinations. In those situations, use the following guidelines (which are listed in preferred order) to determine how to provide the address to the unit.

- Method 1 - The unit determines the address of the DSView software server, based on connection information (address of socket) that is established from the plug-in running on the DSView software server to the unit.
- Method 2 - The plug-in establishes a connection with the unit and based on the connection information (address of socket), the plug-in provides the address to the unit via some configuration protocol.
- Method 3 - The plug-in determines the address(es) of the DSView software server using java.net.NetworkInterface and provides the address to the unit via some configuration protocols.

4.3.2 System properties

Java provides methods to access system properties via System.getProperties(). The DSView software uses these properties for customizations that can be needed during installations to help resolve an issue/problem. The use of properties should only be used as noted in this document or as directed by a Vertiv representative; they should not be expected or required.

Property Name/Key

To prevent plug-ins from encountering namespace issues with system properties, the following property name convention must be followed:

<domainId>|<nmmId>|<property>

Parameters

The following table lists the parameters that comprise the property name convention.

PARAMETER	DESCRIPTION
<domainId>	Value specified in the domain attribute of the identity section of the plug-in's nmm.xml definition file.
<nmmId>	The unique ID of each individual plug-in. The value is specified in the nmmId attribute of the identity section of the plug-in's nmm.xml definition file.
<property>	Name identified by the plug-in for a desired property.

4.4 Programming Interfaces

Java interfaces and XML definition files define the interaction between a plug-in and the DSView software. See the Oracle® Javadoc Tool for more information.

4.5 Java Interfaces

There are four primary Java interfaces:

- DSView™ Interface - enables the plug-in to interact with the DSView™ software.
- Plug-in Interface - enables the DSView software to interact with the plug-in.
- DSView™ Listener - enables the DSView software to send information to the plug-in asynchronously.
- Add-on Context - enables the plug-in to access context information

This section contains details about these primary interfaces, the secondary interfaces and other programming considerations. The Oracle® Javadoc Tool can contain additional information.

4.5.1 DSView interface

The DSView interface enables the plug-in to interact with the DSView software through secondary interfaces (services) that can be retrieved by get functions. The following services can be retrieved by get functions:

- Notification Service
- License Manager Service
- Repository Service
- Utility Service
- Client Session Service
- Operation Service
- Dialup Manager Service
- Email Service
- System Information
- Cached Credential
- Plug-in Communicator Service

Notification Service - getNotificationService()

The DSView software can asynchronously notify plug-ins about specific conditions. These notification messages are sent to registered objects that implement the DSView Listener interface.

The Notification Service can be used to:

- Register or unregister for notifications a plug-in receives
- Specify filter criteria
- Specify the listener/receiver of the notifications

The Notification Service uses the following methods:

- String register (Map registrationMap, DSViewListener listener) - Registers for asynchronous notifications from the DSView software.

    - String (return value) - Registration ID (for use when unregistering a notification).
    - registrationMap - Map containing registration information, including matching criteria to identify the data for which notifications are requested.
    - listener - Object that implements the DSViewListener interface, which is called when notifications match the passed registration.
- unregister (String szRegistrationId) - Unregisters a notification request.
    - szRegistrationId - ID of the registration to be removed/unregistered. If a null ID is passed, all registered notifications for this plug-in are removed.

License Manager Service - getLicenseManagerService()

The License Manager Service can be used to:

- Obtain license information
- Determine if an entity is licensed
- List units or connections covered by a particular license type
- Reserve a license
- Release a license

The License Manager Service uses the following methods:

- Map licenseInfo (Map licenseMap) - Returns information about a license type.
    - licenseMap - Map representing the license type. The map classification is CLASSIFICATION_LICENSE. See the License Classification Attribute table in Classification attributes and values on page 137 for details.
    - Map (return value) - Map containing the license information:

      License type

      Total number of licenses

      Total number of reserved or assigned licenses

      Total number of free or available licenses

The map classification is CLASSIFICATION_LICENSE. See the License Classification Attribute table in Classification attributes and values on page 137 for details.

- boolean isLicensed (Map licenseMap, Map entityMap) - Indicates if an entity is licensed.
    - licenseMap - Map representing the license type. The map classification is CLASSIFICATION_LICENSE. See the License Classification Attribute table in Classification attributes and values on page 137 for details.
    - entityMap - Map representing the entity. If the license type is for power management, this entity is a unit. If the license type is for the data logging service, this entity is a connection. See the Classification Attribute table or the Connection subsection in Classification attributes and values on page 137 for details.
    - boolean (return value) - returns true if the entity is licensed, false if it is not licensed.
- List query (Map licenseMap, Map entityMap) - Returns either unit connections or units that are licensed, based on the license type for an entity.

    - licenseMap - Map representing the license type. The map classification is CLASSIFICATION_LICENSE. See the License Classification Attribute table in Classification attributes and values on page 137 for details.
    - entityMap - Map representing the unit for which license information is wanted. The map classification is CLASSIFCATION_UNIT. See the Classification Attribute table in Classification attributes and values on page 137 for details.
    - list (return value) - List of maps of entities for the license type - connections (for the data logging service license type) or units (for the power management license type). See the Classification Attribute table or the Connection subsection in Classification attributes and values on page 137 for details.
- boolean reserve (Map licenseMap, Map entityMap) - Reserves a license of a specified type.
    - licenseMap - Map representing the license type to reserve. The map classification is CLASSIFICATION_LICENSE. See the License Classification Attribute table in Classification attributes and values on page 137 for details.
    - entityMap - Map representing the entity for which the license is reserved. If the license type is for power management, this entity is a unit. If a power management device (PMD) unit is added to the DSView™ software using the updateTopology() method of the Repository Service, a license is automatically reserved. If the license type is for the data logging service, this entity is a connection. See the Classification Attribute table or the Connection subsection in Classification attributes and values on page 137 for details.
    - boolean (return value) - returns true if the license is successfully reserved, false if not.
- void release (Map licenseMap, Map entityMap) - Releases a license.
    - licenseMap - Map representing the license type to release. The map classification is CLASSIFICATION_LICENSE. See the License Classification Attribute table in Classification attributes and values on page 137 for details.
    - entityMap - Map representing the entity for which the license is released. If the license type is for power management, this entity is a unit. If a power management device (PMD) unit is removed from the DSView software using the updateTopology() method of the Repository Service, the license is automatically released. If the license type is for the data logging service, this entity is a connection. See the Classification Attribute table or the Connection subsection in Classification attributes and values on page 137 for details.

Repository Service - getRepositoryService()

The Repository Service enables management of DSView software data using the following methods:

NOTE: Queries and searches of string values in the database are not case sensitive.

- addElement (Map elementMap) - Adds new elements to the DSView software.
    - elementMap - Map containing the element to be added. Only events and property table elements can be added using this method. See the Event Classification Attribute table or the Property Table Map Classification Attribute table in Classification attributes and values on page 137 for details.
- Map getElement (Map filterMap) - Obtains a specific element from the DSView software.
    - Map (return value) - Map representing the element in the DSView software that matches the filterMap. If more than one element is found that matches the filter, null is returned.

    - filterMap - Map containing a filter definition (the classification of elements to be located and the filtering requirements). Some types of elements cannot be retrieved using this method. See the Query Filter Classification Attribute table or the Query Filter Classification Attribute for Property table in Classification attributes and values on page 137 for details.
- List getElements (Map filterMap) - Obtains elements from the DSView software.
    - List (return value - List of maps representing the elements retrieved from the DSView software that match the filterMap. If no matches are found, an empty list is returned.
    - filterMap - Map containing a filter definition (the classification of elements to be located and the filtering requirements). There are restrictions on the types of elements that can be retrieved using this method. See the Query Filter Classification Attribute table or the Query Filter Classification Attribute for Property table in Classification attributes and values on page 137 for details.
- updateElement (Map elementMap) - Updates existing elements in the DSView™ software.
    - elementMap - Map containing the element to be updated. The map must contain a unique identifier for the element being added; generally, this is the element's OID. Only unit, unit connection and property table elements can be updated using this method. Only attributes specified in the map are updated.
- updateTopology (Map elementMap, Map propertiesMap) - Updates the topology of elements in the DSView software.
    - elementMap - Map containing the topology of the element to be updated. Unit objects with their associated topology (connections and units) can be updated. This map has the same format as the maps returned from ConfigManagementInterface.discovery.
    - propertiesMap - Map containing properties that control how the topology information is used to update the repository.
- deleteElements (Map elementMap) - Provides the ability to delete property table elements in the DSView software. A plug-in can only delete elements that it owns. However, a consolidated plug-in can delete its own elements and the elements of the old plug-ins it replaced. See the Consolidation Section subsection in Plug-in definition file (nmm.xml) on page 103 for details.
    - elementMap - Map containing the elements to be deleted from the DSView software. Only property table elements can be deleted using this method. See the Property Table Map Classification Attribute table in Classification attributes and values on page 137 for details.

Utility Service - getUtilityService ()

The Utility Service provides utilities to the plug-in, using the following methods:

- Certificate generation (self-signed)
- Generate kay pairs (public/private keys)
- Resource lookups

Client Session Service - getClientSessionService()

The Client Session Service allows a plug-in to perform the following functions in the DSView software: establish a new client session, terminate an existing client session, keep alive an existing client session or test whether a session is alive.

To establish a client session, the plug-in sends user information, (such as an HTTP request or response, username, password or certificate) to the DSView software for authentication. The user is then logged into the DSView software and a client session is opened. The DSView software generates a unique client session ID that is sent to the plug-in for subsequent HTTP requests. The DSView software tracks the new client session for licensing purposes.

NOTE: The list of authentication services can be accessed from the *Users - Authentication Services* screen in the DSView software. The number of available client sessions can be accessed from the *System - Licenses - Summary* screen in the DSView software.

The Client Session Service enables management of DSView software client sessions using the following methods:

- Map establishClientSession (Map clientSessionMap) - Establishes a client session in the DSView™ software.
  - clientSessionMap - A map containing the user information necessary to establish a client session in the DSView software. See the Client Session Map Classification Attribute table in Classification attributes and values on page 137 for details.
  - Map (return value) - A map containing the status of the client session establish request. See the Client Session Map Classification Attribute table in Classification attributes and values on page 137 for details.
- void terminateClientSession (Map clientSessionMap) - Terminates a previously established client session in the DSView software.
  - clientSessionMap - A map containing the information for the client session to be terminated in the DSView software. See the Client Session Map Classification Attribute table in Classification attributes and values on page 137 for details.

NOTE: If the client session is terminated by a session timeout, the client session ID is no longer valid in the DSView software. The user must log in to the DSView software to obtain a new client session ID for subsequent HTTP requests. To prevent the DSView software client session from timing out, the plug-in can request to keep the client session alive.

The session timeout value is configured by the DSView software administrator from the *System - DSView software server - DSView Client Sessions* screen in the DSView software.

- void keepAliveClientSession (Map clientSessionMap) - Sends a request to the DSView software to keep the client session alive. When this method is called, the client session timeout is reset.
  - clientSessionMap - A map containing the information for the client session to be kept alive by the DSView software. See the Client Session Map Classification Attribute table in Classification attributes and values on page 137 for details.
- boolean isClientSessionAlive (Map clientSessionMap) - Tests whether a client session is alive.
  - clientSessionMap - A map containing the information for the client session to test. See the Client Session Map Classification Attribute table in Classification attributes and values on page 137 for details.

Operation Service - getOperationService()

The Operation Service allows a plug-in to get a list of operations or execute operations for a given entity, such as a unit.

The following operations are defined by the DSView™ software for target device units and are supported by the Operation Service:

- OPERATION_POWER_ON
- OPERATION_POWER_OFF
- OPERATION_POWER_CYCLE
- OPERATION_IPMI_POWER_ON
- OPERATION_IPMI_POWER_OFF
- OPERATION_IPMI_POWER_CYCLE
- OPERATION_IPMI_RESET
- OPERATION_IPMI_SOFT_SHUTDOWN
- OPERATION_IPMI_IDENTITY_ON
- OPERATION_IPMI_IDENTITY_OFF

The OPERATION_REBOOT operation is defined by the DSView software for appliance units and is supported by the Operation Service.

The Operation Service also supports operations that are defined in the plug-in definition file for target devices and appliance units. Operations that do not define a warning message or a deck are supported.

The Operation Service allows the plug-in to obtain the list of operations that are supported by a given entity, such as a unit, in the DSView software. These operations can have been defined by the DSView software or by other plug-ins in the plug-in definition file. A user must be logged in to the DSView software using the Client Session Service. The plug-in must specify the client session ID to obtain the list of operations that the user is allowed to execute on the unit.

Get operation list

The list of operations includes operations defined by both the DSView software and the plug-ins.

- List getOperationList (Map entityMap, Map operationMap) - The method called by the plug-in to get the list of operations that a user is allowed to execute on a given entity.
  - entityMap - A map of information associated with the entity for which the user can perform an operation. A user can execute operations on units. The entity map must specify at least one of the following attributes: UNIT_OID, UNIT_NAME or UNIT_ADDRESS. See the Classification Attribute table in Classification attributes and values on page 137 for details.
  - operationMap - A map of information necessary to obtain the list of operations that can be performed on a given entity. See the Operation Map Classification Attribute table in Classification attributes and values on page 137 for details.
  - List (return value) - List of maps. A map represents the information for each operation that can be performed by a user on a given entity. If no operations are allowed, an empty list is returned. See the Operation Map Classification Attribute table in Classification attributes and values on page 137 for details.

Execute operation

Map executeOperation (Map entityMap, Map operationMap) - The method called by the plug-in to perform an operation on a given entity. The DSView software logs an event when an operation is executed successfully.

- entityMap - A map of information associated with the entity for which the operation is executed. Operations can only be executed on units. The entity map must specify at least one of the following attributes: UNIT_OID, UNIT_NAME or UNIT_ADDRESS. See the Classification Attribute table in Classification attributes and values on page 137 for details.
- operationMap - A map of information required in order to execute an operation for a given entity. See the Operation Map Classification Attribute table in Classification attributes and values on page 137 for details.
- Map (return value) - A map containing the status after executing an operation. See the Operation Status Map Classification Attribute table in Classification attributes and values on page 137 for details.

Dial-up Manager Service - getDialupManagerService()

You can access the Dial-up Manager of the DSView™ software. The Dial-up Manager provides means for setting and getting dial-up information, requesting a dial-up session and terminating a dial-up session using the following methods:

- boolean isDialupEnabled (Map unitMap) - Indicates if dial-up is enabled for the unit.
  - unitMap - A map representing the unit to query if dial-up is enabled or disabled.
  - boolean - Returns true if dial-up is enabled or false if dial-up is disabled.
- boolean isDialbackEnabled (Map unitMap) - Indicates if dial-back is enabled for the unit.
  - unitMap - A map representing the unit to query if dial-back is enabled or disabled.
  - boolean - Returns true if dial-back is enabled or false if dial-back is disabled for the unit.
- Map isDialupSessionAvailable (Map unitMap) - Indicates if a dial-up session is available for a unit.
  - unitMap - A map representing the unit to query if a dial-up session is available.
  - Map (return value) - A map representing a dial-up session, if available. If no dial-up session is available, this map is empty. If a dialup session is available, an entry in the map, UNIT_MODEM_SESSION_REMOTE_IP, has a value of the dial-up IP address for the appliance. If the main network connection is available, this map also has the entry, UNIT_ADDRESS, with a value of the main network IP address.
- void setDialConfiguration (Map unitMap, Map dialMap) - Configures the dial-up/dial-back settings in the DSView™ software:
  - unitMap - A map representing the unit to set for the dial configuration.
  - dialMap - A map representing a dial-up/dial-back configuration for the unit.

Table 4.1 Map isDialupSessionAvailable

KEY	TYPE	VALUE/COMMENTS
UNIT_MODEM_MODE	Integer	0 = dial-up disabled, 1 = dial-up enabled
UNIT_MODEM_TYPE	Integer	0 = modem, 1=isdn
UNIT_MODEM_DIALBACK_PREFIX	String	Dial-back prefix to be used
UNIT_MODEM_NUMBER1	String	Modem number 1
UNIT_MODEM_NUMBER2	String	Modem number 2
UNIT_MODEM_NUMBER3	String	Modem number 3
UNIT_MODEM_NUMBER4	String	Modem number 4
UNIT_MODEM_SESSION_IP	String	IP address for the dial up session that represents the DSView software server
UNIT_MODEM_SESSION_REMOTE_IP	String	IP address for the dial up session that represents the appliance unit
UNIT_MODEM_PPP_PASSWORD	String	ppp password
UNIT_MODEM_PPP_AUTH	Integer	0 = pap, 1 = chap
UNIT_MODEM_DIALBACK_ENABLE	Boolean	true, false
UNIT_MODEM_OTP ENABLE	Boolean	true, false
UNIT_MODEM_OTP_AUTO_REFRESH	Boolean	true, false
UNIT_MODEM_OTP_PENDING	Boolean	true, false
UNIT_MODEM_OTP_USERNAME	String	otp username
UNIT_MODEM_OTP_PASSWORD	String	otp password

- Map getDialConfiguration (Map unitMap) - Gets the dial-up/dial-back configuration for a unit. The configurations for the unit and the DSView software server that owns the unit are returned.
  - unitMap - A map of the unit that gets the dial-up/dial-back configuration.
  - Map (return value) - A map representing the dial-up/dial-back configuration for the unit.

The following table describes configurable dial-up/dial-back data that is returned from the DSView software server that owns the unit.

Table 4.2 Map getDialConfiguration (Server Settings)

KEY	TYPE	VALUE/COMMENTS
UNIT_MODEM_ANALOG_ DIALBACK_NUMBER	String	The analog dial-back number that is configured in the DSView software server that owns the unit.
UNIT_MODEM_ANALOG_ DIALBACK_ON_HOOK	Integer	The analog dial-back on-hook time. This time is in seconds.
UNIT_MODEM_DIALUP_PREFIX	String	The dial-up prefix.
UNIT_MODEM_FROM_IP_ ADDRESS	String	The dial-up IP address range start address.
UNIT_MODEM_TO_IP_ ADDRESS	String	The dial-up IP address range end address.
UNIT_MODEM_DIALUP_ INACTIVITY	Integer	The dial-up inactivity timeout. This time is in seconds and is the maximum amount of idle time for a dial-up session before it times out.
UNIT_MODEM_DIALUP_ ATTEMPT	Integer	The dial-up attempt time. This time is in seconds and is the maximum amount of time to attempt to establish a dial-up session.
UNIT_MODEM_DIALBACK_ ATTEMPT	Integer	The dial-back attempt time. This time is in seconds and is the maximum amount of time to wait for a dial-back session to be established.
UNIT_SERVER_NAME	String	The DSView software server name that owns the unit.
UNIT_SERVER_OID	Long	The DSView software server OID that owns the unit.

The following table represents the dial-up/dial-back parameters that are configured specifically for the unit that is returned. If no parameters have been configured for the unit, the key is not in the map.

Table 4.3 Map getDialConfiguration (Unit Settings)

KEY	TYPE	VALUE/COMMENTS
UNIT_MODEM_MODE	Integer	0 = dial-up disabled, 1 = dial-up enabled
UNIT_MODEM_TYPE	Integer	0 = modem 1=isdn
UNIT_MODEM_DIALBACK_PREFIX	String	Dial-back prefix to be used
UNIT_MODEM_NUMBER1	String	Modem number 1
UNIT_MODEM_NUMBER2	String	Modem number 2
UNIT_MODEM_NUMBER3	String	Modem number 3
UNIT_MODEM_NUMBER4	String	Modem number 4
UNIT_MODEM_SESSION_IP	String	IP address for the dial-up session that represents the DSView software server
UNIT_MODEM_SESSION_REMOTE_IP	String	IP address for the dial-up session that represents the appliance or the unit
UNIT_MODEM_PPP_USERNAME	String	ppp username
UNIT_MODEM_PPP_PASSWORD	String	ppp password
UNIT_MODEM_PPP_AUTH	Integer	0 = pap, 1 = chap
UNIT_MODEM_DIALBACK_ENABLE	Boolean	true, false
UNIT_MODEM_OTP ENABLE	Boolean	true, false
UNIT_MODEM_OTP_AUTO_REFRESH	Boolean	true, false
UNIT_MODEM_OTP_PENDING	Boolean	true, false
UNIT_MODEM_OTP_USERNAME	String	otp username
UNIT_MODEM_OTP_PASSWORD	String	otp password

- void establishDialConnection (Map unitMap)- To submit a request for a dial connection. This method returns immediately and a call is made to the plug-in to indicate if the connection succeeded or failed.
  - unitMap - A map representing the unit to establish a dial-up connection to.
- void terminateDialupSession (Map unitMap) - To send a request to terminate a dial-up session with a unit.
  - unitMap - A map representing the unit the termination request is for.
- Map testDialupSession (Map unitMap) - To test a modem connection for a unit.
  - unitMap - A map representing the unit to test a modem connection.
  - Map (return value) - A map representing if the dial-up test succeeded or failed. The map contains an entry for UNIT_DIALUP_STATUS with a value of UNIT_DIALUP_STATUS_SUCCEEDED or UNIT_DIALUP_STATUS_FAILED. If the test failed and a reason was provided, a map entry UNIT_DIALUP_FAIL_REASON and corresponding value is listed.
- void primaryNetworkUp (Map unitMap) - To inform the DSView software that the primary network is up for a unit.
  - unitMap - A map representing the unit for which the primary network is up.
- String getIpAddress () - Gets a virtual IP address.
  - String (return value) - A virtual IP address for the plug-in to use.
- boolean isIpAddressAvailable (String szIpAddress) - To test if an IP address is available.
  - szIpAddress - The IP address to test if available.
  - boolean (return value) - returns true if the IP address is available or false if unavailable.
- void addIpAddress (String szIpAddress) – Adds an IP address.
  - szIpAddress - The IP address to add. The IP address must be in the defined IP address range (the range defined from UNIT_MODEM_FROM_IP_ADDRESS to UNIT_MODEM_TO_ADDRESS) and must be available for use.

Email Service - getEmailService()

Plug-ins can send email through the DSView™ Email service using the following methods:

- boolean isMailServiceAvailable (Map paramMap) - Checks if the email service is available for plug-ins.
  - boolean (return value) - If the email service is available, a true value is returned. If unavailable, a false value is returned.
  - paramMap - Map containing one key-value pair:

    key: CLASSIFICATION

    value: CLASSIFICATION_EMAIL_MESSAGE
- sendEmail (Map paramMap) - Send email via the DSView EmailNotificationService.
  - paramMap - Map contains a list of key-value pair. See the Plug-in Email Service Classification Attribute table in Classification attributes and values on page 137 for details.

System information - getSystemInfo()

Plug-ins can retrieve the SystemInfo object to obtain information about the DSView™ software using the following methods:

- String getVersion () - Returns the version of DSView software the plug-in is running on.

Cached Credential Service

If permitted, plug-ins can retrieve user credentials from the credential cache. Vertiv determines on a case-by-case basis whether or not a plug-in can use this service. To use the cached credential service, plug-ins must specify the allowAccessCachedCredential attribute in the identity section of the nmm.xml file. In addition, a DSView software administrator must select *Enable user credential caching* in the *System - Global Properties - User Credentials* screen in the DSView software. The certification process for plug-ins verifies whether or not a given plug-in is allowed to use the cached credential service. Contact the technical support team for more information.

Plug-in Communications Service

A plug-in can invoke methods in another plug-in. For example, if plug-in A needs to invoke an operation or get data from an appliance supported by plug-in B, plug-in A would use this service to forward the request to the plug-in B. Plug-in A then receives a response through a listener when the request is complete. The plug-in communications service determines the NMM that the request should be forwarded to based on the controllerMap. The calling plug-in must determine the controllerMap for the operation it wishes to perform.

The service can also be used to invoke methods in plug-ins residing on another DSView software server.

The calls to this service are asynchronous. A listener is registered with the service and when requests are processed a callback is made to the listener to return the data and status.

NOTE: The use of this service is restricted and is only being permitted to be used by the Power Management plug-in. Any developer who is creating a plug-in that needs to use the plug-in communications service must first discuss the need for its use with the plug-in API team and get approval to use the service.

The plug-in communications service uses the following methods:

- register (PluginCommunicationsListener listener) - Provides the ability to register for asynchronous notifications from the plug-in communications service.
  - listener - An object that implements the PluginCommunicationsListener interface. This object is called when requests are completed.
- unregister (PluginCommunicationsListener listener) - Provides the ability to unregister a notification request listener.
  - listener - An object that implements the PluginCommunicationsListener interface. This object is called when the service is unregistered.

The following operations are performed on other plug-ins:

- void isOperationSupportedByElement (String DSViewServerID, Object requestObj, Map controllerMap, Map entityMap, String szOperationId) - Determines whether the target plug-in supports the given operation. The only expected operations are PowerOn, PowerOff and PowerCycle.

- String DSViewServerID - The GUID of the DSView™ hub or spoke server that the request should be directed to. If the value is null, the local server is used.
- Object requestObj - The object representing the request. This object is returned in the listener and allows the caller to use unique IDs for mapping requests to responses. The object is unique across the system.
- controllerMap - A map that indicates the controlling unit or object for the entity defined in the entityMap. There should be a connection path from this object to the entityObject.
- entityMap - A map that identifies the unit or object for which the operation is intended.
- szOperationId - A unique string that identifies the operation. The operation ID represents the operation type as defined by the plug-in or the DSView system.

For the information that is returned via the listener, check the isOperationSupportedByElement method in the OperationsInterface.

- void performOperationWithStatus (String DSViewServerID, Object requestObj, Map controllerMap, Map entityMap, String szOperationId) - Performs the given operation.
  - String DSViewServerID - The GUID of the DSView hub or spoke server that the request should be directed to. If the value is null, the local server is used.
  - Object requestObj - The object representing the request. This object is returned in the listener and allows the caller to use unique IDs for mapping requests to responses. The object is unique across the system.
  - Map controllerMap - Represents the controlling entity in relation to the target entity. The controlling entity and the target entity can be the same entity. For units, there is a connection path between the controlling entity and the target entity. Only units can perform related operations.
  - Map entityMap - Represents the target entity in relation to the controlling entity. The target entity identifies the entity on which the operation is performed. For units, there is a connection path between the controlling entity and the target entity. Only units can perform related operations.
  - String szOperationId - A unique string that identifies the operation. The operation ID represents the operation type as defined by the plug-in or the DSView software.

For the information that is returned via the listener, check the performOperationWithStatus method in the OperationsInterface.

ConfigurationManagement performed on other plug-ins:

- void getExtendedData (String DSViewServerID, Object requestObj, Map controllerMap, Map entityMap, Map infoMap) - Provides means for the DSView software to retrieve data from the plug-in about the target entity. The only data that is requested is version information. DSView calls this method when performing the following operations:
  - Get Versions from operations in the Unit List View and Firmware Upgraded Wizard for appliances. The data is returned through the registered listener.
  - String DSViewServerID - GUID of the DSView software server (hub or spoke that the request should be directed to. If the value is null, the local server is used.
  - Object requestObj - The object representing the request. This object is returned in the listener. It allows the caller to use unique IDs for mapping requests to responses. The object is unique across the system.

- Map controllerMap - A map representing the controlling entity in relation to the target entity. The controlling entity and the target entity can be the same entity. For units, there is a connection path between the controlling entity and the target entity.
- Map entityMap - A map representing the target entity. The target entity provides a relationship between the target entity and the controlling entity. For units, a connection path between the target entity and the controlling entity identifies the relationship. The target entity identifies the entity for which the data is requested. The target can be any entity that a plug-in recognizes, such as units, PDUs and target devices.
- Map infoMap - A map that lists the data requested by the caller. The caller can ask for multiple sets of data and each is returned in a corresponding map. The map key represents the data set and the associated object is a list representing specific data items.

Methods of the PluginCommunicationsListener:

- response (Object requestObj, String status, Map respData)
- Object requestObj - The object representing the request. The object is returned in the listener. Allows the caller to use unique IDs for mapping requests to responses. The object is unique across the system.
- String status - Overall status of the operation, success or failure and so on.
- Map respData - Map of data that is normally returned by the remote call. For example, calling get data on another plug-in returns a map.

4.5.2 Plug-in interfaces

The primary and secondary plug-in interfaces enable the plug-in to:

- Retrieve operational status of the plug-in
- Send administrative status changes (start, stop and upgrade) to the plug-in
- Request status and general information about the plug-in
- Request a reference to an object for one of the supported feature interfaces

The plug-in interface (NmmAddOnInterface) is the primary Java interface, allowing the DSView™ software to interact with a plug-in. This interface can retrieve other secondary interfaces and implementations that provide detailed data access and control functionality.

The name of the implementation class for this interface is specified in the plug-in's XML definition file (nmm.xml) - the DSView software uses this definition to construct an instance of this object by calling its empty constructor.

The plug-in interface contains a setup method. It also retrieves the following secondary interfaces that provide detailed data access and control:

- Plug-in management interface
- Fault management interface
- Configuration management interface
- Operation interface
- Tool interface
- Dial-up management interface
- Web request interface

The supportedInterfaces tag in the nmm.xml file indicates if each of these interfaces is supported. The plug-in management interface is required and must be supported. If any of the other interfaces are not supported, a null is returned.

Setup method

The setup method provides a plug-in with basic information from the DSView™ software.

- >setup (Map setupMap) - Provides setup information
  - setupMap - Map containing setup information. One of this map's main attributes is a reference to an object that implements the DSViewInterface, which provides the plug-in with access to the DSView software.

Plug-in Management interface - getNmmManagementInterface()

The plug-in Management interface manages a plug-in's operational status using the following methods:

- Map getOperationalStatus () - Retrieves the plug-in's operational status.
  - Map (return value) - Map representing the current operation status of the plug-in.
- Map start () - Starts the plug-in, which can also start any background components used by the plug-in. A prompt return is expected, accompanied by the plug-in's operational status. The DSView software is responsible for calling getOperationStatus() to determine when the plug-in successfully started.
  - Map (return value) - Map representing the current operation status of the plug-in.
- Map stop () - Stops the plug-in, which can also stop any background components used by the plug-in. A prompt return is expected, accompanied by the plug-in's operational status. The DSView software is responsible for calling getOperationStatus() to determine when the plug-in successfully stopped.
  - Map (return value) - Map representing the current operation status of the plug-in.
- Map consolidate (consolidateDataMap) - Allows the plug-in being installed to make changes while consolidating other plug-ins. A prompt return is expected, accompanied by the plug-in's operational status. The DSView software is responsible for calling getOperationStatus() to determine when the consolidation is successful. The plug-in must report a status of NMM_OPER_STATUS_CONSOLIDATING while performing this operation.
  - Map (return value) - Map representing the current operation status of the plug-in.
  - consolidateDataMap - A map containing data the plug-in can use during consolidation.
- Map upgrade (upgradeDataMap) - Updates the plug-in from the version previously installed in the DSView software. A prompt return is expected, accompanied by the plug-in's operational status. The DSView software is responsible for calling getOperationStatus() to determine when the plug-in successfully updated. The plug-in must report a status of NMM_OPER_STATUS_UPGRADING while performing this operation.
  - Map (return value) - Map representing the current operation status of the plug-in.
  - upgradeDataMap - Map containing data the plug-in can use during the upgrade, such as the last known version.

Fault Management interface - getFaultManagementInterface()

The Fault Management interface (FaultManagementInterface) retrieves fault information such as element status and element session activity, using the following methods:

- Map getStatusInformation (Map entityMap) - Retrieves status for the specified entity. The DSView software calls this method during periodic status polling and when it receives SNMP traps.
    - Map (return value) - Map representing the entity status, which includes information for the root element (unit) and connected elements.
    - entityMap - Map representing the entity. Only units and SNMP traps are supported.
- Map getStatusInformation (Map entityMap, List connList) – Retrieves status information based on the passed entity and connections. The DSView software only calls this method from the Unit Overview screen when it is retrieving status for the power connections on a target device. Plug-ins that do not support this method should respond with a DSViewException NOT_SUPPORTED error code. If the DSView software receives this exception it reverts to calling the version of getStatusInformation, which requires a single argument and no list of connections.
    - Map (return value) - Map representing the status for each connection.
    - entityMap - Map representing the entity to retrieve/determine status. Only units and SNMP traps are supported.
    - connList - A list of connection element relationship IDs for each connection for which the status should be returned.
- Map getSessionInformation (Map controllerMap, Map targetMap) - Retrieves session status for the specified entity. The DSView software calls this method during periodic status polling.
    - Map (return value) - Map representing the entity session, which includes information for the root element (unit) and connected elements.
    - controllerMap - Map representing the controlling entity. Only units are supported for this method, which uses a connection path between the controlling entity and the target entity. The controlling and target entities can be the same entity.
    - targetMap - Map representing the target entity, which is the entity for which status is being retrieved. Only units are supported for this method.

Configuration Management interface - getConfigManagementInterface()

The Configuration Management interface performs activities such as entity discovery, enrollment/de-enrollment, address configuration, file management and name management, using the following methods:

NOTE: In the Add Unit Wizard, a user can choose to discover units from a range of IP addresses. The Add Unit Wizard prompts you to choose an IPv4 or IPv6 address range. If IPv4 is selected, the List discovery method is called. If IPv6 is chosen, the List discoveryByRange method is called.

- List discovery (Map entityMap, Map discoveryInfoMap) - Discovers (interrogates) elements based on the provided entity information. Discovery is used to interrogate a unit, its connections and connected units. The DSView software calls this method during the Add Unit Wizard, Resync Wizard and when automatic topology resynchronization operations:
    - List (return value) - List of elements found during the discovery process, with their associated topology (connections and connected units). If no elements are found, an empty list is returned. Only one entry in the return List is expected and supported. This entry is a top-level controlling unit (units with a category of avctCategoryAppliance or avctCategoryController). Additional units can be connected to it.
    - entityMap - Map representing the entity, with parameters to be used for discovery. The entity map can represent:

A unit - Retrieves the updated/current topology definition for that unit.

An address of an element - Determines the element's topology at that address.

- discoveryInfoMap - Map containing information found during discovery that can be useful to other plug-ins that are performing a discovery with the same entity information. This map is expected to contain information such as protocols that are being used and type identifier values. When you are attempting to discover units from a range of addresses, the DSView™ software queries each plug-in to determine if they are able to discover a specific address.

NOTE: Refer to the AddAppliance sample plug-in for a demonstration of how to define a new appliance type and add a unit of the new type to the DSView software.

- List discoveryByRange (Map entityMap, Map discoveryInfoMap) - Discovers multiple top-level controlling units (units with a category of avctCategoryAppliance or avctCategoryController) across an IPv6 address range:
  - List (return value) - List of elements found during the discovery process, with their associated topology (connections and connected units). If no elements are found, an empty list is returned. The list can contain multiple entries and each entry is a top-level controlling unit (units with a category of avctCategoryAppliance or avctCategoryController). Additional units can be connected to it.
  - entityMap - Map representing the entity, with parameters to be used for discovery. The entity map represents a range of addresses to be discovered by the plug-in. The plug-in attempts to discover the topology of an element at each address given.
  - discoveryInfoMap - Map containing information found during discovery that can be useful to other plug-ins that are performing a discovery with the same entity information. Ignored and not used by the discoveryByRange method.
- Map enroll (Map entityMap, Map configMap)class="bold" style="font-family: 'Times New Roman';line-height: 12pt;"> - Enables the plug-in to perform configuration/setup needed for an entity that is being added to the DSView software. The DSView software calls this method when adding elements, such as during the Add Unit Wizard operation.

NOTE: This method is called within the context of a database transaction. The plug-in should not make any calls to the RepositoryService to get unit related information from this method, as database rows or tables can be locked. This could cause a lock timeout error in the database. All the information the method needs should be available in the maps that are passed in.

  - Map (return value) - Optional map that can contain the unit certificate with a private key and/or properties that can be saved for the unit. If a map is returned, it must have a classification of CLASSIFICATION_UNIT.
  - entityMap - Map representing entity to be enrolled. Only units are supported for enrollment.
  - configMap - Map containing configuration properties retrieved from the Add Unit Wizard for the unit type being added. The map's classification must be CLASSIFICATION_UNIT_PROPERTIES.
- deenroll (Map entityMap) - Enables the plug-in to perform any un-configuration needed for an entity that is being removed from the DSView™ software. The DSView software calls this element when deleting elements.

NOTE: This method is called within the context of a database transaction. The plug-in should not make any calls to the RepositoryService to get unit related information from this method, as database rows or tables can be locked. This could cause a lock timeout error in the database. All the information the method needs should be available in the maps that are passed in.

    - entityMap - Map representing the entity to be deenrolled (removed). Only units are supported for deenrollment.
- Map pushFile (Map controllerMap, Map targetMap, Map fileMap) - Pushes files to an entity via the plug-in. This is used for upgrading an entity's files. Firmware, configuration, user configuration and configuration template files are supported.
    - Map (return value) - Map representing the result of a file operation, such as status of the operation and additional steps to be performed, if necessary, such as rebooting.

      If no map is returned, the operation is considered successful and no additional steps are required.
    - controllerMap - Map representing the controlling entity (the entity pushing the file). Only units are supported for this method, which use a connection path between the controlling entity and the target entity.
    - targetMap - Map representing the target entity (where the file is being pushed). Only units are supported for this method, which use a connection path between the controlling entity and the target entity.
    - fileMap - Map representing the file being pushed.
- Map pullFile (Map controllerMap, Map targetMap, Map fileMap) - Pulls files to an entity via the plug-in. This method is used to retrieve files from an entity for storage in the DSView software database. Firmware, configuration, user configuration and configuration template files are supported.
    - Map (return value) - Map representing the retrieved file. The map provides information such as the input stream of the file, type and category.
    - controllerMap - Map representing the controlling entity. Only units are supported for this method, which use a connection path between the controlling entity and the target entity. The controlling and target entities can be the same entity.
    - targetMap - Map representing the source location of the file to be retrieved. Only units are supported for this method, which use a connection path between the controlling entity and the target entity.
    - fileMap - Map representing the file being retrieved.
- Map pushNames (Map entityMap, Map namesMap) - Pushes names to a unit for name synchronization.
    - Map (return value) - Map representing the result of the push names operation.
    - entityMap - Map representing the entity to which names are pushed. Only units are supported for this method.
    - namesMap - Map representing each of the names to be updated. In the map, the key is the element relationship ID of the element with which the unit name is associated, and the value in the map is the name of the unit in the DSView software.
- migrate (Map entityMap, Map propertiesMap) - Migrates a unit from one version to another.
    - entityMap - Map representing the entity to migrate. Only units are supported for migration.
    - propertiesMap - Map representing a set of properties to be used to control the migration.

- Map getIndicators (Map entityMap) - Retrieves indicators used by the DSView™ software.
    - entityMap - Map representing the entity from which the set of indicators is retrieved. Only units and SNMP traps are supported.
    - Map (return value) - Map representing a set of indicators from the plug-in. The DSView software uses these indicators to process topology and name changes from the unit. When Unit Status Polling and Auto Name Pull features are enabled, the plug-in is called to provide indicator information to determine whether name information is changed for the given entity. Similarly, when Unit Status Polling and Auto Topology Update features are enabled, the plug-in is called to provide indicator information to determine if the topology information is changed for the given entity. For example, the plug-in could provide a unique string, such as a timestamp or a number, each time a topology or name change has occurred. The discovery method in this interface is called by the DSView software to obtain name or topology information from the plug-in. If there are no name or topology changes, the plug-in can return an empty map.

Unit name changes in the DSView software

When the name of a unit, such as a target device or SPC, has changed in the appliance, the plug-in can provide the following information to the DSView software:

- Element Relationship - The element relationship ID that points to the unit whose name has changed.
- INDICATOR_NAME_CHANGE - A unique string, such as a number, that uniquely identifies a name change in the appliance. Other name changes return a different value. These values are used during unit status polling to determine if the name changes need to be updated in the DSView™ software.

Suppose there is an appliance unit APP that has a KVM connection to a target device named TD on port 5. Assume that you changed the name of the target device to TD-1 in the appliance. When the DSView software runs the unit status polling for the appliance unit APP, it asks the plug-in if there are name changes in the appliance. In this case, the plug-in responds with the following information:

- Element Relationship: avctRoot|5,avctKvm,Port5 (Connection Path: APP->TD)
- INDICATOR_NAME_CHANGE: “3532”

The DSView software uses this information to update the name of the target device from TD to TD-1 in the DSView software database. Later, the name of the target device is changed from TD-1 to TD-2. When the DSView software asks the plug-in for name change information, the plug-in provides the following information:

- Element Relationship: avctRoot|5,avctKvm,Port5 (Connection Path: APP->TD-1).
- INDICATOR_NAME_CHANGE: “3535”. Note that this is a different value.

The DSView software uses this information to update the name of the target device from TD-1 to TD-2 in the DSView software database. The plug-in can also specify the following information to the DSView software:

- Element Relationship - The element relationship ID that points to the unit whose name has changed.

- INDICATOR_NAME_CHANGE_OCCURRED - A boolean value that indicates a name change has occurred. This boolean value can be used during unit status polling or when the plug-in has received a trap from the appliance that indicates a name change has occurred. In this case, the plug-in informs the DSView software that the name change has taken place so that the DSView software can update the name in the database.

Suppose there is an appliance unit APP that has a KVM connection to the target device named TD on port 5. Assume that you change the name of the target device to TD-1 in the appliance. The appliance generates a trap that indicates a name change has occurred. The plug-in parses the trap information and then informs the DSView software that a name change has occurred as follows.

- Element Relationship: avctRoot|5,avctKvm,Port5 (Connection Path: APP->TD)
- INDICATOR_NAME_CHANGE_OCCURRED: True

The DSView software uses this information to update the name of the target device from TD to TD-1 in the DSView software database.

Unit topology updates in the DSView software

When the unit topology has changed in the appliance, such as a new connection to a target device, a new target device or a deleted target device, the plug-in can provide the following information to the DSView software:

- Element Relationship - The element relationship ID that points to the unit whose topology has changed.
- INDICATOR_TOPOLOGY_CHANGE - A unique string, such as a number, that uniquely identifies a topology change in the appliance. Other topology changes return a different value. These values are used during unit status polling to determine if the topology changes need to be updated in the DSView software.

Suppose there is an appliance unit APP that has a KVM connection to the target device named TD on port 5. Assume that the target device is no longer connected to port 5 on the appliance. When the DSView software runs the unit status polling for appliance APP, it asks the plug-in if there are topology changes in the appliance. In this case, the plug-in responds with the following information:

- Element Relationship: avctRoot|5,avctKvm,Port5 (Connection Path: APP->TD-1)
- INDICATOR_TOPOLOGY_CHANGE: “8861”

The DSView software uses this information to remove the target device named TD from the appliance in the DSView software database. Later, you can add the target device back to Port5. When the DSView software asks the plug-in for topology change information, the plug-in provides the following information:

- Element Relationship: avctRoot|5,avctKvm,Port5 (Connection Path: APP->TD-1)
- INDICATOR_TOPOLOGY_CHANGE: “8863”

The DSView software uses this information to add back the target device named TD to the appliance APP in the DSView software database. The plug-in can also specify the following information to the DSView software:

- Element Relationship - The element relationship ID that points to the unit whose topology has changed.

- INDICATOR_TOPOLOGY_CHANGE_OCCURRED - A boolean value that indicates a topology change has occurred. This boolean value can be used during unit status polling or when the plug-in receives a trap from the appliance that indicates a topology change has occurred. In this case, the plug-in informs the DSView™ software that the topology change has taken place so that the DSView software can update the topology in the database. In this case, the DSView software performs a Resync of the appliance where the change has occurred.

Suppose there is an appliance unit APP that has a KVM connection to the target device named TD on port 5. Assume that the target device named TD is removed from Port5 on the appliance. The appliance generates a trap that indicates a topology change has occurred. The plug-in parses the trap information and then informs the DSView software that a topology change has occurred as follows.

- Element Relationship: avctRoot|5,avctKvm,Port5 (Connection Path: APP->TD)
- INDICATOR_TOPOLOGY_CHANGE_OCCURRED: True

The DSView software uses this information to remove the target device named TD from the appliance in the DSView software database.

NOTE: The AddAppliance sample plug-in demonstrates how to define a new appliance type and how to add an appliance unit of the new type to the DSView software.

- migrate (Map entityMap, Map propertiesMap) - Migrates a unit from one version to another.
  - entityMap - Map representing the entity to migrate. Only units are supported for migration.
  - propertiesMap - Map representing a set of properties to be used to control the migration.
- Map getData (Map controllerMap, Map targetMap, List lDataKeys) - Retrieves data from the plug-in about the target entity. Only version information is requested. This method is called during the following operations: the Get Versions from operations in the Unit List view and the Firmware Upgrade Wizard for appliances.
  - Map (return value) - The map returned provides the set of data that the plug-in is able to return based on the request set of data (lDataKeys). If data keys are requested that are not supported by the plug-in, no data for that data key should be provided in the returned map.
  - controllerMap - A map representing the controlling entity in relation to the target entity. The controlling entity and the target entity can be the same entity. For units, there is a connection path between the controlling entity and the target entity. Only units are supported for retrieving data.
  - targetMap - A map representing the target entity. The target entity provides a relationship between target entity and the controlling entity. For units, a connection path between the target entity and the controlling entity identifies this relationship. The target entity identifies the entity from which data is being requested. Only units are supported for retrieving data.
  - lDataKeys - A list of data keys identifying the data that is being requested. Only version information is requested using the DATA_VERSIONS key.

- Map getExtendedData (Map controllerMap, Map targetMap, Map infoMap) - Retrieves target entity data from the plug-in including power data, environmental data and property data. This method is only called for plug-ins that support data monitoring features like environmental monitoring and power monitoring. All plug-ins must declare this method, but if data monitoring is not supported, the method should throw a DSViewException with the NOT_SUPPORTED error code. If data monitoring is not supported, this method should not be declared in the supportedInterfaces section of the nmm.xml.
    - Map (return value) - The map returned provides the set of data that the plug-in is able to return based on the requested set of data (infoMap). If the requested data keys are not supported by the plug-in, no data for that data key should be provided in the returned map. Normally, all returned values are string values representing the requested data.

      The top-level map returned is a map of maps. The CLASSIFICATION for this map is CLASSIFICATION_DATA.

      For power data, the top level map always contains the DATA_POWER_INFO key with a value of a second level map containing additional maps for each set of power data. This second level map always contains the DATA_POWER_OVERALL key with a value of a third level map containing the set of data for the overall power. Each key in the third level map identifies the data type, and the key value is the data in string format.

      For power data, when multiple phases or multiple circuits are supported, the second level map can contain the DATA_POWER_PER_PHASE or DATA_POWER_PER_CIRCUIT key. The value for each key is a third level map. The third level map contains numerical keys (in string format such as “1” and “2”) that represent the phase number or circuit number. The value for the phase number or circuit number key is a fourth level map containing the set of data for the specific phase or circuit. Each key in the fourth level map identifies the data type, and the key value is the data in string format.

      For environmental data, the top level map contains the DATA_ENVIRONMENT_INFO key with a value of a second level map. Each key in the second level map identifies the data type, and the value for the key is a third-level map. This third-level map contains numerical keys (in string format such as “1” and “2”) that represent the sensor number for that data type. The value for the sensor number key is the data in string format.

      For property data such as the model and vendor for a PDU, the top level map contains the DATA_PROPERTY_INFO key with a value of a second level map. Each key in the second level map identifies the data type, and the key value is the data in string format.

      For power and environmental data that are not readings but are instead status or sensor data, the following values should be returned by the plug-in where applicable:

      DATA_VALUE_OFFLINE - Indicates an offline condition, such as a PDU that is offline or something is turned off, such as a power socket.

      DATA_VALUE_ONLINE - Indicates an online condition, such as a PDU that is online or something is turned on, such as a power socket.

      DATA_VALUE_ERROR - Indicates an error, such as an error when communicating with a PDU.

      DATA_VALUE_ACTIVE - Indicates an active condition, such as if a smoke sensor detects smoke, a buzzer is sounding or overcurrent is detected on a PDU.

DATA_VALUE_INACTIVE - Indicates an inactive condition, such as if a sensor does not detect an alarm condition, a buzzer is not sounding or there is no overcurrent detected on a PDU.

DATA_VALUE_DISCONNECTED - Indicates a disconnected condition.

- controllerMap - A map representing the controlling entity in relation to the target entity. The controlling entity and the target entity can be the same entity. For units, there is a connection path between the controlling entity and the target entity. Only units are supported for retrieving data. The controlling unit can be any entity the plug-in recognizes including appliances and PDUs.
- targetMap - A map representing the target entity in relation to the controlling entity. The controlling entity and the target entity can be the same entity. For units, there is a connection path between the target entity and the controlling entity. The target entity identifies the entity for which data is requested. Only units are supported for retrieving data. The target unit can be any entity the plug-in recognizes including appliances, PDUs and target devices.
- infoMap - A map that lists the data requested by the caller. Each set of data is returned in a corresponding map. The map key represents the requested data set and the associated object is a list representing specific data items.

  The following sets of data are supported as keys in the infoMap:

  - DATA_POWER_INFO - A request for power information data. The value is a list that can contain one or more of the following string constants to identify the request:

    DATA_POWER_AMPS - Amps

    DATA_POWER_MAX_AMPS - Maximum current for PDU, circuit or phase

    DATA_POWER_VOLTS_EST - Voltage estimated by appliance settings

    DATA_POWER_VOLTS_READ - Voltage read by the appliance

    DATA_POWER_WATTS_EST - Power calculated by appliance settings and current

    DATA_POWER_WATTS_READ - Power calculated by read values of current and voltage

    DATA_POWER_PDU_STATUS

    DATA_POWER_OVERCURRENT

    DATA_POWER_TURN_ON_INTERVAL

    DATA_POWER_BUZZER

  - DATA_ENVIRONMENT_INFO - A request for environmental information. The value is a list that can contain one or more of the following string constants to identify the request:

    DATA_ENV_TEMP_C - Temperature (Celsius)

    DATA_ENV_HUMIDITY - Humidity

    DATA_ENV_WATER_SENSOR - If water is present

    DATA_ENV_DOOR_SENSOR - If a door is opened

    DATA_ENV_AIRFLOW - Airflow reading

    DATA_ENV_SECURITY - Security sensor

DATA_ENV_SMOKE - Smoke detector

DATA_ENV_DRY_CONTACT - Possibly the same as WATER_SENSOR

DATA_ENV_VOLTAGE_SENSOR - Voltage sensor

- DATA_CURRENT_VALUES - Indicates if the request is for current or historical values. The value is Boolean. If the value is true, current values are requested. If the value is false, historical values are requested. If the key is not present, no historical values are requested.

NOTE: Requests for historical data are only permitted for the power management plug-in, so for all other plug-ins the value must be true.

- DATA_HISTORICAL_START and DATA_HISTORICAL_END (power management plug-in only) - Used to specify the date range for which historical values are requested. The value for each key is a long that represents GMT time.
- DATA_INTERVAL (power management plug-in only) - Indicates if the historical data date range is hourly, daily, weekly or monthly. The value is one of the following string constants:

  DATA_INTERVAL_HOURLY

  DATA_INTERVAL_DAILY

  DATA_INTERVAL_WEEKLY

  DATA_INTERVAL_MONTHLY

- DATA_PROPERTY_INFO - Provided when asking for properties on a target entity that is a PDU device or an environmental device. A plug-in that supports PDUs or environmental devices must return this information. The plug-in stores this data in the extended properties of the unit when the unit is first discovered and added. The plug-in updates this information in the database if it changes. When requested, the plug-in must retrieve the information from the extended properties and not directly from the device so the call returns quickly. The value is a list that can contain one or more of the following string constants to identify the request:

  DATA_PROPERTY_MODEL

  DATA_PROPERTY_VENDOR

  DATA_PROPERTY_NUM_PHASES

  DATA_PROPERTY_NUM_CIRCUITS

  DATA_PROPERTY_NUM_OUTLETS_PER_CIRCUIT

  DATA_PROPERTY_NUM_OUTLETS

Operation interface - getOperationInterface()

The Operation interface can be used to define custom operations performed by a plug-in. These operations are displayed in either the Operations Menu in the Units View screen or the Appliance Overview screen, or both. The plug-in can optionally define a warning message that is displayed by the DSView software prior to performing an operation. The supported interface section of the nmm.xml file should only define either isOperationSupported or isOperationSupportedByElement. If both methods are defined, the DSView software only calls the isOperationSupportedByElement method.

In addition to the operations that are defined by the plug-in, the plug-in must implement the functionality for the following operations that are defined by the DSView software:

- OPERATION_REBOOT - Invoked from:
  - The Operations menu in a Units View when you select an appliance
  - An appliance overview screen
- OPERATION_POWER_ON, OPERATION_POWER_OFF, OPERATION_POWER_CYCLE - Invoked from:
  - The Operations menu when you select a target device connected to a power connection
  - A target device overview screen
  - The Add Task Wizard when you create and run a task to control the power on target devices

The Operation interface uses the following methods:

- boolean isOperatonSupported (String szOperationId) - Indicates if an operation is supported.
  - boolean (return value) - Returns true if the operation is supported, false if it is not.
  - szOperationId - Unique string that identifies the operation. The operation ID represents the operation type defined in the plug-in or the DSView™ software.
- boolean isOperationSupportedByElement (String szOperationId, Map entityMap) - Indicates if an operation is supported. The DSView ™software gives precedence to this method over the isOperationSupported method.
  - boolean (return value) - The returned boolean indicates if an operation is supported.
  - szOperationId - A unique string that identifies the operation. The operation ID represents the operation type as defined by the plug-in or the DSView software.
  - entityMap - A map that identifies the entity for which the call is made. The entity can be any appliance type, including appliances, PDUs and target devices.

Some system-defined operations are executed completely by the DSView software and the plug-in is not called to execute the performOperation method. However, the plug-in must still report whether or not the operation is supported. For example, OPERATION_DELETE is executed by the DSView software and most plug-ins support this operation for all known entities. In this case, the plug-in returns a value of true. In a case where a plug-in does not support the delete operation, the plug-in returns a value of false.

Each plug-in must report on the following system defined operations:

- OPERATION_DELETE - Typically, the plug-in returns true.
- OPERATION_RESYNC - If the value is true, the Reysnc Unit Wizard is supported for the entity. If the value is false, the Reysnc Unit Wizard is not supported and the operation does not appear in the DSView software for the entity.

- OPERATION_DATA_MONITORING - Indicates whether or not data monitoring is supported for the entity. The plug-in returns true if power or environmental monitoring is supported for the entity.
- OPERATION_ENVIRONMENTAL_MONITORING - The plug-in returns true if environmental monitoring is supported for the entity.
- OPERATION_POWER_MONITORING - The plug-in returns true if power monitoring is supported for the entity.
- OPERATION_POWER_CONTROL - The plug-in returns true if any kind of power control (on, off, cycle) is supported for the entity.
- OPERATION_POWER_ON - The plug-in returns true if it can turn on the entity.
- OPERATION_POWER_OFF - The plug-in returns true if it can turn off the entity.
- OPERATION_POWER_CYCLE - The plug-in returns true if it can power cycle the entity.
- OPERATION_IPMI_POWER_ON - The plug-in returns true if it can turn on the entity through IPMI.
- OPERATION_IPMI_POWER_OFF - The plug-in returns true if it can turn off the entity through IPMI.
- OPERATION_IPMI_POWER_CYCLE - The plug-in returns true if it can power cycle the entity through IPMI.

NOTE: For OPERATION_POWER_ON, OPERATION_POWER_OFF, OPERATION_POWER_CYCLE, OPERATION_IPMI_POWER_ON, OPERATION_IPMI_POWER_OFF and OPERATION_IPMI_POWER_CYCLE, the plug-in is required to execute the operation through the performOperation method of the OperationInterface.

- performOperation (Map controllerMap, Map targetMap, String szOperatonId) - Performs a specified operation.
    - controllerMap - A map representing the controlling entity. The controlling entity and the target entity can be the same entity. For units, there is a connection path between the controlling entity and the target entity.
    - targetMap - A map representing the target entity. For units, a connection path between the target entity and the controlling entity identifies this relationship. The target entity identifies the entity that the operation is being performed on including PDUs (where applicable) for power control operations. Only units are supported for performing operations.
    - szOperationId - A unique string identifying the operation type defined in the plug-in or the DSView software.
- Map performOperationWithStatus (Map controllerMap, Map targetMap, Map infoMap, String szOperationId) - Performs the specified operation and returns the status. This method is only called for the power management plug-in. Other plug-ins must declare this method, but should throw a DSViewException with the NOT_SUPPORTED error code. This method should not be declared in the supportedInterfaces section of the nmm.xml.
    - Map (return value) - Indicates the success or failure of the operation and provides additional data that can be needed by the caller.
    - controllerMap - A map representing the controlling entity in relation to the target entity. The controlling entity and the target entity can be the same entity. For units, there is a connection path between the controlling entity and the target entity. Only units are supported for performing operations.

- targetMap - A map representing the target entity. The target entity provides a relationship between the target entity and the controlling entity. For units, a connection path between the target entity and the controlling entity identifies this relationship. The target entity identifies the entity on which the operation is performed. Only units are supported for performing operations.
- infoMap - A map providing data that is required for the operation.
- szOperationId - A unique string identifying the operation. The operation ID represents the operation type as defined by the plug-in or the DSView software.

NOTE: The performOperationWithStatus method can be used to perform functions from the Operations menu that require user-entered data, or for backing up plug-in specific data. If you are using this method, contact Vertiv SDK Support for more information.

Operations for Unit Elements

When an operation is to be performed on an appliance unit, both the controller map and the target map parameters provide information related to the appliance unit. The operation is displayed on both the Operations Menu in the Units View and the Appliance Overview screens.

NOTE: The AddApplianceOperation sample plug-in demonstrates how to define a new operation type and how to add an operation type that is associated with an appliance unit in the DSView software.

The plug-in can optionally specify a deck that displays after you select the operation from either the Operations Menu in the Units View or the Appliance Overview screen.

NOTE: The ApplianceOperationWithDeck sample plug-in demonstrates how to define and display a deck that is associated with an existing operation for an appliance unit. The deck can be either a single screen or a wizard.

Operations for Connection Elements

When an operation is to be performed on a connection to a target device unit, the controller map provides information related to the appliance unit, the target map provides information related to the target device unit and both maps provide information about the connection path between the appliance and the target device units. The operation is displayed on the Operations Menu in the Units View.

NOTE: The AddTargetDeviceOperation sample plug-in demonstrates how to define a new operation type and how to add an operation type that is associated with a target device unit in the DSView software.

The plug-in can optionally specify a deck that is displayed after you select the operation. Refer to Deck controller on page 95 and Operations and Tools on page 174 for details on how to define a deck, such as Operation Action or Operation Wizard, for an operation.

NOTE: The TargetDeviceOperationWithDeck sample plug-in shows how to define and display a deck that is associated with an existing operation for a target device unit. The deck can be either a single screen or a wizard.

Built-in operations

The plug-in can also support the following operations defined by the DSView software:

- OPERATION_REBOOT: This operation applies to appliance units and can be invoked in the following ways:

    - From the Operations Menu in the Units View when you select an appliance unit
    - From the Appliance Overview screen if you have the right to reboot the appliance
- OPERATION_POWER_ON, OPERATION_POWER_OFF, OPERATION_POWER_CYCLE: This operation applies for power connections to target device units and can be invoked in the following ways:
    - From the Operations Menu when you select a target device connected to a power connection
    - From the target device overview screen
    - From the Add Task Wizard when you create and run a task to control the power on target devices

Tool interface - getToolInterface()

The Tool interface can be used to retrieve a list of redirection objects that represent items such as wizards and viewers. It can also be used to perform any necessary setup before launching a tool. The Tool interface uses the following methods:

- List getToolList (Map controllerMap, Map targetMap) - List of redirection objects for tools supported by the target unit, based on the controller unit.
    - List (return value) - List of maps containing information about tools supported by the target unit. Maps contain the following entries:

      CLASSIFICATION - Must be CLASSIFICATION_NMM_TOOL_DATA

      TOOL_REDIRECT - Value of the redirection object

      TOOL_KEY - String used to look up a <tool> entry in the nmm.xml definition file by the unique tool identifier

    - controllerMap - Map representing the controlling unit, which can be a unit connection path. The connection path provides the relationship between the controlling unit and the target unit. The controlling and target units can be the same unit.
    - targetMap - Map representing the target unit, which can be a unit connection path. The set of tools returned are for this unit, based on its relationship to the controller unit.
- Boolean isToolSupported (String szToolType, Map entityMap) - Determines if the plug-in supports the specified tool.
    - boolean (return value) - The boolean returned indicates if the plug-in supports the given tool type.
    - szToolType - A unique string identifying the tool type. The tool type can be defined by the DSView software or by the plug-in.
    - entityMap - A map identifying the unit object.
- Map performToolSetup (Map controllerMap, Map targetMap, Map propertiesMap) - Performs the necessary environment setup to allow the tool to be started. An example is authorizing a session and returning a session certification and other needed information.
    - Map (return value) - Map of properties needed to launch the tool. The map's content depends on what was requested in the original call to perform ToolSetup(). The only required map element is CLASSIFICATION, which must be CLASSIFICATION_TOOL_SETUP_REPLY. The only supported request is preauthorization. The following table lists the map response elements.

- controllerMap - Map representing the controlling unit, which can be a unit connection path. The connection path provides the relationship between the controlling unit and the target unit. The controlling and target units can be the same unit.
- targetMap - Map representing the target unit, which can be a unit connection path.
- propertiesMap - Map representing the type of setup to be performed and any other information needed for the setup. The map must contain the following entries:

  CLASSIFICATION - Must be CLASSIFICATION_TOOL_SETUP

  TOOL_SETUP_ACTION - specifies the action

  The only supported action is TOOL_SETUP_PREAUTH, which requires the following entries:

  TOOL_SESSION_MODE

  TOOL_SESSION_TYPE

  TOOL_SETUP_USER - name of the user requesting the tool be started

  TOOL_SETUP_PREEMPTION - effective preemption level of the user (java.lang.Integer)

  TOOL_SETUP_RIGHTS - rights of the user; must be one of RIGHTS_USER, RIGHTS_USER_ADMIN or RIGHTS_CONTROLLER_ADMIN

The following table lists the elements of a preauthorization response.

Table 4.4 Preauthorization Response Elements

KEY	CONDITION	VALUE
SESSION_STATUS	Always	One of the following values can be appended to the SESSION_STATUS_ key: • NOERROR • BUSY • BLOCKED • UNAVAILABLE • UPGRADING • TOOMANYSESSIONS
SESSION_TYPE	When the status is NOERROR and the requested type is NATIVE.	One of the session types in Plug-in interfaces on page 74
SESSION_USER	When the status is BUSY or BLOCKED.	String username
SESSION_ PREEMPTION_ LEVEL	When the status is BUSY or BLOCKED.	Integer
SESSION_STATE_ SHAREABLE	When the status is BUSY.	Boolean
SESSION_STATE_ STEALTHABLE	When the status is BUSY.	Boolean
TOOL_SETUP_ REPLY_CERT	When the status is NOERROR	java.security.cert.X509Certificate
TOOL_SETUP_ REPLY_KEYPAIR	When the status is NOERROR	java.security.KeyPair **NOTE: The private key in the pair is used to sign the java.security.cert.X509Certificate.**
SESSION_ CHANNEL	When the status is NOERROR.	Map of maps **NOTE: The outer map represents the channel. The inner map contains information about the channels.**

The following table lists the SESSION_CHANNEL map elements of a preauthorization response.

Table 4.5 SESSION_CHANNEL Map Elements of a Preauthorization Response

MAP LEVEL	KEY	VALUE
Outer	SESSION_CHANNEL_SERIAL	Inner map of information about a serial channel
	SESSION_CHANNEL_VIDEO	Inner map of information about a video channel
	SESSION_CHANNEL_VIRTUALMEDIA	Inner map of information about a virtual media channel
	SESSION_CHANNEL_KM	Inner map of information about a keyboard/mouse channel
Inner	SESSION_CHANNEL_PORT	Integer port number
	SESSION_CHANNEL_ENCRYPTED	Boolean

The following table lists the valid values for the TOOL_SESSION_MODE entry.

Table 4.6 TOOL_SESSION_MODE Valid Values

CONSTANT	REQUESTS
SESSION_MODE_NORMAL	Normal session
SESSION_MODE_STEALTH	Stealth session
SESSION_MODE_EXCLUSIVE	Exclusive session
SESSION_MODE_PREEMPT_NON_EXCLUSIVE	Preempt a non-exclusive session
SESSION_MODE_PREEMPT_EXCLUSIVE	Preempt an exclusive session
SESSION_MODE_SHARE	Shared session
SESSION_MODE_PASSIVE_SHARE	Passively shared session
SESSION_MODE_SCAN	Scan session
SESSION_MODE_FORCE_SHARE	Force a shared session
SESSION_MODE_PASSIVE_FORCE_SHARE	Force a passively shared session

The following table lists the valid values for the TOOL_SESSION_TYPE entry in the Map performToolSetup method.

Table 4.7 TOOL_SESSION_TYPE Valid Values

CONSTANT	REQUESTS
SESSION_TYPE_ NATIVE	Controller's native session type. The response from the controller must indicate the type of session that was established.
SESSION_TYPE_KVM	KVM session.
SESSION_TYPE_ SERIAL	Serial session.
SESSION_TYPE_VM	Virtual media session.

Dial-up Interface (DialupManagementInterface)

The DialupManagementInterface is a secondary Java interface of the plug-in. This interface provides means to send information to the plug-in for dial-up activities such as a session disconnect, dial request results and a dial-back phone number change. The DialupManagementInterface uses the following methods:

- void disconnectModemSession(Map controllerMap, Map targetMap, Map sessionMap) - The DSView software calls this method to inform the plug-in that a dial-up session is terminating.
  - controllerMap - A map representing the controlling entity. The controlling entity provides a relationship between the controlling entity and the target entity. For units, a connection path between the controlling entity and the target entity identifies this relationship.The controlling entity and the target entity can be the same entity.
  - targetMap - A map representing the target entity. The target entity provides a relationship between the target entity and the controlling entity. For units, a connection path between the target entity and the controlling entity identifies this relationship. The target entity identifies the entity for which session status information is being retrieved.
  - sessionMap - A map representing a dial-up session. The classification for this map is CLASSIFICATION_MODEM_SESSION.

If a dial-up session is in progress, the map contains an entry for UNIT_MODEM_SESSION_REMOTE_IP containing the dial-up IP address that is being disconnected. The map also contains an entry for UNIT_DIALUP_STATUS with a status of UNIT_DIALUP_STATUS_SUCCEEDED.

If a dial-up session is not in progress, the map contains an entry for UNIT_DIALUP_STATUS with a value of UNIT_DIALUP_STATUS_FAILED. The map also contains an entry for UNIT_DIALUP_FAIL_REASON, which provides the failure reason.

- void dialRequestComplete(Map controllerMap, Map targetMap, Map sessionMap) - The DSView software calls this method to inform the plug-in that the request for a modem connection is complete.
  - controllerMap - A map representing the controlling entity. The controlling entity provides a relationship between the controlling entity and the target entity. For units, a connection path between the controlling entity and the target entity identifies this relationship. The controlling entity and the target entity can be the same entity.
  - targetMap - A map representing the target entity. The target entity provides a relationship between the target entity and the controlling entity. For units, a connection path between the target entity and the controlling entity identifies this relationship. The target entity identifies the entity for which session status information is being retrieved.
  - sessionMap - A map representing a dial-up session. The classification for this map is CLASSIFICATION_MODEM_SESSION. The map contains an entry for UNIT_DIALUP_STATUS with a value of UNIT_DIALUP_STATUS_SUCCEEDED or UNIT_DIALUP_STATUS_FAILED to indicate if the modem connection was successful or if it failed. If it failed, there is an entry for UNIT_DIALUP_FAIL_REASON with a value of the reason why the request failed. If the dial-up request was successful, there is an entry for UNIT_MODEM_SESSION_REMOTE_IP with a value of the modem session IP address.
- void dialbackNumberChanged (Map dialbackMap, List unitList) – The DSView software calls this method to inform the plug-in that the dial-back phone number is changed in the DSView software. This dial-back phone number is configured per the DSView software server.
  - dialbackMap - A map representing the changed dial-back phone number configured in the DSView software. The classification of this map is CLASSIFICATION_DSVIEW. The map contains the entry UNIT_MODEM_ANALOG_DIALBACK_NUMBER for the analog dial-back phone number with a value of the new analog dial-back phone number.
  - unitList - A list of maps representing the units that are owned by the DSView software server and require the dial-back phone number(s) changed.
- boolean isNetworkUp (Map unitMap) - The DSView software calls this method to test if the primary network is up for a specified unit.
  - unitMap - A map representing the unit to test the network connectivity.

Web Request Interface (WebRequestInterface)

The WebRequestInterface is a secondary Java interface of the plug-in. The WebRequestInterface provides the means to redirect both the HttpServletRequest and HttpServletResponse jetties to the plug-in. The plug-in defines the supported URLs and protocols in the webRequestFilter element in the nmm.xml. This method allows the plug-in to process and handle HttpServletRequest and set HttpServletRespones independently of the DSView software server.

The plug-in is responsible for handling the request, including writing out the response. Once this method is called, the DSView software does not provide any additional processing of the request.

The WebRequestInterface uses the following methods:

- >void webRequest(Map requestMap) - The DSView software calls this method to transfer web request parameters to the plug-in.
  - requestMap - A map containing the HttpServletRequest and the HttpServletResponse.

4.5.3 DSView™ Listener interface

The DSView™ Listener interface (DSViewListener) enables the DSView™ software to notify plug-ins asynchronously, based on notification requests registered by the plug-in.

The DSView Listener interface uses the following method:

- notification (Map notificationMap) - Called when data/conditions match one or more of the registered notification requests.
  - notificationMap - Map containing an object based on the type of data/condition found that matches the notification request. This can be an SNMP trap, Syslog event, server property change or email notification.

Unit notification map

If the notification is for a unit, the following map is passed to the notification method of the DSView Listener interface.

Table 4.8 Unit Notification Map

ATTRIBUTE	VALUE TYPE	VALUE SET/RESTRICTIONS	REQUIREMENTS
CLASSIFICATION	String	CLASSIFICATION_NOTIFICATION_UNIT_FILTER	Required
UNIT_OID	Long	The OID of the unit.	Required
UNIT	Map	The entity map of the unit. This attribute is provided for update and insert events, but not for delete events.	Optional
EVENT_SEQUENCE_NUMBER	Long	The sequence number of this event, which can be used to sort events in order of occurrence.	Required
EVENT_NAME	String	The notification event type. The valid values are: • EVENT_NOTIFICATION_UPDATE • EVENT_NOTIFICATION_INSERT • EVENT_NOTIFICATION_DELETE	Required

Unit notification extended properties map

If the notification is for a unit whose extended properties have been modified, the following map is passed to the notification method of the DSView Listener interface.

Table 4.9 Unit Notification Extended Properties Map

ATTRIBUTE	VALUE TYPE	VALUE SET/RESTRICTIONS	REQUIREMENTS
CLASSIFICATION	String	CLASSIFICATION_NOTIFICATION_UNIT_ EXTENDED_PROPERITES_FILTER	Required
UNIT_OID	Long	The OID of the unit.	Required
UNIT	Map	The entity map for the unit.	Required
EVENT_SEQUENCE_ NUMBER	Long	The sequence number of this event, which can be used to sort events in order of occurrence.	Required
EVENT_NAME	String	The notification event type. The valid values are: EVENT_NOTIFICATION_UPDATE	Required

Connection notification map

If the notification is for a connection, the following map is passed to the notification method for the DSView™ Listener interface.

Table 4.10 Connection Notification Map

ATTRIBUTE	VALUE TYPE	VALUE SET/RESTRICTIONS	REQUIREMENTS
CLASSIFICATION	String	CLASSIFICATION_NOTIFICATION_UNIT_ CONNECTION_FILTER	Required
CONNECTION_ OID	Long	The OID of the connection.	Required
CONNECTION_ AEND	Map	The entity map for the unit that is connected to the AEND of the connection, if any. This attribute is provided for update and insert events if a unit exists for the AEND of the connection, but not for delete events.	Optional
CONNECTION_ ZEND	Map	The entity map for the unit that is connected to the ZEND of the connection, if any. This attribute is provided for update and insert events if a unit exists for the ZEND of the connection, but not for delete events.	Optional
EVENT_ SEQUENCE_ NUMBER	Long	The sequence number of this event, which can be used to sort events in order of occurrence.	Required
EVENT_NAME	String	The notification event type. The valid values are: • EVENT_NOTIFICATION_UPDATE • EVENT_NOTIFICATION_INSERT • EVENT_NOTIFICATION_DELETE	Required

4.5.4 Add-on Context interface

The Add-on Context interface enables a plug-in to access context information. The NmmAddonContext object can be retrieved using the NmmAddonContextFactory plug-in:

- NmmAddOnContext nmmCtx = NmmAddonContextFactory
- GetNmmAddonContextFactory().getNmmAddonContext()

The Add-on Context interface uses the following methods:

- String getUsername () - Retrieves the username associated with the current operation being performed that is causing the plug-in to perform some processing. This value is updated each time the plug-in is requested to perform some operation.
  - String (return value) - Username
- Map getDomainMap () - Retrieves a map containing domain level information that is populated by the plug-ins of the domain, which are not tied to a specific session.
  - Map (return value) - Map containing domain level information. This map can be modified by the plug-in to hold information based on the plug-in's domain.
- Map getSessionMap () - Retrieves a map containing session level information that is populated by the plug-in.
  - Map (return value) - Map containing session level information.
- DSViewInterface getDSView () - Retrieves the DSViewInterface associated with the plug-in.
  - DSViewInterface (return value) - DSViewInterface.
- long getUserOid () - Retrieves the OID of the user associated with the current operation that requires processing by the plug-in. This value is updated each time the plug-in is requested to perform an operation.
  - long (return value) - The OID of the user associated with the current operation.
- long getUserActiveZoneOid () - Retrieves the OID of the active zone that is associated with the current user.
  - long (return value) - The OID of the active zone that is associated with the current user.

4.5.5 Deck controller

The deck controller (DeckController) is an object identified by the plug-in for each defined deck. These deck controllers provide implementation-specific functionality such as reading/writing from a device. They can also provide additional validation and transitions beyond those specified in the plug-in configuration file.

The deck controller has the following properties:

- Controller data - Provides access to the deck data being sent to/from the user. This property cannot be set by the deck controller.
- Stateless - Indicates whether or not the controller is stateless. A stateless controller is used only once. Stateful controllers are preserved across multiple accesses. Most deck controllers should be stateless. Wizard controllers must be stateful.

A deck controller has three stages:

- Initialization - Called before the first execution of a deck controller. By default, nothing is performed; custom actions can be performed by overriding initialize().
- Execution - Called after initialization. Any user-submitted data is available to the deck controller.
- Population - Called after execution. This finalizes the display of data on a card. At this point, any card transitions should have been completed. Custom population of the card data can be performed by overriding populateView().

The deck controller methods are:

- void initialize()
- void execute()
- void populateCard()
- boolean isStateless()

Custom execution

Custom execution can be performed by either overriding the execute() method or providing an alternative execution method. When the execution phase begins, the method to execute is chosen by first searching for a method tied to the selected control and/or the current card. The search order and nomenclature for such methods is:

1. public void executeCONTROLIDOnCARDID()
2. public void executeCONTROLID()
3. public void executeOnCARDID()
4. public void execute()

The CONTROLID and CARDID follow Java bean property naming conventions. For a control with the ID "save" and a card with the ID "properties", the search order is:

1. public void executeSaveOnProperties()
2. public void executeSave()
3. public void executeOnProperties()
4. public void execute()

In all cases, the methods must be declared as public, return void and throw no exceptions. Otherwise, they are not interpreted as valid execute functions. The default execute() function is shown in the following example.

Example: Default execute() function

```
public void execute()
{
 ControllerData cData = getControllerData();
 cData.performDefaultCardValidation();
 if ( cData.getErrors().isEmpty() )
 {
 cData.performDefaultCardTransition();
 }
}
```

For a WizardController, a custom execute method is defined for the back control. In this case, field validation is not needed as shown in the following example.

Example: executeBack() method

```
public void executeBack()
{
 ControllerData cData = getControllerData();
 cData.performDefaultViewTransition();
}
```

To hide form prompts and section labels:

1. Find the form in which the prompt to be hidden is located in the nmm.xml file as shown in the following example.

Example: Hidden prompt in the nmm.xml file

```
<!-- Settings KVM -->
 <form formId="settingsKVM">
 <section label="form.settingsKVM.section.inactivity.label">
 <prompt property="enabledInactivity"
 label="form.settingsKVM.checkbox.enabledInactivity.label"
 classification="checkbox">
 <prompt property="inactivityTimeout"
 label="form.settingsKVM.list.inactivityTimeout.label"
 classification="text"/>
 </prompt>
 </section>
```

2. Find the deck controller java class associated with this form in the nmm.xml file as shown in the following example.

Example: nmm.xml file containing the deck controller java class

```
<!-- Settings KVM -->
 <deck deckId="settingsKVM" label="deck.settingsKVM.label"
 classification="standard"
 controller="com.avocent.dsrPlugin.controllers.Settings
 KVMController">
```

3. Open the deck controller java file and add the hidePrompt() method as shown in the following example.

Example: hidePrompt() method

```
/**
 * Hides a prompt inside a panel form.
 *
 * @param fd the FormData object for the panel
 * @param szProperty the name of the property on the form.
 */
 private void hidePrompt( FormData fd, String szProperty )
 {
 String szHidePrompt = "HIDE_PROMPT";
 Map mapOptions = fd.getPropertyOptionsMap( szProperty );
 if ( mapOptions != null )
 {
 // Hides prompts with options
 mapOptions.put( szHidePrompt, szHidePrompt );
 }
 else
 {
 // Hides prompts without options
 fd.setPropertyValue( szProperty, szHidePrompt );
}
}
```

4. Open the deck controller java file and add the hidePromptAndSectionLabel() method as shown in the following example.

Example: hidePromptAndSectionLabel() method

```
/**
 * Hides a prompt and the section label inside a panel
 *
 * @param fd the FormData object for the panel
 * @param szProperty the name of the property on the form.
 */
 protected void hidePromptAndSectionLabel( FormData fd, String szProperty )
 {
 String szHidePrompt = "HIDE_PROMPT_AND_SECTION_LABEL";
 Map mapOptions = fd.getPropertyOptionsMap( szProperty );
 if ( mapOptions != null )
 {
 mapOptions.put( szHidePrompt, szHidePrompt );
}
 else
 {
 fd.setPropertyValue( szProperty, szHidePrompt );
 }
}
```

5. The following code hides the prompt “inactivityTimeout” in the populateCard() method as shown in the following example.

Example: Code to hide inactivityTimeout in the populateCard() method

```
public void populateCard()
 {
 ControllerData cd = getControllerData();
 FormData fd = cd.getCardForm();
 // Hide Prompt
 hidePrompt( fd, "inactivityTimeout" );
 }
```

6. The following code hides the prompt “inactivityTimeout” and the label for the section that contains the prompt in the populateCard() method as shown in the following example.

Example: Code to hide inactivityTimeout and the label for the section containing the prompt in the populateCard() method

```
public void populateCard()
 {
 ControllerData cd = getControllerData();
 FormData fd = cd.getCardForm();
 // Hide Prompt and Label for the section that contains the prompt
 hidePromptAndSectionLabel( fd, "inactivityTimeout" );
 }
```

4.5.6 Controller data interface

The Controller Data (ControllerData) interface provides the deck controller with access to deck and plug-in data. It also provides access to default transition and validation mechanisms based on plug-in configuration files.

The controller data object’s properties are:

- Errors - Errors to be displayed to the user. No errors are entered before deck controller execution.
- Entity maps - Target and controller entity maps are available, which represent the unit targeted for an operation and the unit through which the operation is occurring. In some cases, these are the same unit; in other cases, there is no target or controller.
- FormData - The data being received from and sent to the user, which can be accessed from the context of the entire deck or the current card.
- DeckId - The ID of the deck. This matches the DeckId in the plug-in configuration file.
- CardId - The ID of the current card. This matches the CardId in the plug-in configuration file. The controller can change this to transition from one card to another.
- ControlId - The Id of the control selected by a user. This matches the ControlId of the plug-in configuration file or the next/back function in wizards.
- Aux Data Map - Map with auxiliary data. Typically, this is an empty map, but can contain data for specific user interface extensions.
- Target resource list - List of strings that indicate a targeted subresource. The list is automatically populated with keys of drilled-down tables.

In addition to the properties, the controller data interface provides the following functions:

- void performDefaultCardValidation() - Validates the data in CardData using the validation rules in the plug-in’s validation.xml definition file.

- void performDefaultCard Transition() - Attempts to transition between views using the transition information in the plug-in's nmm.xml definition file.
- void redirect (Redirection redirection)

4.5.7 Form data

A form data object enables data to be transferred from the user to the deck controller and back to the user. The DSView software reads and writes values and options between the user and the deck controller. All access uses property names; these properties are declared in the form of declarations in the nmm.xml definition file.

For each property, the following data is available: value(s), option map and multivaluedness.

- Value(s) - May be accessed as single values using the getValue() and setValue() methods, or as lists using the getValues() and setValues() methods.

  Both methods can be used regardless of their multivalued nature. However, a non-multivalued property cannot have a list with more than one item set to it using the setValues() method; otherwise, an exception is thrown. Also, if getValue() is called and multiple values are set on a property, only one is returned. For checkboxes, the value should always be either true or false.

- Option map - Stores the user-displayed options for properties requiring selection, such as list, membership and tables. The map stores the possible property values as the map's keys and the display strings as the map's values. For a table, the map's values must be a list of strings representing the table's columns. The option map is always empty when a deck controller instance is first called. The user only submits the selected keys.
- Multivaluedness - Indicates whether or not the property can contain multiple values. Examples of multivalued properties are multilist, membership, ordered membership and table.

The classification for each prompt that is defined in the nmm.xml file can be accessed using the following methods:

- String getPromptClassification (String szProperty) - This method returns the classification for the given property that is associated with a prompt in the form data.
- String getPromptClassificationsMap() - This method returns a map that contains the classification for each prompt that is defined in the form data. The key in the map is the name of the property that is associated with a prompt. The value in the map is the classification for the prompt.

4.5.8 Add Unit wizard extensions

The Add Unit wizard is extensible by declaring decks with specific names. The information collected by these wizard fragments is submitted to the discovery and enrollment interfaces of the plug-in as key/value pairs in the submitted map.

The following wizard fragments can be created:

- avctAddUnitDiscoverExtension - Executed immediately after the user has selected a unit type to add in the Add Unit Wizard. It obtains all information needed for discovery, including the IP address (if needed).
- avctAddUnitEnrollExtension - Executed after the unit is discovered, but prior to its addition to the DSView™ software database.

In addition to the specific required DeckId, the decks must have two special declared cards which are used to navigate through the extension back into the Add Unit wizard. These cards must have the following form IDs:

- avctEntryForm (this must be used on the default card)
- avctExitForm

4.5.9 Redirection

Redirection is an object provided by the ToolsInterface.getToolList() that identifies how a tool is presented and launched from the DSView™ software. This section describes the supported redirection object types.

Browser redirection

The BrowserRedirection object launches a browser screen to a URL. The constructors are:

- BrowserRedirection (String szUrl)

  szUrl - URL address of the tool

- BrowserRedirection (String szUrl, Map params)

  szUrl - URL address of the tool

  params - Map of keys/values that is added as parameters to the URL. Each of the keys and values must be a string.

- BrowserRedirection (String szUrl, Map params, Map browserOptions)

  szUrl - URL address of the tool

  params - Map of keys/values that is added as parameters to the URL.

  browserOptions - Map of keys/values used to customize the browser screen, such as height and width

Deck redirection

The DeckRedirection object launches a specific deck, such as a wizard. The constructor is:

- DeckRedirection (String deckId)
  - deckId - Identifier of the deck to be used as the tool

Generic redirection

The AvctGenericRedirection object launches a built-in viewer included in the viewer launching process using a command line startup. Generic redirection is supported only on win32 platforms. The following viewers are included in the launching process for win32:

- testviewer.exe - Opens a screen that shows command line arguments used to start the viewer. This is useful for development and debugging.
- vpclient.exe - Cyclades ACS viewer.

The constructor of AvtGenericRedirection is:

- AvctGenericRedirection (String viewerCommand, String cmdLineArgs)

- viewerCommand - Native command to launch the viewer. To preserve cross-platform compatibility, do not use filename extensions or paths.
- cmdLineArgs - Arguments to be supplied to the command. If there are no arguments, pass an empty string.

Avocent redirection

The Avocent redirection objects launch a built-in serial viewer. The constructors of these redirection object types have no parameters:

- AvctSerialRedirection - Launches a built-in Avocent serial viewer
- AvctKvmRedirection - Launches a built-in Avocent video viewer
- AvctTelnetRedirection - Launches a built-in Avocent serial viewer

4.5.10 Exceptions in the Java interfaces

All calls must have time limits and handle all throwables/exceptions in a timely and appropriate manner.

Each of the Java interfaces can generate exceptions when exceptional conditions are encountered. These types of conditions can arise when unexpected data is received or unexpected results are encountered.

Other than Java system exceptions, only DSViewException or exceptions extending DSViewException are generated. The DSViewException can provide one of the following:

- Exception message (not localized)
- Resource key and attributes for building a localized message
- Original exception (if this exception is wrapping another exception)
- Error code (constants defined in DSViewException)

4.6 XML Interface

XML definition files define the plug-in's meta-data. Sample definition files are available on the Vertiv web site. The following sections describe the elements and attributes in the definition files. An element or attribute can be:

- Required - The element/attribute is required. Required entries can be specified only if their containing object is specified.
- Optional - The element/attribute is optional.
- Restricted - The use/definition of the element/attribute is restricted. Not all definition files can define elements/attributes.

Once a plug-in is officially released, certain meta-data defined by the plug-in cannot change in newer versions of the plug-in. This meta-data is permanently registered in the DSView software database; thus, if this registered data are to change in a future plug-in version, unexpected results could occur in the DSView software. The plug-in certification process includes steps to ensure that new plug-in versions comply with this rule.

The type, classification and scope attributes for meta-data defined in the following nmm.xml sections cannot be modified or removed from the plug-in once released:

- Units
- Connections

- Events
- File types
- Tools
- Operations
- Views

4.6.1 Plug-in definition file (nmm.xml)

The plug-in definition file (nmm.xml) defines the functionality and capabilities of the plug-in. This definition file is required.

Identity section

The identity section of the nmm.xml definition file contains plug-in identification information.

The nmmId and domain values help with name space issues when different plug-ins use the same values for types they are defining. The scope setting for the element, operation and viewer definitions determines which values are used.

- For local scope, both nmmId and domain values are used to make the definition unique.
- For domain scope, only the domain value is used to make the definition unique within the domain.
- For global or system scope, neither nmmId nor domain are used.

Table 4.11 Identity Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
identity	N/A	Required	Set of attributes that identify the plug-in.
	nmmId	Required	(Defined by the plug-in developer) Identifier that is unique within the specified domain. Can contain up to 64 ASCII characters. Cannot contain a pipe (|) or a space.
	domain	Required	Domain name associated with the plug-in developer. Must be in standard dot notation using ASCII characters; multilingual domain names are allowed. Cannot contain a pipe (|) or a space.
	licenseId	Optional	(Defined by Vertiv for each plug-in that must be licensed). Integer license identifier.
	vendor	Required	Resource key to retrieve the plug-in vendor.
	name	Required	Resource key to retrieve the plug-in name.
	description	Required	Resource key to retrieve the plug-in description.
	version	Required	Plug-in version string, containing up to 32 characters - a series of four values, separated by periods (for example, 1.2.3.4). The most significant version component is the leftmost value and the values become less significant as you go to the right.
	implementation	Required	Name of the Java class that supports the NmmAddOnInterface.
	priority	Restricted	Priority level of the plug-in. The lower the value, the higher the priority. Only specific plug-ins can specify this value.
	locales	Required	Locales are supported by the plug-in, specified as a series of locale identifiers and separated by commas. A locale identifier has the format: <language>_<country>_<variant> This is the same format that identifies resource files in Java. <language> - ISO Language Code - lowercase two-letter code defined by ISO-639 (http://www.ics.uci.edu/pub/ietf/hettp/related/iso639.txt). <country> - ISO Country Code - uppercase two-letter code defined by ISO-3166 (http://www.iso.org/iso/en/prods-services/iso3166ma/index.html). <variant> - Vendor and browser-specific. Some variants can be composed of multiple variants; each variant is separated by an underscore.
	allowAccess CachedCredential	Optional	Indicates if a plug-in is permitted to use the cached credential service. The value is true or false.

Requirements section

The requirements section of the nmm.xml definition file contains the plug-in's requirements and dependencies.

Table 4.12 Requirements Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
requirements	N/A	Required	Set of requirements that must be met for the plug-in to operate in the DSView™ and Rack Power Manager software environment.
requirement	N/A	Optional	Specification of a requirement.
	classification	Required	Type of requirement that must be met. The following classification is supported: avctMinimumVersion - Minimum version of the DSView™ and Rack Power Manager software.
	value	Required	Value associated with the classification: minimum version of the DSView™ and Rack Power Manager software.

Web Request Filter Section

The webRequestFilter section of the nmm.xml definition file provides the ability to define URLs and/or paths the plug-in supports. The DSView™ and Rack Power Manager software defines a root URL that must be used in all web requests directed to the plug-in. The URLs defined in the webRequestFilter should be appended to the DSView™ and Rack Power Manager software root URL. The root URL is defined in WEB_REQUEST_URL_ROOT.

The following values are used as an example:

- The root URL for the DSView™ and Rack Power Manager software as defined in WEB_REQUEST_URL_ROOT has the value "/dsview/plugins/webrequest/".
- A plug-in defines a URL pattern of "myproduct/*" for the HTTP protocol.
- The address of the DSView software server that the web request is going to be sent to is 192.168.0.100 and the port is 443.

Given this information, the device sending the web request to the DSView™ and Rack Power Manager software for redirection to a plug-in would make an HTTP request using the following URL:

http://192.168.0.100:443/dsview/plugins/webrequest/myproduct/somepage.

Table 4.13 Web Request Filter Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
webRequestFilter	N/A	Optional	Identifies the plug-in's supported URLs and protocols.
urls	N/A	Optional	URLs the plug-in supports.
url	N/A	Required	A URL the plug-in supports.
	names	Required	Path of the URL the plug-in supports.
protocols	N/A	Optional	Protocols the plug-in supports.
protocol	N/A	Required	A protocol the plug-in supports.
	name	Required	A protocol's name. Must be HTTP or HTTPs. If HTTP, the data must be encrypted by the plug-in and the client application.

Consolidation section

In previous releases of the DSView™ and Rack Power Manager software, a separate plug-in must be created for each supported unit type. Previously released plug-ins supporting a single unit type can be replaced with a new plug-in supporting multiple unit types. The new plug-in is referred to as a "consolidated" plug-in.

The Consolidation section of the nmm.xml file defines the old plug-ins that are replaced by the new consolidated plug-in. When a new plug-in that defines a consolidation section is added, the DSView™ and Rack Power Manager software perform the necessary updates to the system to replace the separate plug-ins with the new plug-in. The old separate plug-ins are automatically deleted from the DSView™ and Rack Power Manager software once the new consolidated plug-in is successfully added.

Table 4.14 Consolidation Service of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
Consolidation	N/A	Optional	Defines the plug-ins that are being consolidated (replaced with a new plug-in).
Consolidate	N/A	Optional	Specification of a plug-in to be consolidated. Plug-ins are uniquely identified by their nmmID and domain.
	nmmid	Required	Identifies the nmmID that corresponds to the plug-in to be overridden by this plug-in.
	domain	Required	Identifies the domain that corresponds to the plug-in to be overridden by this plug-in.

Help section

The help section of the nmm.xml definition file specifies the plug-in's support for custom help screens.

Table 4.15 Help Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
help	N/A	Required	Identifies the plug-in's help.
	key	Required	Identifier of the help within the domain. If multiple plug-ins share/use the same help, this key should be the same for each plug-in. For plug-ins that have the same domain and the same help key, one help is presented, based on the version. The last version of the help is displayed.
	description	Required	Resource key for the help description.
	label	Required	Resource key for the link to launch the help.
	index	Required	Relative root path for the help.
	version	Required	Version number for the help.

Supported interfaces section

The supported interfaces section of the nmm.xml definition file defines the interfaces and methods supported by the plug-in.

Table 4.16 Supported Interfaces Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
supportedInterfaces	N/A	Required	Identifies the plug-in's supported interfaces and methods.
	timeout	Optional	Overrides the system-defined timeout value for loading interfaces and the response time from methods.
interface	N/A	Optional	Identifies an interface the plug-in supports.
	name	Required	Interface name.
	timeout	Optional	Overrides the timeout in supportedInterfaces (if defined). Maximum amount of time (in milliseconds) the DSView™ and Rack Power Manager software waits for the interface to load.
method		Optional	Method in the interface.
	name	Required	Method name.
	timeout	Optional	Overrides the timeout in interface and supportedInterfaces (if defined). Maximum number of milliseconds the DSView™ and Rack Power Manager software waits for a response from the method.

Navigations section

The navigations section of the nmm.xml file identifies sets of navigation definitions for units and connection types.

Table 4.17 Navigations Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
navigations	N/A	Required	Sets of navigation definitions for elements (units and connection types) in the definition file.
navigationSet	N/A	Optional	Set of navigations. Each set has one or more navigation entries, which have well-known keys.
	navSetId	Required	Unique identifier of the navigation set.
navigation	N/A	Optional	Navigation definition for a specific navigation key - a series of node entries that define the navigation.
	key	Required	Well-known navigation definition key that indicates where the navigation is to be used. The following keys are supported: • properties - Navigation nodes for the properties section of the unit navigation tree. • management - Navigation nodes for the management section of the unit navigation tree. • drilldown - Navigation that is presented when a user drills down from an entry in a table.
node	N/A	Optional	Navigation node and optionally an associated screen that is presented for this node. Nodes can be nested to define a navigation tree.
	nodeId	Required	Unique identifier of the navigation node.
	label	Required	Resource key to retrieve the navigation tree text.
	deckId	Optional	ID of the deck to be presented when this node is selected. If this attribute is omitted, navigation automatically drills down the navigation tree until a node is found with a deck ID.

Forms section

The forms section of the nmm.xml file contains form definitions.

Table 4.18 Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION
forms	N/A	Required	Set of form definitions.
form	N/A	Optional	Specification of a form definition.
	formId	Required	Unique form identifier.
	deckId	Optional	Deck identifier.
section	N/A	Optional	Defines a section within a form.
	label	Required	Resource key to retrieve the section text.
prompt	N/A	Optional	Prompt within a section.
	property	Required	Unique property associated with the prompt value.
	label	Optional	Resource key to retrieve the prompt text.
	value	Optional	Value of the prompt. Valid only and required for radio prompts. Only one radio button can have a value of 'on'; all others must have a value of off.
	classification	Required	Classification of the prompt to be presented. Valid values are: • notice - Displays a note with static or dynamic text. For static text, a label attribute (no property) is required. For dynamic text, a property attribute (no label) is required. • combo - Displays a drop-down list. • list - Displays a single select list. • multilist - Displays a multi-select list. • membership - Displays an unorderable membership control. • orderedmembership - Displays an orderable membership control. • text area - Displays a multi-line text edit box. • pre - Displays a preformatted read-only value.
(continued)			• dynamic label - Displays an uneditable (read-only) value. The value is assigned at runtime. Label attributes with an empty value and property attributes are required. • label - Displays an uneditable (read-only) value. Property and label attributes are required. • warning - Displays an image with an exclamation point followed by an uneditable value (read-only). Property and label attributes are required. • text - Displays a single-line text edit box. Property and label attributes are required. • checkbox - Displays a checkbox. Property and label attributes are required. There can be other prompts that are enabled/disabled based on the checkbox being enabled/disabled. • password - Displays a password edit box. Property and label attributes are required. • radiobox - Displays a collection of radio prompts. Property and label attributes are required. This prompt must contain only child radio prompts.

Table 4.18 Section of the nmm.xml Definition File (continued)

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION
			• numberspinner - Numerical spin box. Property, label, minimum and maximum attributes are required. Only numerical values are permitted. • radio - Displays a radio button. Property and label attributes are required. There can be other prompts that are enabled/disabled based on the radio being enabled or disabled.
table	N/A	N/A	Table definition in the form.
	property	Required	Unique property associated with the rows of the table.
	rowIndex	Required	Property that identifies the index of each row.
	deckId	Optional	Deck to present when drilling down on an entry in the table.
	customizeView	Optional	Identifies the name of the customizable view for the table. All names must be unique. If this attribute is provided, the DSView™ and Rack Power Manager software provide the Customize link above the table, which allows you to customize which columns are displayed and in what order they are displayed. Use the fixed attribute and the hiddenByDefault attribute of the column element to further control how customize works for individual columns.
controls	N/A	Optional	Set of controls associated with the table.
control	N/A	Optional	Definition of a control for the table.
	controlId	Required	Unique control identifier.
	label	Required	Resource key to retrieve the control (button) text.
	classification	N/A	Control classification. Valid values are: • deck - Action to launch/go to a specific deck. • action - Operation (action).
	deckId	Optional	Deck ID that is displayed when the control is activated. This attribute is required when the classification attribute is a deck.
	rightsRule	Optional	Rights required to allow specific control to the user. The format is an expression of rights identifiers, operators ('and', 'or' and 'not') and grouping symbols '( )'. For example: avctDomainUnitView and (not avctDomainUnitEdit or avctDomainUnitGroupEdit)
	minSelected	Optional	Minimum number of selections required for this control. The default value is 0.
	maxSelected	Optional	Maximum number of selections required for this control. The default is -1, which indicates any number of selections.
	scope	Optional	Scope of the control.
	confirmMsgLabel	Optional	A resource key that specifies the text to be displayed in action confirmation messages. If the key is not specified, a confirmation message is not displayed. The classification of this control is "action".
columns	N/A	Required	Set of table columns.
column	N/A	Required	Column definition.

Table 4.18 Section of the nmm.xml Definition File (continued)

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION
property	N/A	Required	Unique column property.
	label	Required	Resource key to retrieve the column header text.
	fixed	Optional	Identifies whether or not the column is fixed or customizable. Valid when the customizeView element is defined for a table. Valid values are: • true - Column is customizable • false (default value) - Column in not customizable
	hiddenByDefault	Optional	Identifies whether or not the column is hidden by default. Valid when the customizeView element is defined for a table. Valid values are: • true - The column is hidden unless you customize the table and enable the column. • false (default value) - The column appears by default. You can disable the column by customizing the table.
	deckId	Optional	Indicates the column is a link to the deck that is displayed when the URL is activated.

Decks section

A deck is a high level view of a user interface entry point. It contains its type information plus a series of one or more cards.

A card is a single view for the overall deck. For example, a single step in a wizard is a card. A card has an associated form, as well as controls and transition information. Transition information is used during card processing to advance you to a different card in the same deck depending on your input.

A form is a set of fields. The set of fields contain data you use to input data into the form, as well as fields that are filled by the custom deck controller to present to you depending on your input.

There are five deck types (classifications):

- Standard - A standard deck is the most common type. It is tied to a specific navigation location within the DSView™ and Rack Power Manager software. Upon rendering, it displays the navigation tree with its navigation location highlighted. Any card level controls on a standard deck are rendered in the button bar area directly under the deck title bar.
- Action - An action deck is not tied to any navigation location. Upon rendering, it does not display any navigation information. Any card level controls on an action deck are rendered in the button bar area directly under the deck title bar.
- Wizard - A wizard deck is tied to the navigation location from which it was reached. Upon rendering, it displays the navigation tree with its navigation location highlighted. Card level controls on a wizard deck are not supported. Instead, Next, Back, Cancel and Finish controls are displayed, based on the transition information specified for the card.
- Operation Action - An operation action deck has a Submit control that allows you to submit an operation that was previously selected from either the Operations Menu or the Appliance Overview screen in the DSView™ and Rack Power Manager software.
- Operation Wizard - An operation wizard deck is not tied to any navigation location. This deck has a Submit control in the last screen of the wizard that allows you to submit an operation that was previously selected from either the Operations Menu or the Appliance Overview screen in the DSView™ and Rack Power Manager software.

- Flex Screen - A deck that provides the ability to play an Adobe Flash movie provided by the plug-in. Flex programs are written using Adobe's ActionScript and MXML languages. This type of deck gives the plug-in developer the ability to create a richer user interface. The deck must define a card that refers to a form that identifies the Flash movie to play. Examples of the required property definitions used to define the card are as follows:
  - <prompt property="movie" classification="hidden" value="<movieName>"/>
  - <prompt property="flashvars" classification="hidden" value="<vars>"/>
  - <prompt property="title" classification="hidden" label="<resource key>"/>

  In the example, the property definitions are as follows:

  - <movieName> is replaced with the name of the Flash movie to play
  - <vars> is replaced with the variables needed for the movie
  - <resource key> is the resource key for the title to display on the JSP screen that the system provides to load the movie.

  All Flash movies should be placed in the swf directory of the plug-in image.

  The Flex application must be written to communicate with the plug-in using web requests over HTTPS. The plug-in must implement the WebRequestInterface to receive and reply to requests from the Flex application.

NOTE: The use of the Flex Screen deck type is limited at this time to the power management plug-in.

Table 4.19 Wizard Controls

CONTROL	DISPLAYED	ENABLED
Next	If next transition exists.	Always.
Back	Always.	If back transition exists.
Cancel	Always.	If next transition exists.
Finish	If next transition does not exist.	Always.
Submit	For an Operation Action deck, this control is always displayed. For an Operation Wizard deck, this control is displayed in the final screen of the wizard only if the final screen of the wizard is not the Failure screen.	Always.

Table 4.20 Decks Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
decks	N/A	Required	Set of deck definitions.
deck	N/A	Optional	Deck definition.
	deckId	Required	Unique deck identifier. Some deck IDs are reserved for extending existing DSView™ and Rack Power Manager software decks: • avctAddWizard - Extends the Add Wizard. The classification attribute must be a wizard. • avctResyncWizard - Extends the Resync Wizard. The classification attribute must be a wizard. • Decks for a wizard that is being extended have an entry card (cardId = entry, formId = avctEntryForm) and an exit card (cardId = exit, formId = avctExitForm).
	label	Required	Resource key to retrieve the text for the deck title and header. This label can be dynamically changed using the ControllerData interface.
	classification	Required	Deck classification. Valid values are: • standard - Represents a standard deck, which is tied to a specific navigation location. • wizard - Deck used to represent a wizard, which provides flow between a series of cards. • action - Similar to a standard deck, except it is not tied to any navigation location.
(continued)			Deck classification. Valid values are: • standard - Represents a standard deck, which is tied to a specific navigation location. • wizard - Deck used to represent a wizard, which provides flow between a series of cards. • action - Similar to a standard deck, except it is not tied to any navigation location. • operationAction - Similar to an action deck. The deck has a Submit control to allow you to submit an operation from either the Operations Menu or the Appliance Overview screen in the DSView software. When the operation is submitted, the performOperation method of the Operations Interface is called. • operationWizard - Similar to a wizard deck except that it is not tied to any navigation location. The deck has a Submit control in the last screen of the wizard to allow you to submit an operation from either the Operations Menu or the Appliance Overview screen in the DSView software. When the operation is submitted, the performOperation method of the Operations Interface is called. • flexpage - Represents a Flex deck, which is tied to an Adobe Flash movie created using AcitionScript and MXML languages. Allows a Flash movie to be launched for the deck. Requires the deck to define a card that refers to a form that defines the Flash movie to play.
	controller	Optional	Name of the Java class that extends DeckController.
	rightsRule	Optional	Rights required for the deck. The format is an expression of rights identifiers, operators (‘and’, ‘or’ and ‘not’) and grouping symbols ‘( )’. For example, avctDomainUnitView and (not avctDomainUnitEdit or avctDomainUnitGroupEdit).

Table 4.20 Decks Section of the nmm.xml Definition File (continued)

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
card	N/A	Required	Definition of a card in the deck. At least one card must be defined for a deck. Decks for a wizard that is being extended have an entry card (cardId = entry, formId = avctEntryForm) and an exit card (cardId = exit, formId = avctExitForm).
	cardId	Required	Unique card identifier.
	formId	Required	ID of a form (defined in the forms section) associated with the card.
	default	Optional	Indicates this card is the default card to be processed. One card must be identified as the default card.
	label	Optional	Resource key to represent this card.
controls	N/A	Optional	Set of controls (buttons) to be presented with the card. See the Forms Section in Classification attributes and values on page 137 for more information.
transitions	N/A	Optional	Transitions associated with a card in a wizard deck. This is required for wizard decks.
transition	N/A	Optional	Transition associated with the card - contains one or more destination (dest) elements that determine the destination card, based on a matching dest element. A final dest element is expected at the end to handle situations when none of the previous dest elements are met.
	controlId	Required	Identifier of the control (button) associated with this transition.
dest	N/A	Required	Conditional destination screen within the deck. The condition (if specified) for each of the destination entries for a navigation action are evaluated in the order specified. The first destination with a condition that evaluates to true is used. The cardId attribute for that destination is the card presented.
	cardId	Required	Destination card presented when a match occurs.
	if	Optional	Conditional statement to determine if this destination should be used. If this attribute is omitted, the condition is treated as being met (true).

Images section

The images section of the nmm.xml definition file defines sets of image definitions for defined elements, viewers and actions.

Table 4.21 Images Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
images	N/A	Required	Sets of image definitions for defined elements, viewers and actions.
imageSet	N/A	Optional	Definition of an image set. Each set of images has one or more image entries. The image entries have well-known keys that help determine which image should be used in different locations in the DSView™ and Rack Power Manager software.
	imageSetId	Required	Unique image set identifier.
image	N/A	Optional	Image definition for an image key. Well-known image definition key that indicates where the image is to be used. Valid values are: • launchStandard - Standard image to be used when launching a viewer or initiating an action/operation. This must be a 22 x 22 pixel gif image. • listStandard - Standard image to be used when presenting an image in a list, such as a unit list. This must be a 19 x 19 pixel gif image. • toolStandard - Standard image to be used when displaying a tool in the Appliance Overview screen. This must be 32 x 32 pixel gif image. (This can be used to display operations defined by the plug-in. See the Operations section on page 115.)
	key	Required	
	name	Required	Name of the image file, which is located in the images directory of the plug-in jar file.

Operations section

The operations section of the nmm.xml definition file define the set of operations supported by the plug-in. System-defined operations do not need to be specified if they are used by elements supported by the plug-in.

Table 4.22 Operations Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
operations	N/A	Required	Set of operations supported by the plug-in.
operation	N/A	Optional	Operation specification. The operation can be associated with the following element classifications: • Unit - An operation can be associated with an appliance unit. The operation is specified in the capabilities section of the element. See the Elements Section in Classification attributes and values on page 137 for details. • Connection - An operation can be associated with a connection to a target device unit. The operation is specified in the capabilities section of the element.
	operationId	Required	Unique operation identifier.
	label	Required	Resource key that specifies the text to be displayed for this operation.
	type	Required	Unique operation type.
	rightsRule	Required	Right required for the operation. The format is an expression of rights identifiers, operators ('and', 'or' and 'not') and grouping symbols '( )'. For example, avctDomainUnitView and (not avctDomainUnitEdit or avctDomainUnitGroupEdit) See Access Rights on page 183 for more information.
	scope	Optional	Scope of the operation definition. Valid values are: • avctLocal - Definition is unique to this plug-in only, not shared. • avctDomain - Definition is unique to the domain of this plug-in, shared with other plug-ins with the same domain • avctGlobal - Definition is shared with all plug-ins. If this attribute is omitted, the default is avctLocal.
	imageSetId	Optional	Image set that is displayed for the operation. If this attribute is omitted, the operation is not displayed in the Appliance Overview screen in the DSView™ and Rack Power Manager software.
	confirmMsgLabel	Optional	Resource key that specifies the text to be displayed to the user in a confirmation dialog before the operation is performed. If this attribute is omitted, there is no confirmation dialog before the operation is performed.
	deckId	Optional	Unique deck identifier.

Tools section

The tools section of the nmm.xml definition file defines a set of tools the plug-in supports. System defined tools do not need to be specified if they are used by elements supported by the plug-in.

Table 4.23 Tools Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
tools	N/A	Required	Set of tools the plug-in supports.
tool	N/A	Optional	Tool specification.
	toolId	Required	Unique tool identifier.
	label	Required	Resource key to retrieve the tool text.
	type	Required	Tool type.
	classification	Required	Tool classification. Valid values are: • standard - Standard tools, generally displayed in Unit Overview screens. • viewer - Viewer tools generally displayed in Unit Overview screens and in the Action field or pull-down menu.
	scope	Optional	Scope of the tool definition. Valid values are: • avctLocal - Definition is unique to this plug-in only, not shared. • avctDomain - Definition is unique to the domain of this plug-in, shared with other plug-ins with the same domain • avctGlobal - Definition is shared with all plug-ins. If this attribute is omitted, the default is avctLocal.
	imageSetId	Optional	Image set that defines the images for the tool.

Elements section

The elements section of the nmm.xml definition file covers unit types, connection types, event categories, event IDs and file types supported by the plug-in.

Resource keys for elements of type avctEventId include the following elements:

- label.name - Event name resource key
- label.long - Event's long description resource key
- label.short - Event's short description resource key

Table 4.24 Elements Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
elements	N/A	Required	Set of element definitions.
element	N/A	Required	Element definition. At least one element definition must have the classification avctUnit (only one avctUnit definition is allowed) that identifies the unit managed by this plug-in. The plug-in can provide support for multiple units. To support multiple units, the plug-in should define an element with a classification of type “avctUnit” for each unit being supported. The “avctUnit” definitions identify the elements that are in this plug-in.

Table 4.24 Elements Section of the nmm.xml Definition File (continued)

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
(continued)	elementId	Required	Unique element identifier.
	classification	Required	Element type. Valid values are: • avctUnit - Unit • avctConnection - Connection • avctEventCategory - Event category • avctEventId - Event ID • avctFile - File • avctUnitCustomizedColumn - Defines a custom column to be displayed in a Units View screen in the DSView™ and Rack Power Manager software.
	scope	Optional	Scope of the element definition. Valid values are: • avctLocal - Definition is unique to this plug-in only, not shared. • avctDomain - Definition is unique to the domain of this plug-in, shared with other plug-ins with the same domain. • avctGlobal - Definition is shared with all plug-ins. • avctSystem - Definition is defined by the system, shared by all. If this attribute is omitted, the default value is avctLocal. Some elements support only certain scope values: • avctUnit - Supports only avctLocal, avctGlobal or avctSystem. For avctGlobal, the owner attribute must also be specified. • avctConnection - Supports only avctLocal, avctDomain and avctSystem. • avctEventCategory - Supports only avctLocal and avctDomain. • avctEventId - Supports only avctLocal and avctDomain. • avctFile - Supports avctLocal, avctDomain and avctGlobal. For avctFile definitions of category avctFirmware, the scope must be avctGlobal. • avctUnitCustomizedColumn - Supports avctGlobal.
	owner	Optional	Identifies the plug-in as the primary or secondary owner for the element. This attribute is only used (and required) for elements of classification avctUnit when the scope is avctGlobal. The valid values are: • avctOwnerPrimary • avctOwnerSecondary
	type	Required	Element type. Can contain up to 64 ASCII characters. Cannot contain a pipe (|), comma (,) or space.

Table 4.24 Elements Section of the nmm.xml Definition File (continued)

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
(continued)	label	Required	Resource key to retrieve the element type text for this element.
	family	Optional	Family with which the element is associated. This attribute is valid (and required) only for elements of classification avctFile and category avctFirmware. This value must match one of the Product Family Codes specified in the Avocent Flash Upgrade File Header Format.
	oem	Optional	OEM with which the unit is associated. This attribute is valid (and required) only for elements of classification avctFile and category avctFirmware. This value must match one of the OEM Type Codes in the Avocent Flash Upgrade File Header Format.
	navigationSetId	Optional	Navigation set that defines the element navigation. This attribute is valid only for elements that are in the avctUnit and avctConnection classifications.
	imageSetId	Optional	Image set that defines the images for the element. This attribute is valid only for elements that are in the avctUnit and avctConnection classifications.
	severity	Optional	Element severity. This attribute is valid only for elements that are in the avctEventId classification. Valid values are: • avctNonRecoverable • avctCritical • avctNonCritical • avctOk • avctInformation • avctMonitor
	category	Optional	Element category. This attribute is valid only for elements that are in the avctFile or avctUnit classifications. Valid values for the avctFile classification are: • avctFirmware - Firmware • avctUserConfig - User configuration • avctConfig - Configuration • avctConfigTemplate - Configuration template The following set of category definitions are supported for classification avctUnit (for backward compatibility, if the category attribute is not provided then avctCategoryAppliance is assumed): • avctCategoryAppliance • avctCategoryTargetDevice • avctCategoryCascadeDevice • avctCategorySoftware • avctCategoryChassis • avctCategoryCard
(continued)	(continued)	(continued)	• avctCategorySensor • avctCategoryController Multiple comma separated category values can be specified for units. See Category attribute for units on page 127 for more information.
	relationshipType	Optional	Identifies the relationship type for an element. This attribute is only valid for elements of classification avctConnection. Valid values are:

Table 4.24 Elements Section of the nmm.xml Definition File (continued)

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
			• avctRunsOn (avctRuns) • avctConnectedTo (avctConnectsTo) • avctActivatedBy (avctActivates) • avctInstalledOn (avctContains) • avctSupportedBy (avctSupports) • avctManagedBy (avctManages) The default value avctConnectedTo is assigned if the relationshipType attribute is not specified.
capabilities	N/A	Optional	Sets of plug-in capability definitions. This element is valid only for elements of classification avctUnit and avctConnection.
capability	N/A	Optional	Element capability.
	classification	Required	Classification. Valid values are operation and viewer.
	type	Required	Capability type.
	scope	Required	Scope of the capability.
	sessionSuport	Optional	Defines a supported session mode. The DSView™ and Rack Power Manager software only recognize value avctExclusive, which indicates the KVM connection represented by this capability supports exclusive KVM sessions.
relationships	N/A	Optional	Set of relationship definitions of the plug-in. This element is valid only for elements of classification avctEventId.
relationship	N/A	Optional	Relationship of the element to other elements.
	classification	Required	Classification of the element with which this element has a relationship. The valid value is: child - Element is a child of the element
	type	Required	Relationship type.
	scope	Required	Scope of the relationship.
supportedFileTypes	N/A	Optional	Set of filetypes supported by the element. This element is valid only for elements of classification avctUnit.
fileType	N/A	Optional	Filetype supported by the element.
	scope	Optional	Scope of the file type.
	type	Required	Type of file supported by the element.
properties	N/A	Optional	Properties for the unit type defined in the plug-in.
property	N/A	Optional	Property definition.
	key	Required	Property key.
	value	Required	Property value.
rights	N/A	Optional	Set of rights for the element.
right	N/A	Optional	Identifies a right.
	name	Required	Right name.

Additional information for classification avctEventId

Elements in the avctEventId classification require a specific format for the resource keys for the label attribute. The specific resource key elements for the label attribute are as follows:

- label.name - event name resource key.
- label.long - event's long description resource key.
- label.short - event's short description resource key.
- trap.<enterprise>.<specificId>.long - event's long description resource key for traps, where the trap enterprise is <enterprise> and the trap specific id is <specificId>. The overwriting of built-in trap definitions is not supported for plug-ins.

Additional information for the classification avctUnitCustomized column

The avctUnitCustomizedColumn classification provides the means for a plug-in to define its own custom column to appear in the DSView™ software Units tab. The column is hidden by default but you can enable the column by clicking the *Customize* link. When a plug-in defines a column to appear in the Units View, it must also provide the data to appear in the column for each unit for which the column is supported. The data is stored in the extended properties of the unit. The key for the data in the extended properties of the unit must match the value specified in the type attribute of the avctUnitCustomizedColumn element definition.

The plug-in returns the data for the columns it defines in the extended properties of the discovery information, so that valid data appears for the units when they are first added to the DSView™ and Rack Power Manager software. The plug-in should also update the extended properties of a unit with any changes that occur on the data by calling the repository service updateElement method with the updated extended properties for the unit. The following element attributes can be updated:

- label attribute of element - Elements of classification avctUnitCustomizedColumn must have a specific format for the resource key for the label attribute. The format must be column.customized.<name>.label where <name> is replaced with a plug-in defined name that matches the name used in the type attribute.
- type attribute of element - Elements of classification avctUnitCustomizedColumn need to have a specific format for the value of the type attribute. The format must be column.customized.<name> where <name> is replaced with a plug-in defined name that matches the name used in the label attribute.

Example: Defining custom columns in an nmm.xml

```
<element elementId="dsrOscar" classification="avctUnitCustomizedColumn"
 scope="avctGlobal" type="column.customized.dsrOscar"
 label="column.customized.dsrOscar.label">
</element>

 <element elementId="dsrDhcp" classification="avctUnitCustomizedColumn"
 scope="avctGlobal" type="column.customized.dsrDhcp"
 label="column.customized.dsrDhcp.label" >
 </element>
```

4.6.2 Views section

The views section of the definition file defines a list of views that are supported by the plug-in. This feature enables plug-ins to add new views to the DSView software. Similar to the system-defined Appliance and Target Devices views, plug-ins can define new views to display the defined plug-in unit types. New views appear between the Target Devices and Sites navigation node. Each new unit type defined for that view has its own navigation node under the new view as well as an All navigation node to display all the units for that view.

Table 4.25 Views Section of the nmm.xml Definition File

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION/RULES
views	N/A	Required	Set of view definitions.
view	N/A	Required	Definition of a view.
	viewId	Required	Identifier of the view; all view Ids must be unique.
	scope	Optional	Identifies the scope of the view definition. Valid values are: • avctLocal - Definition is unique to this plug-in only, not shared. • avctDomain - Definition is unique to the domain of this plug-in, shared with other plug-ins with the same domain. • avctGlobal - Definition is shared with all plug-ins. • avctSystem - Definition is defined by the system, shared by all. Supports only avctApplianceView or avctTargetDeviceView. If this attribute is omitted, the default value is avctLocal.
(continued)	type	Required	Identifies the type of the view. Can contain up to 64 ASCII characters. Cannot contain a pipe (|), comma (,), period (.) or space. Certain types are system defined, so an element that appears under a corresponding system view must specify the appropriate type. The following types must have a scope of avctSystem:
	(continued)	(continued)	• avctApplianceView - Defines system appliance view. • avctTargetDeviceView - Defines system target device view. • Any element of avctUnit not defined with a view determines its view based on the element categories. If no category is defined for that element, the default category is defaulted to avctApplianceView.
	label	Required	A resource key that can be used to retrieve the view type text to be presented for this view.
	elementId	Optional	Identifies the elements that appear in this view. These elements must have a classification of avctUnit. System-defined units can be added to a specific view by adding the DSView software string representation of the unit type. For example: avctDsr2010, avctCcm1650, avctDsr1020_1X16.
viewcontrol	N/A	Optional	Set of view controls available for each view; any view controls that are not defined are disabled.
	controlId	Required	Identifies the view control on the view. This attribute identifies all available controls that the plug-in can disable or enable for this view. Valid values are: • avctScan - Scan Button • avctAdd - Add Button • avctDelete - Delete Button • avctRights - Rights Button • avctTopology - Topolgy Button • avctOperations - Operations Button • avctTreeView - Display view as a Tree view (true) or List view (false)
	enable	Required	Sets the view control as enabled or disabled. Valid values are: • true - enable control • false - disable control

4.6.3 Validation definition file

The validation definition file (validation.xml) contains validation definitions for the prompts specified in the screens section of the nmm.xml definition file. The validation definition file is not required.

The format of this file follows the format specified for the Apache Software Foundation Commons Validator Rules Configuration (http://jakarta.apache.org/commons/validator/) of the commons-validator package, with the exceptions noted in the following. See the validator_1_0.dtd file which can be downloaded from an open source code library for additional details about the XML elements and attributes:

- The value of the name attribute of the form element must match one of the form IDs in the nmm.xml file.
- The value of the property attribute in the field element must match one of the prompt IDs for the form in the nmm.xml file.

Example: Validation definition file

```
<form_validation>
 <formset>
 <form name="<form id in nmm.xml>">
 <field property="<prompt id in nmm.xml for form>"
depends="required,maxlength,minlenth">
```

4.6.4 Shared element types

The plug-in API provides a means of sharing element types across plug-ins through the use of the scope attribute of the element tag in the nmm.xml. The following scope attribute values allow sharing of data:

- avctSystem - Shares the element type across all plug-ins. The element types are defined and owned by the DSView software core. The plug-ins are allowed to use these system defined types.
- avctDomain - Shares the element type across all plug-ins that are defined with the same domain attribute value such as “avocent.com”. Each plug-in must define certain attributes of the element exactly the same such as the type, category and display name.
- avctGlobal - Shares the element type across all plug-ins. Each plug-in must define certain attributes of the element exactly the same such as the type, category and display name.

This section defines the elements that are shared across plug-ins for scope values avctGlobal and avctDomain for elements with a classification of avctUnit or avctConnection. The element types defined with a scope value of avctSystem are not discussed in this manual. These types can be found in the DSViewConstantsInterface.java file.

Shared Unit Types

The following tables summarizes unit types shared across plug-ins for elements with a classification of avctUnit. The attribute of the element is avctGlobal. The Primary Owner identifies the plug-in that defines the “owner” attribute with the value “avctOwnerPrimary”. The Type column identifies the value for the “type” attribute of the element. The Category column identifies the values for the “category” attribute of the element. The Display Name column identifies the value that should be assigned to the property identified by the “label” attribute of the element.

NOTE: Only one plug-in can be the primary owner; all other plug-ins must specify the owner attribute with the value "avctOwnerSecondary". The primary owner is the plug-in that provides controller-like functionality for the units, such as allowing the units to be added as top-level units through the Add Unit Wizard, providing status information through the getStatusInformation method of the FaultManagementInterface, or providing support for firmware upgrades. The secondary owners are plug-ins that utilize the unit types in a secondary way, such as reporting them as attached devices during the discovery of other top-level appliances or controllers supported by the secondary plug-in.

Table 4.26 Shared Unit Types for avctGlobal- Primary Owner is the Blade Chassis Plug-in

TYPE	CATEGORY	DISPLAY NAME
HPBladeSystemcClass	avctCategoryChassis, avctCategoryController	HP BladeSystem c-Class
HPBladeSystempClass	avctCategoryChassis, avctCategoryController	HP BladeSystem p-Class
IBMBladeCenter	avctCategoryChassis, avctCategoryController	IBM BladeCenter
IBMBladeCenterH	avctCategoryChassis, avctCategoryController	IBM BladeCenter H
IBMBladeCenterT	avctCategoryChassis, avctCategoryController	IBM BladeCenter T
IBMBladeCenterHT	avctCategoryChassis, avctCategoryController	IBM BladeCenter HT
GenericBladeChassis	avctCategoryChassis, avctCategoryController	Generic Blade Chassis
Blade	avctCategoryTargetDevice, avctCategoryCard	Blade
DellDracMC	avctCategoryChassis, avctCategoryController	Dell DRAC MC
HPBladeSystem	avctCategoryChassis, avctCategoryController	HP BladeSystem
Type	Category	Display Name
virtualCenter	avctCategorySoftware, avctCategoryController	VMWare VirtualCenter
virtualEsx	avctCategorySoftware, avctCategoryController	VMWare ESX Server
virtualMachine	avctCategorySoftware, avctCategoryTargetDevice	Virtual Machine

4.6.5 Shared views

A plug-in that is a secondary owner of global unit types can need to define the same views that are defined by the plug-in that is the primary owner of those unit types. This is necessary if the units added by the secondary plug-in should appear in the same views defined by the primary plug-in (typically this would be desired).

For example, the Blade Chassis plug-in is the primary owner of the global blade chassis unit types. The blade chassis plug-in defines the Blade Chassis view where all blade chassis appear. All secondary plug-ins that support any of the global blade chassis unit types should also define the Blade Chassis view so that chassis added by the secondary plug-ins appear in the common Blade Chassis view.

Example: XML definition for the Blade Chassis view that must be included at the end of the plug-in's nmm.xml file when the plug-in supports any of the global blade chassis unit types

```
<views>
 <view viewId="bladeChassis" type="chassis" label="unit.navnode.bladechassis.label"
 scope="avctGlobal" elementId="HPBladeSystemcClass,HPBladeSystempClass,IBMBladeCenter,
 IBMBladeCenterH,IBMBladeCenterHT, IBMBladeCenterT,
GenericBladeChassis,avctDellBlade,DellDracMc,HPBladeSystem">
 <viewcontrol controlId="avctScan" enable="true"/>
 <viewcontrol controlId="avctOperations" enable="true"/>
 <viewcontrol controlId="avctTopology" enable="true"/>
 <viewcontrol controlId="avctTreeView" enable="true"/>
 </view>

 <view viewId="blades" label="views.BladesCards.label" type="avctTargetDeviceView"
 scope="avctSystem" elementId="Blade">
 </view>
</views>
```

Two views are defined, one for the Blade Chassis view and one for the blade cards so that the cards appear in the DSView Software Target Devices view since they are also categorized as target devices.

NOTE: The value of the type attribute for the blades view is avctTargetDeviceView.

The elementId attribute defines a comma separated list of unit types that appear in the view. This list only contains the types the secondary plug-in supports. These types are defined as unit elements in the elements section of the nmm.xml file of the secondary plug-in.

The label attribute defines a resource key to lookup the display name for the view. The resource key can be anything, but the value of the key should be “Blade Chassis” for the Blade Chassis view and “Blade Cards” for the Blades view.

Since the Target Devices view is defined by the system, the scope attribute must have a value of avctGlobal for the Blade Chassis view and avctSystem for the Blades view.

Shared connection types

It is recommended that plug-ins defining their own connection types specify a scope of avctDomain for those connection types. This allows all plug-ins within the same domain to share the same connection types. This makes it easier to reuse the same connection types in the future as new plug-ins are developed that need to support the same types of connections. Following this rule also reduces the chances of duplicate connection types being created by different plug-ins.

NOTE: The scope of a connection type for a previously released plug-in can be modified by releasing a new version of the plug-in that defines a change in the scope for the connection. This is done using the avctModifyScope property on the connection element in the nmm.xml. Contact Vertiv SDK Support for more information.

4.6.6 Category attribute for units

The DSView™ and Rack Power Manager software originally used the appliance/target device model, which supported three unit types (appliances, target devices and cascade devices).

The model is now expanded to allow plug-ins to assign categories for a broader definitions of units. Future enhancements to the DSView software provide additional views and capabilities based on the new categories.

With the legacy model, all elements with the avctUnit classification are assumed to be appliances now, units with a classification of avctUnit can be assigned multiple categories, listed in the following table. A unit type that can be discovered as a plug-in must have identical category types defined in both the nmm.xml and the entity maps.

Table 4.27 avctUnit Categories

CATEGORY	DESCRIPTION
avctCategoryAppliance*	Indicates the unit is an appliance. Assign this category to units that typically appear in the Appliances view of DSView software. If the plug-in does not include the category attribute in the element definition, the Appliance category is assigned by default. An appliance is a self contained physical entity with a well known set of functionality, such as a KVM switch or serial console appliance. The DSView software core interacts with appliances through the plug-in API interfaces such as the ConfigManagementInterface, FaultManagementInterface, DialupManagementInterface, ToolInterface and OperationInterface.
avctCategoryTargetDevice*	Indicates the unit is a target device. Assign to units that typically appear in the Target Devices view of the DSView software. When defining unit elements in the nmm.xml, a unit of this category must also have at least one other category assigned. A target device is any entity that can be accessed from the DSView software either directly over the network, or indirectly through an attached appliance or controller. This category must not be used to define generic target devices. Generic target devices continue to be supported in the same way as before.
avctCategoryCascadeDevice*	Indicates the unit is a cascade device. This category cannot be assigned by a plug-in. A cascade device is either a KVM switch or power controller that is attached to an appliance.
avctCategorySoftware	Indicates the unit is a software product.
avctCategoryChassis	Indicates the unit is a chassis product. For example, the unit can be a Blade Chassis that contains cards.
avctCategoryCard	Indicates the unit is a card within an enclosure. For example, a blade card in a blade chassis.
avctCategorySensor	Indicates the unit is a sensor. For example, a temperature or humidity sensor.
avctCategoryController	Indicates the unit is a controller. Assign to units that are not network appliances in the traditional sense, but have capabilities that might normally be found in an appliance, such as being able to discover and report attached units. A controller is a unit capable of discovering and or controlling other units. For example, a VMWare ESX server is capable of discovering Virtual Machines, so the VMWare ESX server defined by a plug-in is a controller. The DSView software core interacts with controllers through the plug-in API interfaces such as the ConfigManagementInterface, FaultManagementInterface, DialupManagementInterface, ToolInterface and OperationInterface.
avctCategoryUnmanaged	Indicates the unit is unmanaged. Assign this category to discovered units to indicate they are not considered manageable by the plug-in or the DSView software. You cannot define this category in the nmm.xml, you must use the Unit entity maps. When a unit has this category, its functionality is limited. This category is only used with Virtual Machine type units that are added to the database but are not yet licensed.
*These categories are from the legacy appliance/target device model and are mutually exclusive. All other combinations are allowed.	

4.6.7 Relationship type attribute for connections

A relationship represents an association between two entities.

The DSView™ and Rack Power Manager software originally supported only a “connected-to” relationship, which supported three unit types (appliances, target devices and cascade devices).

The model is now expanded to allow plug-ins to specify multiple types of relationships. Future enhancements to the DSView software provide additional views and capabilities based on the following relationship types:

- avctRunsOn (avctRuns)
- avctConnectedTo (avctConnectsTo)
- avctActivatedBy (avctActivates)
- avctInstalledOn (avctContains)
- avctSupportedBy (avctSupports)
- avctManagedBy (avctManages)

NOTE: The value specified is assigned to the Z end of the connection. There is an implied type for the A end of the connection based on the type specified for the Z end. The implied type is automatic and cannot be specified by the plug-in. The implied types for the A end of the connections is shown in parentheses in the preceding list.

Examples

The following examples demonstrate how the new category and relationship type attributes can be used to model two typical network configurations that could be supported by a plug-in.

NOTE: Only three tiers of connectivity are permitted between units. In the future, this limitation can be removed.

Figure 4.1 Unit Modeling Example 1

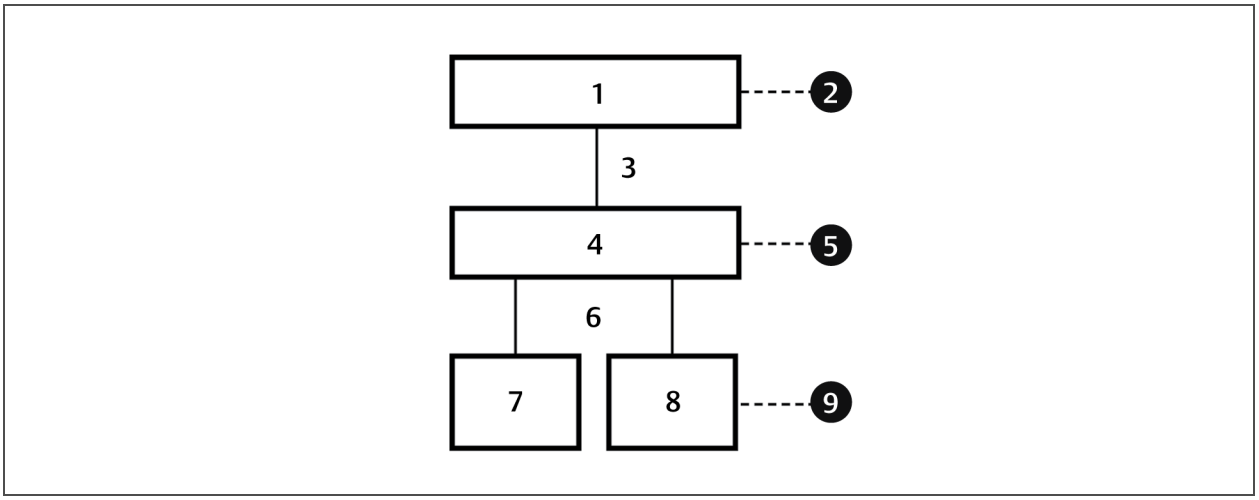

Table 4.28 Unit Modeling Example 1 Component Descriptions

ITEM	DESCRIPTION
1	MergePoint Unity switch
2	avctCategory Appliance
3	Connected to via network cable
4	Blade Chassis
5	avctCategoryChassis avctCategoryController
6	Directly installed on
7	Blade Card 1
8	Blade Card 2
9	avctCategoryCard avctCategoryTargetDevice

Figure 4.2 Unit Modeling Example 2

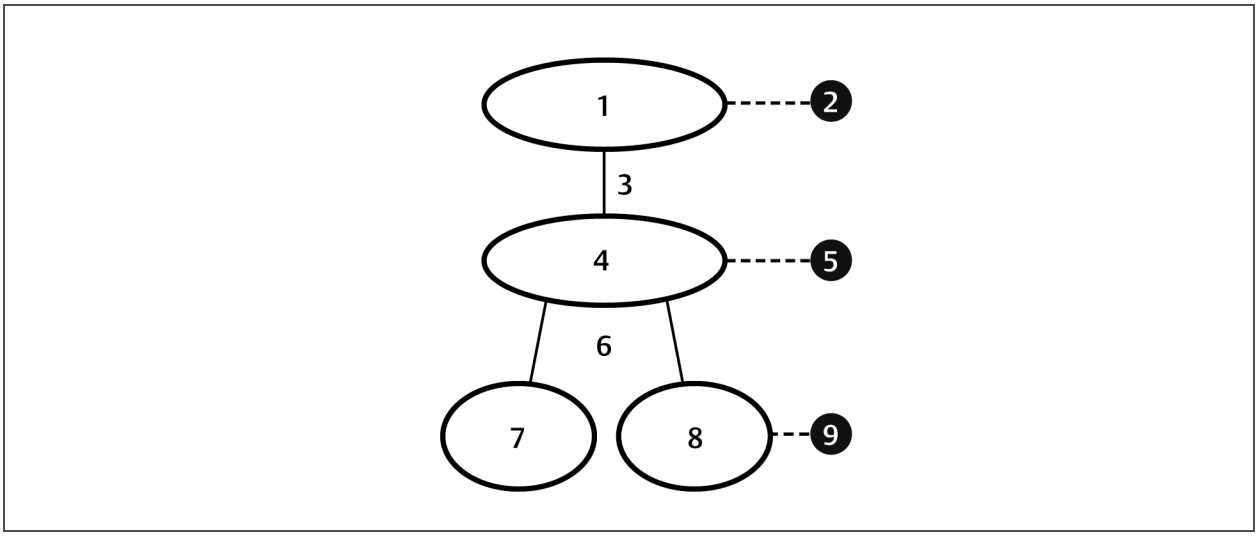

Table 4.29 Unit Modeling Example 2 Component Descriptions

ITEM	DESCRIPTION
1	Virtual Center
2	avctCategorySoftware avctCategoryController
3	Managed by
4	VMWare ESX
5	avctCategorySoftware avctCategoryController
6	Runs on
7	Virtual Machine
8	Virtual Machine
9	avctCategorySoftware avctCategoryTargetDevice

4.6.8 Configuration template file

The configuration template file, which is created by the unit's plug-in, is activated by using the Save Configuration Template service. This file contains unit information to be used as a template. This file also contains information about the appliance configuration template and the additional unit types for which the appliance configuration template can be applied. The plug-in determines the settings stored in the file, the value for each setting and the property setting names. Each template submitted to the DSView™ and Rack Power Manager software by the plug-in is validated by the Document Type Definitions file (DTD).

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION
!DOCTYPE configurationtemplate SYSTEM	N/A	Required	This field is required for all submitted templates to validate the structure with the DSView software DTD. Contact Vertiv Technical Support for the latest DTD.
configurationtemplate	N/A	Required	Set of definitions describing the configuration template.
identity	N/A	Required	Set of attributes identifying the configuration template.
	templateName	Required	Name of the configuration template.
	unitTypes	Required	Unit types supporting the configuration template.
	nmmId	Required	The nmmId of the plug-in that created the template.
	version	Required	The version of the NMM that created the template.
	templateId	Required	The template ID of this template.
	vendor	Optional	The NMM vendor that created the template.
	domain	Optional	The NMM domain that created the template.
template	N/A	Required	Set of property definitions describing all the available property settings. All properties within the properties node must be unique.
optionlist	N/A	Optional	Identifies a list of options that can be used by one or more option lists. An entire option list can be created under the optionlist node and then be identified by an option node with the optionlistId attribute.
	optionlistId	Required	A unique identifier of an optionlist node.
property	N/A	Optional	Identifies a property associated with a setting.
	propertyId	Required	A unique identifier of a property setting applied by the appliance plug-in as part of the configuration template.
	value	Required	The value applied to this setting property.
	classification	Required	The type of value that is allowed to be saved to this setting property. If no type attribute is defined, the default classification is String. The following set of types are supported: • String - Indicates the value is a string value; creates a text box for user editing. • Integer - Indicates the value is an integer type; creates a text box for user editing. • UTF8 - Indicates the value is an UTF8 value; creates a text box for user editing.
(continued)	(continued)	(continued)	• Float - Indicates the value is a Float value; creates a text box for user editing. • IPAddress - Indicates the value is an IP address; creates a text box for user editing. • Boolean - Indicates the value is true or false. Templates pushed to the plug-in receive a value of true or false. By default, any value pulled from the plug-in that does not specify true is assumed false; creates a checkbox for user editing. • ReadOnly - Indicates a value is not editable and is only stored to be returned to the plug-in.

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION
			• SingleSelect - With the use of the option node, creates a drop-down selection. The user-selected value is populated in the property value. • MultipleSelect - With the use of the option node, creates a drop-down selection. The user-selected value is populated in the property value.
	label	Required	Resource key to retrieve the prompt text.
	labelvalues	Optional	Used in conjunction with the label attribute, the labelvalues attribute holds a comma delimited list of values that can be inserted within the label string. The label string value contains placeholders for the labelvalues. The placeholders must be in the format of "{<labelvalues position>}" within the string value. For example: • Resource file: label.user=Username {0} is {1} • Template file: <property propertyId="TestA" classification="String" label="label.user" value="test" labelvalues="1,Bob"/> • Display: Username 1 is Bob Values used in the labelvalues attributes are not translated and are placed in the string as is.
	delimiter	Optional	Used in conjunction with a classification attribute of MultipleSelect. This attribute allows the selected values to be delimited by the character defined in the delimiter attribute. Only one character can be defined by the delimiter attribute. If providing values in the value attribute, the selection must also be delimited by the character given in the delimiter attribute.
(continued)	size	Optional	For properties with the classification String or UTF8, a maximum string size is defined for the setting property.
	minRange	Optional	For properties with the classification Integer or Float, a range can be defined if the value must be within a number range. The minimum attribute defines the lowest point of this range.
	maxRange	Optional	For properties with the classification of Integer or Float, a range can be defined if the value needs to be within a number range. The maximum attribute defines the highest point of this range.
option	N/A	Optional	For property nodes that have a predefined set of values, the option node can be used to define the allowable values for the value attribute.
	label	Optional	Resource key to retrieve the prompt text.
	value	Optional	The value applied to this setting property.
	optionlistId	Optional	Identifies a pre-existing list of options defined in an optionlist node. This option list can be used in place of defining a new list of options for this property. If an optionlistId attribute is used, all options defined for this property attribute are ignored.
	labelvalues	Optional	Used in conjunction with the label attribute, the labelvalues attribute holds a comma delimited list of values that can be inserted within the label string. The label string value must contain placeholders for the labelvalues. The placeholders must be in the format "{<labelvalues position>}" within the string value. For example:

ELEMENT	ATTRIBUTE	REQUIREMENT	DESCRIPTION
			• Resource file: label.user=Username {0} is {1} • Template file: <option label="label.user" value="test" labelvalues="1,Bob"/> • Display: Username 1 is Bob Values used in the labelvalues attributes are not translated and are placed in the string as is.
elements	N/A	Required	Set of element and property definitions describing all available settings. All elements' nodes must be unique.
	elementsId	Required	A unique identifier of elements the appliance plug-in applies as part of the configuration template.
	label	Optional	Resource key to retrieve the text that is displayed for the group of element and property nodes for this element's node.
	labelvalues	Optional	Used in conjunction with the label attribute, the labelvalues attribute holds a comma delimited list of values that can be inserted within the label string. The label string value must contain placeholders for the labelvalues attributes. The placeholders must be in the format "{<labelvalues position>}" within the string value. For example: • Resource file: label.user=Username {0} is {1} • Template file: < elementsId="test" label="label.user" labelvalues="1,Bob"/> • Display: Username 1 is Bob Values used in the labelvalues attributes are not translated and are placed in the string as is.
element	N/A	Optional	Identifies an element of a setting. This setting is used to allocate additional fields associated with an element's node.
	elementId	Required	A unique identifier of element setting the appliance plug-in applies as part of the configuration template.
	value	Optional	The value applied to this element's property.

4.7 Data

All data transferred between the DSView software and the plug-ins is in standard Java objects. Each map object has an identifying classification attribute. The following Java object value types are supported for data transfer:

- String - java.lang.String
- Integer - java.lang.Integer
- Long - java.lang.Long
- List - java.util.List
- Map - java.util.Map
- X509Certificate - java.security.cert.X509Certificate
- PrivateKey - java.security.PrivateKey

Attributes and values are constants available through the DSViewInterface object.

NOTE: Attributes and values are specified in uppercase letters and italicized. For example, in the attribute DSViewInterface.*CLASSIFICATION*, DSViewInterface is the attribute and *CLASSIFICATION* is the value. The actual value of all attributes begins with "Avct" and the actual value of all value objects begins with "avct."

4.7.1 Element relationship ID

The element relationship ID (specified by the ELEM_RELATIONSHIP_ID attribute) is a unique identifier for a specific entity, for example, a unit or connection. The following are relationship definitions:

- For the root unit, the element relationship ID is: <ELEM_ID_ROOT>.
- For a single connection between a controlling unit and a target unit, the element relationship ID is:

  Controlling Unit < Connection 1 > Target Unit

  <ELEM_ID_ROOT>

  <ELEM_RELATIONSHIP_ID_DELIMITER><Connection Element ID>
- For multiple connections between a controlling unit and a target unit, the element relationship ID is:

  Controlling Unit < Connection 1 > Cascade Unit < Connection 2 > Target Unit

  <ELEM_ID_ROOT>

  <ELEM_RELATIONSHIP_ID_DELIMITER><Connection Element ID>

  <ELEM_RELATIONSHIP_ID_DELIMITER><Connection Element ID>
- The format of the <Connection Element ID> is:

  <CONNECTION_AEND_PORT>

  <ELEM_CONNECTION_ID_DELIMITER><CONNECTION_TYPE>

  <ELEM_CONNECTION_ID_DELIMITER><CONNECTION_IDENTIFIER>
- An identifier can use the following delimiters:

  The ELEM_RELATIONSHIP_ID_DELIMITER value is the pipe symbol (|).

  The ELEM_CONNECTION_ID_DELIMITER value is a comma (,).

  The ASCII character set must be used and the pipe (|), comma (,) or space characters cannot be used in the CONNECTION_TYPE or CONNECTION_IDENTIFIER attributes.

Example: Element relationship ID for a KVM connection between controlling unit C1 (port 2) and target unit T1

```
avctRoot|2,avctKvm,Port2
```

Example: Element relationship ID for a KVM connection between controlling unit C1 (port 3) and cascade unit CASCADE1 and a KVM connection between cascade unit CASCADE1 (channel 5) and target unit T1

```
avctRoot|3,avctKvm,Port3|5,avctKvm,Channel5
```

4.7.2 Classification attributes and values

The classification attribute identifies the object represented by a map. Some objects represent elements that are managed by the DSView™ and Rack Power Manager software and others represent objects that provide data associated with a managed element. The following sections describe the supported classification attribute values.

Unit

The following table lists the unit classification attributes.

Table 4.30 Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_UNIT.
UNIT_ADDRESS *	String	Optional **	Address of the unit.
UNIT_NAME *	String	Optional **	Name of the unit.
UNIT_OID *	Long	Optional **	Unique identifier defined by the DSView software for the unit.
UNIT_SERVER_OID *	Long	Optional	Unique identifier defined by the DSView software for the DSView software server that is managing this unit.
UNIT_TYPE *	String	Optional	Type of unit.
UNIT_ACONNECTIONS	List	Optional	List of connections, where this unit is the A end of the connection. This attribute is provided only when identifying a unit's connection path.
UNIT_ZCONNECTIONS	List	Optional	List of connections, where this unit is the Z end of the connection. This attribute is provided only when identifying a unit's connection path.
UNIT_DESCRIPTION *	String	Optional	Description.
UNIT_ASSET_TAG *	String	Optional	Asset tag.
UNIT_MODEL *	String	Optional	Model.
UNIT_PART_NUMBER *	String	Optional	Part number.
UNIT_SERIAL_NUMBER *	String	Optional	Serial number.
UNIT_MIGRATION_NEEDED *	String	Optional	Indicates if the unit requires migration (true or false).
UNIT_CERTIFICATE	X.509 Certificate	Optional	Certificate.
UNIT_CERTIFICATE_PRIVATE_KEY	PrivateKey	Optional	Private key.
UNIT_BROWSER_URL *	String	Optional	Browser URL.
UNIT_TELNET_PORT *	Integer	Optional	Telnet port.
UNIT_ACCOUNT_INFO *	String	Optional	Account information.
UNIT_FIRST_CONTACT_NAME *	String	Optional	First contact name.
UNIT_FIRST_CONTACT_PHONE *	String	Optional	First contact phone number.
UNIT_SECOND_CONTACT_NAME *	String	Optional	Second contact name.
UNIT_SECOND_CONTACT_PHONE *	String	Optional	Second contact phone number.
UNIT_CUSTOM_0 *	String	Optional	Custom 0 field.
UNIT_CUSTOM_1 *	String	Optional	Custom 1 field.
UNIT_CUSTOM_2 *	String	Optional	Custom 2 field.
UNIT_NOTES *	String	Optional	Notes.

Table 4.30 Classification Attribute (continued)

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
UNIT_EXTENDEDPROPERTIES	Map	Optional	Map of a selected unit's properties. Attributes and values are expected to be strings.
UNIT_POWER_STATUS	String	Optional	Power status. This attribute is needed only when providing discovery information to the DSView software. It is not provided to the plug-in when the Repository Service retrieves unit information. Valid values (appended to UNIT_POWER_STATUS_) are: ON OFF If the Add Unit Wizard is being used, a unit is not added if its power status is off. The Resync Wizard and the automatic topology synchronization operations have user-settable properties that indicate if offline connections should be included.
UNIT_IS_DEFAULT_NAME	Boolean	Optional	Indicates if the name specified in UNIT_NAME is a default name or not. The default value is false. The property is not returned by the DSView software in queries; it is only used when the unit is being added to the DSView software via Add Unit, Resync or Auto Topology.
UNIT_CATEGORY *	List	Optional	Indicates the category assigned to the unit. Valid values are: UNIT_CATEGORY_APPLIANCE UNIT_CATEGORY_CASCADE_DEVICE UNIT_CATEGORY_TARGET_DEVICE UNIT_CATEGORY_SOFTWARE UNIT_CATEGORY_CHASSIS UNIT_CATEGORY_CARD UNIT_CATEGORY_SENSOR UNIT_CATEGORY_CONTROLLER UNIT_CATEGORY_UNMANAGED
NOTE: The UNIT_CATEGORY attribute was introduced in the DSView software version 3.4 and had a value type of String. Starting in the DSView software version 3.4.1, the value type was changed to a List, which is now the preferred type. In DSView software version 3.5 and later, a unit can be assigned multiple categories.			

Table 4.30 Classification Attribute (continued)

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
UNIT_MOBILE_DEVICE_ACCESS *	String	Optional	Indicates whether the unit is allowed to be managed from a mobile device or not. Valid values are true and false.
UNIT_OPERATIONS	List	Optional	A list of maps. Each map contains information about an operation that is supported for the unit.
NOTE: The UNIT_OPERATIONS attribute is provided in repository query or search results when the query or search specifies a client session ID. The operations returned are the operations that you have the access rights to.			
UNIT_ZONE_OID*	Long	Optional	A unique identifier defined by the DSView software for the zone associated with the unit.
NOTE: The UNIT_ZONE_OID attribute, introduced in DSView software version 3.5.1, is provided when the unit is being added to the DSView system. The plug-in can access the value of the attributes but cannot modify its contents.			
* This attribute can be used in the query filter. ** At least one of the attributes UNIT_ADDRESS, UNIT_NAME or UNIT_OID is required to identify the unit.			

Unit properties

The following table lists the unit properties, classification attributes.

Table 4.31 Unit Properties Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_UNIT_PROPERTIES
UNIT_PROPERTIES	Map	Required	Properties to be updated.

Connection

The connection object defines the relationship between two units. Two levels of connection relationships are supported. The A end of a connection is the endpoint of a connection that is closest to the controlling unit. The Z end of a connection is the endpoint that is closest to the target unit. The supported connection relationships and endpoints are:

- Controlling Unit Connection 1 Cascade Unit Connection 2 Target Unit:
  - Connection 1 - The A end is the controlling unit; the Z end is the cascade unit.
  - Connection 2 - The A end is the cascade unit; the Z end is the target unit.
- Controlling Unit Connection 1 Target Unit:
  - Connection 1 - The A end is the controlling unit; the Z end is the target unit.

Table 4.32 Connection Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_CONNECTION
CONNECTION_TYPE *	String	Required **	Connection type. Can be defined by either the plug-in or by the DSView software CONNECTION_TYPE_ <specific>, where <specific> is a DSView software defined connection type.
CONNECTION_OID *	Long	Optional	Unique identifier defined by the DSView software.
CONNECTION_AEND *	Map	Optional	Map representing the unit at the A end of the connection; see Classification attributes and values on page 137.
CONNECTION_AEND_PORT *	Integer	Required	Connection point of the A end of the unit. Must be unique for connections of the same type and the same A end unit. Generally, for physical ports, this is the port number. The value should be well understood by the plug-in, as it is used in calls to the plug-in to perform operations, launch tools and modify settings related to this connection.
CONNECTION_AEND_LABEL *	String	Required	Label associated with the connection point of the A end unit. Generally, for physical ports, this is the label silk-screened on the port. For non-physical connection points, this string indicates the connection point.
CONNECTION_ZEND *	Map	Optional	Map representing the unit at the Z end of the connection; see Classification attributes and values on page 137.
CONNECTION_ZEND_PORT *	Integer	Optional	Not used.
CONNECTION_ZEND_LABEL *	String	Optional	Label associated with the connection point of the Z end unit. Generally, for physical ports, this is the label silk-screened on the port. For non-physical connection points, this string indicates the connection point.
CONNECTION_IDENTIFIER *	String	Required **	Identifier that provides a clear understanding/identification of the connection to the plug-in.
CONNECTION_EXTENDED_PROPERTIES	Map	Optional	Map of the custom properties associated with the connection.

Table 4.32 Connection Classification Attribute (continued)

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CONNECTION_STATUS	String	Optional	This attribute is needed only when providing discovery information to the DSView software. It is not provided to the plug-in when the RepositoryService retrieves connection information. Valid values (appended to CONNECTION_STATUS_) are: • ONLINE • OFFLINE **NOTE: If the Add Unit Wizard is being used, a unit is not added if its connection status is off. The Resync Wizard and the automatic topology synchronization operations have user-settable properties that indicate if offline connections should be included.**
CONNECTION_AEND_RELATIONSHIP ***	String	Optional	The relationship type for the A end of the connection. Valid values (appended to CONNECTION_RELATIONSHIP_) are: • RUNS • CONNECTS_TO • ACTIVATES • CONTAINS • SUPPORTS • MANAGES
CONNECTION_ZEND_RELATIONSHIP ***	String	Optional	The relationship type for the Z end of the connection. Valid values (appended to CONNECTION_RELATIONSHIP_) are: • RUNS_ON • CONNECTED_TO • ACTIVATED_BY • INSTALLED_ON • SUPPORTED_BY • MANAGED_BY

* This attribute can be used in the query filter. See Query filter on page 154 for details.

** The CONNECTION_TYPE and CONNECTION_IDENTIFIER attributes should not include delimiters that are used in the relationship ID. See Element relationship ID on page 136 for details.

*** See Relationship type attribute for connections on page 129 for more information about relationships.

Server

The following table lists the server classification attributes.

Table 4.33 Server Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_SERVER.
SERVER_OID *	Long	Required *	Unique identifier defined by the DSView software.
SERVER_GUID *	String	Required	Server identifier.
SERVER_PORT *	Integer	Required	Port number of the server (used for communication with other servers).
SERVER_ ADDRESS *	String	Required	Address of the server that the DSView software uses to identify the server. This address should not be provided to the plug-in as the DSView software server address.
SERVER_NAME *	String	Required	Server name.
SERVER_ROLE *	String	Required	Server role; valid values (appended to SERVER_ROLE_) are: • PRIMARY • BACKUP • UNAFFILIATED
SERVER_SSH_ PORT	String	Required	SSH port number.
SERVER_HTTPS_ PORT	String	Required	HTTPS port number.
SERVER_HTTP_ PORT	String	Required	HTTP port number.
SERVER_ SYSLOG_PORT	String	Required	Syslog port number.
SERVER_PROXY_ PORT	String	Required	Proxy port number.
SERVER_ CERTIFICATE	X509 Certificate	Required	Server certificate.
* This attribute can be used in the query filter. See Query filter on page 154 for details.			

Event

The following table lists the event classification attributes.

Table 4.34 Event Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_EVENT.
EVENT_NAME	String	Required	Event name.
EVENT_UNIT	Map	Optional	Unit associated with the event.
EVENT_USER	String	Optional	Name of user associated with the event.
EVENT_ARGS	String{}	Optional	Arguments to be substituted with parameters of the event message.
EVENT_TRAP_TIME_STAMP	Long	Optional*	Timestamp of trap to generate the event.
EVENT_TRAP_ENTERPRISE_ID	String	Optional*	Enterprise ID of the trap to generate the event.
EVENT_TRAP_GENERIC_ID	Integer	Optional*	Generic ID of the trap to generate the event.
EVENT_TRAP_SEPCIFIC_ID	Integer	Optional*	Specific ID of the trap to generate the event.
* All of these attributes are required if any of them are specified.			

File

The following table lists the file classification attributes.

Table 4.35 File Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_FILE.
FILE_DATA_INPUTSTREAM	InputStream	Optional	Input stream of the file.
FILE_TYPE	String	Optional	File type.
FILE_CATEGORY	String	Optional	File category. Valid values are: • FILE_CATEGORY_FIRMWARE • FILE_CATEGORY_CONFIG • FILE_CATEGORY_USER_DB • FILE_CATEGORY_TEMPLATE
FILE_OID	Long	Optional	Database OID of the file being pushed. This is placed in the map only for an appliance file for a firmware upgrade.
FILE_PUSH_REASON	String	Optional	Identifies why a file is pushed to an entity. Valid values are: • FILE_PUSH_REASON_REPLACEMENT • FILE_PUSH_REASON_PROVISIONING

File operation

The following table lists the file operation classification attributes.

Table 4.36 File Operation Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_FILE_OPERATION.
FILE_STATUS	String	Optional	File operation status.
FILE_REBOOT	Boolean	Optional	Whether a reboot of the unit is necessary to fully process the file operation.
FILE_PUSH_ RESULT_ MESSAGE	String	Optional	Result of applying a template to a specific unit. This should supply information if only part of the template properties have been applied. This information is displayed at the end of an Apply Configuration Template or Appliance Replacement operation to inform you that some properties are not updated to the unit. The string information placed in this map property should be translated prior to placing the string in the map.

License

The following table lists the license classification attributes.

Table 4.37 License Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_LICENSE.
LICENSE_TYPE	String	Required	Valid values (appended to LICENSE_TYPE_) are: • POWER • DATA_LOGGING
TOTAL_LICENSE_COUNT	Integer	Optional	Total number of licenses.
TOTAL _LICENSES_AVAILABLE	Integer	Optional	Number of licenses available.
TOTAL_LICENSES_RESERVED	Integer	Optional	Number of reserved licenses.

SNMP trap

The following table lists the SNMP trap classification attributes.

Table 4.38 SNMP Trap Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_SNMP_TRAP.
SNMPTRAP_UNIT	Map	Optional	Unit definition of the agent that generated the SNMP trap; see Classification attributes and values on page 137.
SNMPTRAP_TIME_STAMP	Long	Required	Timestamp of the trap.
SNMPTRAP_ENTERPRISE_ID	String	Required	Enterprise of the trap.
SNMPTRAP_GENERIC_ID	Integer	Required	Generic ID of the trap.
SNMPTRAP_SPECIFIC_ID	Integer	Required	Specific ID of the trap.
SNMPTRAP_AGENT_ADDRESS	String	Required	Address of the agent.
SNMPTRAP_AGENT_PORT	Integer	Required	Port of the agent.
SNMPTRAP_COMMUNITY_STRING	String	Required	Community.
SNMPTRAP_VARBINDINGS	Map	Required	Set of varbinds associated with the trap.

Syslog event

The following table lists the syslog event classification attributes.

Table 4.39 Syslog Event Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_SYSLOG_EVENT.
UNIT	Map	Required	Definition of the unit generating the Syslog event. See Classification attributes and values on page 137 for details.
SYSLOG_EVENT_TIME_STAMP	String	Required	Timestamp of the Syslog event.
SYSLOG_EVENT_ID	Integer	Required	Event ID of the syslog event.
SYSLOG_EVENT_MESSAGE	String[]	Required	Syslog event message.

Entity status

The following table lists the entity status classification attributes.

Table 4.40 Entity Status Attributes

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
ELEMENT_ RELATIONSHIP_ID	String	Required	Element relationship ID of the element with which the status is associated; see Element relationship ID on page 136.
ELEMENT_STATUS	Map	Required	Map representing the status of the element; see Element status on page 147.

Element status

The following table lists the element status classification attributes.

Table 4.41 Element Status Attributes

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
OPERATIONAL STATUS	String	Required	Operational status of the element. Valid values (appended to OPERATIONAL_ STATUS_) are: • RESPONDING • NOT_RESPONDING • NOT_AVAILABLE
SESSION_ STATUS	Map	Optional	Map representing the status of each session type supported/associated with this element; see Session status on page 147.
POWER_ STATUS	String	Optional	Power status of the element. Valid values (appended to POWER_STATUS_) are: • ON • OFF • UNKNOWN

Session status

The following table lists the session status classification attributes.

Table 4.42 Session Status Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
SESSION_TYPE	String	Required	Session type. Valid values (appended to SESSION_TYPE_) are: • KVM • VM • SERIAL • SOL • PLUGIN_DEFINED The SESSION_TYPE_PLUGIN_DEFINED value is the default value for connections not defined by the DSView software.
SESSION_STATUS	String	Required	Status of the specified session type. Valid values (appended to SESSION_STATUS_) are: • IDLE • ACTIVE • BLOCKED (not supported for SERIAL, SOL or PLUGIN_DEFINED session types) • SUSPENDED (not supported for KVM, VM, SERIAL or SOL session types)

Entity session

The following table lists the entity session classification attributes.

Table 4.43 Entity Session Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
SESSION_LIST	List	Required	List of sessions associated with the entity. See Element session on page 148.

Element session

The following table lists the element session classification attributes.

Table 4.44 Element Session Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
ELEMENT_RELATIONSHIP_ID	String	Required	Element relationship ID of the element with which the session is associated. See Element relationship ID on page 136.
SESSION_ATTRIBUTES	Map	Required	Map representing session information. See Session on page 148.

Session

The following table lists the session classification attributes.

Table 4.45 Session Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
SESSION_TYPE	String	Required	Session type. Valid values (appended to SESSION_TYPE_) are: • NATIVE • KVM • SERIAL • SOL • VM
SESSION_MODE	String	Required	Session mode. Valid values (appended to SESSION_MODE_) are: • NORMAL • STEALTH • SCAN • NATIVE • SERIAL • VIDEO • VIRTUAL_MEDIA • EXCLUSIVE • PREEEMPT_NON_EXCLUSIVE • PREEMPT_EXCLUSIVE • SHARE • SHARE_NON_EXCLUSIVE • PASSIVE • PASSIVE_NON_EXCLUSIVE • PASSIVE_SHARE • FORCE_SHARE • PASSIVE_FORCE_SHARE
SESSION_OWNER	String	Optional	Username of session owner.
SESSION_START_TIME	Long	Required	Time the session started.
SESSION_PREEMPTION_LEVEL	Integer	Optional	Session preemption level.

Entity indicator

The following table lists the entity indicator classification attributes.

Table 4.46 Entity Indicator Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
ELEMENT_RELATIONSHIP_ID	String	Required	Element relationship ID of the element with which the indicators are associated; see Element relationship ID on page 136.
<indicators>	Map	Optional	Map containing a set of indicators used by the DSView software. The name of the indicator is the map key. Valid values (appended to INDICATOR_) are: • NAME_CHANGE (String) • TOPOLOGY_CHANGE (String) • NAME_CHANGE_OCCURRED (Boolean) • TOPOLOGY_CHANGE_OCCURRED (Boolean)

Configured discovery

The following table lists the configured discovery classification attributes.

Table 4.47 Configured Discovery Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_CONFIGURED_DISCOVERY.
DISCOVERY_ ADDRESS	String	Required	Address to use when attempting to communicate with the element.
DISCOVERY_ READ_ COMMUNITY	String	Optional	Read community to be used when communicating with the element.
DISCOVERY_ PROPERTIES	Map	Required	Map providing additional data that is entered in the associated form(s). Generally, this is data in the Add Unit and Resync wizards. The keys for this data match the name of the properties associated with the form(s).
DISCOVER_ REASON	String	Optional	Reason for discovering the device; used if the unit needs to be rediscovered for an Appliance Replacement operation. Valid values are: • FILE_PUSH_REASON_REPLACEMENT

Configured discovery by range

The following table lists the configured discovery by range classification attributes.

Table 4.48 Configured Discovery Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_RANGE_DISCOVERY
DISCOVERY_ IPV4_START	String	Optional	The starting range for IPV4 addresses; must be used with DISCOVERY_IPV4_END.
DISCOVERY_ IPV4_END	String	Optional	The ending range for IPV4 addresses; must be used with DISCOVERY_IPV4_ START.
DISCOVERY_ IPV6_NETWORK	String	Optional	The IPV6 network.
DISOVERY_IPV6_ BITS	String	Optional	The number of bits to be used from the IPV6 network.
DISCOVERY_ IPV6_LEVEL	String	Optional	Defines the level of the IPV6 discovery. Valid values are: • DISCOVERY_IPV6_LEVEL_LL (Link Local) • DISCOVERY_IPV6_LEVEL_UL (Unique Local) • DISCOVERY_IPV6_LEVEL_GL (Global)
UNIT_TYPE	String	Optional	The appliance type to use when attempting to communicate with the element, if needed.
DISCOVER_ READ_ COMMUNITY	String	Optional	The read community to be used when communicating with the element, if needed.
DISCOVERY_ PROPERTIES	Map	Required	A map providing additional data that you enter in the associated form(s). Generally this is data in the Add Unit and Resync Wizards. The keys for this data match the name of the properties associated with the form(s).

Discovery information

The following table lists the discovery information classification attributes.

Table 4.49 Discovery Information Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
DISCOVERY_INFO_SYSOBJECTID	String	Optional	Value of the sysObjectId found during discovery. This is the value returned from the SNMP MIBII variable sysObjectId variable.
DISCOVERY_INFO_SNMP_STATUS	String	Optional	Status of the SNMP communication attempt. Expected values (appended to DISCOVERY_INFO_SNMP_STATUS_) are: • RESPONDING • NOT_RESPONDING
DISCOVERY_INFO_AIDP_STATUS	String	Optional	Status of the AIDP communication attempt. Expected values (appended to DISCOVERY_INFO_AIDP_STATUS_) are: • RESPONDING • NOT_RESPONDING
DISCOVERY_INFO_AIDP_TYPE	String	Optional	Type determined by the AIDP communication.
DISCOVERY_INFO_ASMP_STATUS	STring	Optional	Status of the ASMP communication attempt. Expected values (appended to DISCOVERY_INFO_ASMP_STATUS_) are: • RESPONDING • NOT_RESPONDING
DISCOVERY_INFO_ASMP_TYPE	String	Optional	Type determined by the ASMP communication.
DISCOVERY_INFO_HTTP_STATUS	String	Optional	Status of the HTTP communication attempt. Expected values (appended to DISCOVERY_INFO_HTTP_STATUS_) are: • RESPONDING • NOT_RESPONDING
<domain>	Map	Optional	Map providing additional data such as information or status about plug-in communication attempts that is understood by plug-ins in the same domain.

Discovery information map for domain

The following table lists the discovery information maps for domain classification attributes.

Table 4.50 Discovery Information Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
Protocol X Status	String	Optional	Status representing the protocol X communication attempt.
Protocol X Type	String	Optional	Type determined by the protocol X communication.
Protocol Y Status	String	Optional	Status representing the protocol Y communication attempt.
Protocol Y Type	String	Optional	Type determined by the protocol Y communication.

SNMP trap notification filter

The following table lists the SNMP trap notification's filter classification attributes.

Table 4.51 SNMP Trap Notification Filter Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_SNMP_TRAP_FILTER.
SNMP_TRAP_ PROPERTIES	List	Optional	Each list entry is a set of properties for matching traps. If no trap properties are specified, all traps are returned.

SNMP trap properties filter

The following table lists the SNMP trap properties' filter classification attributes.

Table 4.52 SNMP Trap Properties Filter Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_TRAP_PROPERTIES_FILTER.
SNMP_TRAP_FILTER_ ENTERPRISE	String	Optional	Match criteria for the trap's enterprise, in dot notation (for example, 1.3.6.4.1.10418.1.2.3.4).
SNMP_TRAP_FILTER_ SPECIFIC	Integer	Optional	Match criteria for the trap's specific ID.
SNMP_TRAP_FILTER_ GENERIC	Integer	Optional	Match criteria for the trap's generic ID.

Syslog notification filter

The following table lists the syslog's notification filter classification attributes.

Table 4.53 Syslog Notification Filter Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_SYSLOG_FILTER.
SYSLOG_ PROPERTIES	List	Optional	Each entry in the list is a set of properties for matching syslogs. If no properties are specified, all syslogs are returned.

Syslog properties filter

The following table lists the syslog's properties filter classification attributes.

Table 4.54 Syslog Properties Filter Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_SYSLOG_PROPERTIES_FILTER.
SYSLOG_FILTER_EVENT_ID	Integer	Optional	Match criteria for the event ID of a Syslog event.

Email config filter

The following table lists the email config filter's classification attributes.

Table 4.55 Email Config Filter Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_EMAIL_CONFIG_FILTER.
EMAIL_CONFIG_SERVER_ADDRESSES	List	Required	A list of the server addresses this notification registers.

Server properties notification filter

The following table lists the server's properties notification filter classification attributes.

Table 4.56 Server Properties Notification Filter Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_SERVER_PROPERTIES_FILTER.
SERVER_ PROPERTIES	Map	Required	A map representing properties for a DSView software server. This map can contain only the following two entries: • A CLASSIFICATION of CLASSIFICATION_SERVER_PROPERTIES. The value is a String. • The SERVER_ADDRESS attribute defines who receives the notification. The value is a String.

Unit notification filter

The following table lists the unit's notification filter classification attributes.

Table 4.57 Unit

ATTRIBUTE	VALUE TYPE	VALUE SET/RESTRICTIONS	REQUIREMENTS
CLASSIFICATION	String	CLASSIFICATION_NOTIFICATION_UNIT_FILTER.	Required
NOTIFICATION_ FILTERS	Map	Contains the filter information identifying the types of units and the unit categories that generate notifications. The map can contain one or both of the following keys: • UNIT_TYPE • UNIT_CATEGORY The value for each key is a list. The unit type list contains the filtered unit types. The unit category list contains the filtered unit categories. If both lists are provided, a notification occurs when the unit type and the unit category match. The plug-in is notified if a unit that matches the filter is added, modified or deleted.	Required

Connection notification filter

The following table lists the connection's notification filter classification attributes.

Table 4.58 Connection Notification Filter

ATTRIBUTE	VALUE TYPE	VALUE SET/RESTRICTIONS	REQUIREMENTS
CLASSIFICATION	String	CLASSIFICATION_NOTIFICATION_UNIT_FILTER.	Required
NOTIFICATION_FILTERS	Map	Contains the filter information that identifies the types of units and the unit categories that generate notifications. The map can contain one or both of the following keys: • UNIT_TYPE • UNIT_CATEGORY The value for each of these keys is a list. The unit type list contains the filtered unit types. The unit category list contains the filtered unit categories. If both lists are provided, a notification is sent when the unit type and the unit category match. The plug-in is notified if a unit that matches the filter is added, modified or deleted.	Required

DSView Software

The following table lists the DSView software's classification attributes.

Table 4.59 DSView Software Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_DSVIEW.
DSVIEW_CERTIFICATE	X509 Certificate	Required	DSView software certificate.
DSVIEW_PRIVATE_KEY	PrivateKey	Required	DSView software private key.
DSVIEW_ADDRESS	String	Required	Address of the DSView software server responding to this request.
DSVIEW_SERVERS	List	Required	List of addresses of each of the DSView software servers associated with this DSView software system, including the address of the DSView software server responding to this request.

Query filter

The following table lists the query's filter classification attributes.

Table 4.60 Query Filter Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_QUERY_FILTER.
QUERY_FILTER_CLASSIFICATION	String	Required	Classification of the type of object being retrieved. Valid values are: • CLASSIFICATION_DSVIEW • CLASSIFICATION_SERVER • CLASSIFICATION_UNIT • CLASSIFICATION_CONNECTION • CLASSIFICATION_PROPERTY_TABLE • CLASSIFICATION_AUTH_SERVICE
QUERY_FILTER_ATTRIBUTES	Map	Required	Map of attributes identifying the criteria for matching. The key is the attribute name; the value is the value to be used in the comparison. When more than one attribute is provided, they are automatically combined within the API using the AND operator. Valid values are: • For CLASSIFICATION_DSVIEW, there are no filter attributes. • For CLASSIFICATION_SERVER, valid attributes are noted with an asterisk. For the current DSView software server's information, specify SERVER_ADDRESS_LOCALHOST. • For CLASSIFICATION_UNIT, valid attributes are noted with an asterisk. • For CLASSIFICATION_CONNECTION, valid attributes are noted with an asterisk. When querying connections, the CONNECTION_AEND, CONNECTION_ZEND or CONNECTION_OID attribute must be specified. For CLASSIFICATION_AUTH_SERVICE, valid attributes are: • AUTH_SERVICE_NAME • AUTH_SERVICE_TYPE • AUTH_SERVICE_OID

Query filter for the property table

The following table lists the query's filter for property table classification attributes.

Table 4.61 Query Filter Classification Attribute for Property Table

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_QUERY_FILTER.
QUERY_FILTER_CLASSIFICATION	String	Required	CLASSIFICATION_PROPERTY_TABLE.
QUERY_FILTER_ATTRIBUTES	Map	Required	A map of query data for matching entries owned by the calling plug-in, QUERY_FILTER_RETURN_TYPE, which specifies the desired return type for the value of each entry found. One of the following types can be specified: • RETURN_PROPERTIES • RETURN_BYTE_ARRAY If a type is not specified, RETURN_BYTE_ARRAY is assumed.

Table 4.61 Query Filter Classification Attribute for Property Table (continued)

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
	N/A	N/A	The data returned from the QUERT_FILTER_RETURN_TYPE can be one of the following values: • RETURN_STRING
(continued)			• RETURN_BYTE • RETURN_CHAR • RETURN_DOUBLE • RETURN_FLOAT • RETURN_INTEGER • RETURN_LONG • RETURN_SHORT The value returned is the equivalent Java object for the type specified. For example, RETURN_LONG returns a long object and RETURN_INTEGER returns an integer object. The following string values can be used in conjunction with the QUERY_FILTER_ATTRIBUTES plug-in and any combination of the QUERY_FILTER_RETURN_TYPES unless otherwise specified. • PROPERTY_FIND_ALL, finds all property table entries owned by the calling plug-in. The value must be PROPERTY_FIND_ALL; cannot be combined with other query attributes. • PROPERTY_ID_LIST, a list of OID values of entries in the table. This value is a list containing a long object for each OID value; cannot be combined with other query attributes. • PROPERTY_VALUE, a string value to use for finding entries that contain this string. If the value of an entry contains this string, it is considered a match. Can only be used when the QUERY_FILTER_RETURN_TYPE is RETURN_STRING or RETURN_PROPERTIES; can be combined with the PROPERTY_GROUP, PROPERTY_NAME and PROPERTY_TARGET_ID attributes. • PROPERTY_GROUP, a string value representing an optional group name to include in the query. This string value only entries that have this group name is returned; can be combined with the PROPERTY_VALUE, PROPERTY_NAME and PROPERTY_TARGET_ID attributes. • PROPERTY_NAME, a string value representing an optional name to include in the query. This string value only entries that have this name is returned; can be combined with the PROPERTY_VALUE, PROPERTY_GROUP and PROPERTY_TARGET_ID attributes. • PROPERTY_TARGET_ID, a string value representing an optional target OID to include in the query. This string value only entries that have this target OID is returned; can be combined with the PROPERTY_VALUE, PROPERTY_GROUP and PROPERTY_NAME attributes.

Search filter

The following table lists the search filter's classification attributes.

Table 4.62 Search Filter Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_SEARCH_FILTER.
SEARCH_FILTER_CLASSIFICATION	String	Required	Classification of the type of element being searched. Expected values (appended to CLASSIFICATION_) are UNIT and CONNECTION.
SEARCH_FILTER_FIELDS	List	Required	A list of the searchable fields based on the type of element being searched (see SEARCH_FILTER_CLASSIFICATION on page 158). At least one field must be defined. For CLASSIFICATION_UNIT, the following fields can be searched (appended to UNIT_): • CUSTOM_0 • CUSTOM_1 • CUSTOM_2 • SERIAL_NUMBER • FIRST_CONTACT_NAME • FIRST_CONTACT_PHONE • SECOND_CONTACT_NAME • SECOND_CONTACT_PHONE • MIGRATION_NEEDED • MOBILE_DEVICE_ACCESS • OID • NAME • ADDRESS • NOTES • DISPLAY_TYPE • ASSET_TAG • DESCRIPTION • ACCOUNT_INFO • PART_NUMBER • MODEL • CLIENT_SESSION_SESSION_ID (optional) The session ID value is the ID of a user logged into the DSView software and the ID provided by the ClientSessionService when a new session is established by the plug-in. Only the units you have access to are returned. • RIGHTS_RULE (optional) The returned units include only the units that match the rights rule provided, such as avctUnitConfigure or avctUnitControlTdPower. The rights rule pertains to the rights that the user who is specified by CLIENT_SESSION_SESSION_ID attribute must have on a unit. The CLIENT_SESSION_SESSION_ID attribute must also be provided to use the RIGHTS_RULE attribute. For the CLASSIFICATION_CONNECTION attribute, the following fields can be searched by appending the field name to CONNECTION_: • OID • AEND • AEND_LABEL • AEND_PORT • ZEND_RELATIONSHIP • ZEND • ZEND_LABEL • ZEND_PORT • IDENTIFIER
SEARCH_FILTER_STRING	String	Required	A search string you enter in the SEARCH_FILTER_FIELDS to use in a search. See Search filter string on page 159 for details.
QUERY_FILTER_ATTRIBUTES	Map	Optional	QUERY_FILTER_ATTRIBUTES (optional) A map of query data that is used in addition to the search filter string and search filter fields to find matching elements. The map values are the same values defined for the QUERY_FILTER_ATTRIBUTES of a query filter with the CLASSIFICATION_UNIT or CLASSIFICATION_CONNECTION attribute. See Query filter on page 154.

Search filter string

For all examples shown in following table, an asterisk (*) can be used before and/or after text strings as a wildcard. For example, searching for emailserver* finds elements containing emailserver in any part of the fields being searched.

Table 4.63 Search Filter String

FORMAT OF VALUE PROVIDED IN SEARCH_FILTER_ STRING	RESULTS
<String>	Finds elements that contain the specified value. For example, searching for email finds any elements that contain the string email, followed by a space or punctuation mark. When providing multiple words separated by spaces and not logical operators, OR is assumed and each word is treated separately. For example, typing **email server** finds elements containing email or server.
"<String>"	Surrounding the string with quotation marks finds elements containing the exact string, including spacing and punctuation. For example, typing **email server** finds items that contain email server. A closing quotation mark is assumed if not provided.
<String1> AND <String2>	Using the AND logical operator finds elements that contain both strings. For example, searching for email AND server will find elements named email-server-3, email-server-2, server email and so on.
<String1> OR <String2>	Using the OR logical operator finds elements that contain at least one of the strings. For example, searching for email OR server finds any items that contain the string email or the string server.
(<String>)	Parentheses can be used to override the default (left to right) order of precedence during evaluation of the search filter string. For example, searching for email AND server OR service would be the equivalent of ((email and server) or service), which cannot be the intended search. The order of precedence can be changed by grouping the search terms with parentheses, such as (email) AND (server or service).
NOT <String>	Preceding the string with NOT finds all elements that do not contain the string. For example, searching for NOT email finds all elements except those containing email, such as email, email server and email-server-1.

Plug-in setup

The following table lists the plug-in's setup classification attributes.

Table 4.64 Plug-in Setup Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_NMM_SETUP.
NMM_SETUP_DSVIEW_ ITNERFACE	DSViewInterface	Required	Instance of an object that supports the DSView software interface.

Plug-in status

The following table lists the plug-in's status classification attributes.

Table 4.65 Plug-in Status Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_NMM_STATUS.
NMM_STATUS_OPER_STATUS	String	Required	Operational status of the plug-in. Valid values (appended to NMM_STATUS_OPER_STATUS_) are: • INACTIVE • INITIALIZING • ACTIVE • SHUTTING_DOWN • UPGRADING
NMM_STATUS_OPER_STATUS_STRING	String	Optional	Detailed debugging message about the operational status of the plug-in (not displayed)
NMM_STATUS_OPER_STATUS_KEY	String	Optional	Resource key that represents the operational status of the plug-in. This key is used to look up a displayed localized string. The string should contain user-friendly information. Generally, this value should not repeat the NMM_STATUS_OPER_STATUS value; it should be used only to provide additional information, such as when a plug-in is in the active state, but has a serious problem that should be communicated. Another example is when there is useful information to display while the plug-in is initializing, shutting down or upgrading.

Plug-in upgrade data

The following table lists the plug-in's upgrade data classification attributes.

Table 4.66 Plug-in Upgrade Data Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_NMM_UPGRADE_DATA.
NMM_UPGRADE_LAST_VERSION	String	Required	Last known version of the plug-in that was running before the upgrade was called.
NMM_STATUS_OPER_STATUS_STRING	String	Optional	Detailed debugging message about the operational status of the plug-in (not displayed to users).

Plug-in consolidation data

The following table provides an overview of the attributes associated with the NMM Consolidation Data Map and their requirements.

Table 4.67 Plug-in Consolidation Data Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_NMM_CONSOLIDATION_DATA.
CONSOLIDATED_NMMS	Map	Required	A map that identifies the old plug-ins and is consolidated with the new plug-in. The key for each entry in the map is long, which represents the OID of a plug-in being consolidated. The value of each entry is the unique ID of the plug-in being consolidated.

Topology update properties

The following table lists the topology update properties' classification attributes.

Table 4.68 Topology Update Properties Classification Attributes

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_TOPOLOGY_UPDATE _PROPERTIES.
TOPOLOGY_ UPDATE_ REMOVE_ OFFLINE_ CONNECTIONS	Boolean	Optional	Indicates if offline connections should be removed; the default value is false.
TOPOLOGY_ UPDATE_ MERGE_UNITS_ BY_NAME	Boolean	Optional	Indicates if units with the same name should be merged; the default value is false.
TOPOLOGY_ UPDATE_ DELETE_UNITS_ WITH_NO_ CONNECTIONS	Boolean	Optional	Indicates if an updated unit with no connectivity should be deleted; the default value is false.
TOPOLOGY_ UPDATE_ INCLUDE_PMD _ CONNECTIONS	Boolean	Optional	Indicates if connections of type avctPmd and licensed power management device (PMD) units should be added or removed from the DSView software. When added to the DSView software, a license is reserved for the PMD unit. When removed from the DSView software, a license associated with the PMD unit is released. You can use the License Manager Service to check for the number of available power licenses before adding PMD units to the DSView software; the default value is false.

Default name settings

The following table lists the default name settings' classification attributes.

Table 4.69 Default Name Settings Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_DEFAULT_NAMES_SETTINGS.
<connection type>, such as avctSerial, avctPower, avctKvm, ...	Boolean	N/A	Indicates if units with default names should be added if associated with this connection type; the default value is false.

Data

The following table lists the data's classification attributes.

Table 4.70 Data Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_DATA.
DATA_VERSIONS	Map	Optional	Map containing the version(s), which could be the application version or boot version. The application version is also called the firmware version. The keys to this map are as follows. The value for each version is a string representing the version, such as "2.1". DATA_VERSION_APP - Key for the application version (firmware). DATA_VERSION_BOOT - Key for the boot version (firmware). Only the versions available need to be provided in this map.
DATA_REBOOT_REQUIRED	Boolean	Optional	Indicates whether a reboot is required for an appliance.

Client session map

The following table lists the client session map's classification attributes.

Table 4.71 Client Session Map Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_CLIENT_SESSION.
CLIENT_SESSION_CLASSIFICATION	String	Required	Valid values (appended to CLASSIFICATION_CLIENT_SESSION_) are: • ESTABLISH • TERMINATE • KEEP_ALIVE • IS_ALIVE
CLIENT_SESSION_USER_NAME	String	Optional	The username to be logged into the DSView software. The user is authenticated using the authentication services configured in the DSView software. See Classification attributes and values on page 137 for details. The CLIENT_SESSION_USER_NAME and CLIENT_SESSION_USER_PASSWORD attributes are optional when the CLIENT_SESSION_CLASSIFICATION value is CLASSIFICATION_CLIENT_SESSION_ESTABLISH. When establishing a client session, the HTTP request is checked for a user certificate. If a user certificate is present, an attempt is made to authenticate the user. If authentication fails with the certificate, or if a certificate is not provided, authentication using the username and password is attempted if these attributes are present.
CLIENT_SESSION_USER_PASSWORD	String	Optional	The user password to be logged into the DSView software. See Classification attributes and values on page 137 for details. The CLIENT_SESSION_USER_NAME and CLIENT_SESSION_USER_PASSWORD attributes are optional when the CLIENT_SESSION_CLASSIFICATION value is CLASSIFICATION_CLIENT_SESSION_ESTABLISH. When establishing a client session, the HTTP request is checked for a user certificate. If a user certificate is present, an attempt is made to authenticate the user. If authentication fails with the certificate, or if a certificate is not provided, authentication using the username and password is attempted if these attributes are present.
CLIENT_SESSION_HTTP_REQUEST	HttpServlet Request	Optional	The HTTP request that is used to establish the DSView software client session. This attribute is required when the CLIENT_SESSION_CLASSIFICATION value is CLASSIFICATION_CLIENT_SESSION_ESTABLISH.
CLIENT_SESSION_HTTP_RESPONSE	HttpServlet Response	Optional	The HTTP response that is used to establish the DSView software client session. This attribute is required when the CLIENT_SESSION_CLASSIFICATION value is CLASSIFICATION_CLIENT_SESSION_ESTABLISH.
CLIENT_SESSION_SESSION_ID	String	Optional	The unique session ID of a client session. This value is assigned by the DSView software when the session is established. This attribute is required when the CLIENT_SESSION_CLASSIFICATION value is one of the following: • CLASSIFICATION_CLIENT_SESSION_TERMINATE • CLASSIFICATION_CLIENT_SESSION_KEEP_ALIVE • CLASSIFICATION_CLIENT_SESSION_IS_ALIVE

Client session status map

The following table lists the client session status map's classification attributes.

Table 4.72 Client Session Status Map Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_CLIENT_SESSION_RESULTS.
CLIENT_SESSION_ SESSION_STATUS	String	Required	The status of the client session request. The client session status can have one of the following values: • CLIENT_SESSION_STATUS_ SUCCESS • CLIENT_SESSION _ STATUS_INVALID_ CREDENTIALS • CLIENT_SESSION _ STATUS_USER_ PASSWORD_EXPIRED • CLIENT_SESSION _ STATUS_ USER_ DISABLED_OR_LOCKED • CLIENT_SESSION _ STATUS _LICENSE_ LIMIT_REACHED • CLIENT_SESSION_STATUS_ PROMPTING_REQUIRED
NOTE: The plug-in API does not support multi-step authentication. For example, if you belong to the RSA SecurID external authentication service and New Pin Mode is configured, additional prompting at login is required and the authentication from the plug-in API fails. The status is CLIENT_SESSION_STATUS_PROMPTING_REQUIRED. You must log in to the DSView software and change the account settings so that multi-step authentication is not required.			
CLIENT_SESSION_ SESSION_ID	String	Optional	The client session ID generated by the DSView software after the client session is established. The client session ID can be used in subsequent requests to the DSView software. This attribute is returned only if the DSView software was able to establish a client session.
CLIENT_SESSION _AUTH_SERVICE _OID	Long	Optional	The OID associated with the authentication service that was used by the DSView software to authenticate the user while establishing the client session. This OID can be used to get the authentication service properties from the DSView software repository. This attribute is returned only if the DSView software was able to establish a client session.

Plug-in email service

The following table lists the plug-in email service's classification attributes.

Table 4.73 Plug-In Email Service Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_EMAIL_MESSAGE.
EMAIL_SERVICE_TO_ ADDRESSES	String	Required	A string containing a list of recipients that receives the email. A comma is used to separate email addresses.
EMAIL_SERVICE_FROM_ ADDRESS	String	Required	A string containing the sender's address. The plug-in is responsible for providing the correct email address.
EMAIL_SERVICE_ SUBJECT	String	Required	A string containing the subject information for the email.
EMAIL_SERVICE_BODY	String	Required	A string containing the body of the email.
EMAIL_MESSAGE_MIME_ TYPE	String	Optional	This attribute indicates the mime type for the email message. The following values are supported: • text/plain • text/html If not provided, the type is assumed to be text/plain.

Authentication service types

The following table lists the authentication service types' classification attributes.

Table 4.74 Authentication Service Type Classification Attributes

AUTH SERVICE TYPE	VALUE TYPE	DESCRIPTION
AUTH_SERVICE_ TYPE_INTERNAL	String	This represents the internal authentication service provided by the DSView software. It is based on the authentication of the username and password. There is no additional required DSView software configuration.
AUTH_SERVICE_ TYPE_LDAP	String	This represents the external LDAP authentication service that is used to authenticate a user, which must be configured by a DSView software administrator.
AUTH_SERVICE_ TYPE_ACTIVE_ DIRECTORY	String	This represents the external Active Directory authentication service that is used to authenticate a user, which must be configured by a DSView software administrator.
AUTH_SERVICE_ TYPE_RADIUS	String	This represents the external RADIUS authentication service that is used to authenticate a user, which must be configured by a DSView software administrator.
AUTH_SERVICE_ TYPE_NT	String	This represents the external NT authentication service that is used to authenticate a user, which must be configured by a DSView software administrator.
AUTH_SERVICE_ TYPE_TACACS	String	This represents the external TACACS authentication service that is used to authenticate a user, which must be configured by a DSView software administrator.

Property table map

The following table lists the property table map's classification attributes.

Table 4.75 Property Table Map Classification Attribute

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS	
CLASSIFICATION	String	Required	CLASSIFICATION_PROPERTY_TABLE.	
PROPERTY_ID	Long	Required	The OID of the property table entry. This is a unique value provided by the DSView software. The value is automatically determined by the DSView software when a new entry is added to the Property Table.	
PROPERTY_ NAME	String	Required	The name of the property. The owner (which is always the plug-in), the name and the target ID together uniquely identify an entry.	
PROPERTY_ TARGET_ID	Long	(Optional) If not provided, the plug-in owner is automatically assigned to the target OID.	The OID of a target that the property is associated with; provides a means of associating a property with another element in the repository. The target OID can be the OID of any of the following types of repository elements: Unit, Connection or DSView software server.	
PROPERTY_ GROUP	String	Optional	A group name that is associated with the property. Can be any String value the plug-in wants to assign. This attribute can be used to help categorize entries in the Property Table or can be used to help facilitate searching.	
PROPERTY_ VALUE	Object	Required	The value of the property. The value can be any of the following Java Objects:	
			• String • Properties • byte[] • Byte • Character • Double • Float	• Integer • Long • Number • Short • BigDecimal • BigInteger

Web request map

The following table lists the web request map's classification attributes.

Table 4.76 Web Request Map Classification Attributes

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_WEB_REQUEST.
WEB_REQUEST_SERVLET_REQUEST	HttpServlet Request	Required	The HttpServletRequest object.
WEB_REQUEST_SERVLET_RESPONSE	HttpServlet Response	Required	The HttpServletResponse object.

Authentication service map

The following table lists the authentication service map's classification attributes.

Table 4.77 Authentication Service Map Classification Attributes

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_AUTH_SERVICE.
AUTH_SERVICE_OID	Long	Optional	The OID of the authentication service table entry. This is a unique value provided by the DSView software.
AUTH_SERVICE_NAME	String	Optional	The name of the service.
AUTH_SERVICE_TYPE	String	Optional	The authentication service type; can be one of the following values: • AUTH_SERVICE_TYPE_ACTIVE_DIRECTORY
AUTH_SERVICE_DOMAIN	String	Optional	The domain for the authentication service.
AUTH_SERVICE_USER_FOLDER	String	Optional	The folder where users are located on the authentication server.
AUTH_SERVICE_GROUP_FOLDER	String	Optional	The folder where user groups are located on the authentication server.
AUTH_SERVICE_USERNAME_PREFERENCE	String	Optional	The preference for how usernames are to appear when names are browsed; can be one of the following values: • AUTH_SERVICE_USERNAME_TYPE_FULL_WIN2K • AUTH_SERVICE_USERNAME_TYPE_PARTIAL_WIN2K • AUTH_SERVICE_USERNAME_TYPE_FULL_PREWIN2K • AUTH_SERVICE_USERNAME_TYPE_PARTIAL_PREWIN2K
AUTH_SERVICE_SSL_MODE	String	Optional	Indicates the SSL mode for communicating with the authentication server; can be one of the following values: • AUTH_SERVICE_SSL_MODE_NONE • AUTH_SERVICE_SSL_MODE_TRUST_ALL • AUTH_SERVICE_SSL_MODE_CERTIFICATE
AUTH_SERVICE_KERBEROS	String	Optional	The value indicates if Kerberos should be used for user authentication; the value can be true or false.
AUTH_SERVICE_GLOBAL_CATALOG	String	Optional	The value indicates if the global catalog should be used; the value can be true or false.
AUTH_SERVICE_REFERRAL_CHASING	String	Optional	The value indicates if referral chasing should be enabled; the value can be true or false.
AUTH_SERVICE_BROWSE_MODE	String	Optional	Indicates the browse mode to use when browsing for users or user groups; the value can be one of the following: • AUTH_SERVICE_BROWSE_MODE_ANONYMOUS • AUTH_SERVICE_BROWSE_MODE_CREDENTIALS
AUTH_SERVICE_CERTIFICATE_CHAIN	X509 Cert.	Optional	This value is an array of type java.security.cert.X509Certificate containing the certificate chain that identifies the authentication server. This value can optionally be returned when the AUTH_SERVICE_SSL_MODE has a value of AUTH_SERVICE_SSL_MODE_CERTIFICATE.

Operation map

The following table lists the operation map's classification attributes.

Table 4.78 Operation Map Classification Attributes

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_OPERATION.
OPERATION_ CLASSIFICATION	String	Required	Can be one of the following values: • CLASSIFICATION_OPERATION_GET_LIST • CLASSIFICATION_OPERATION_EXECUTE • CLASSIFICATION_OPERATION_DATA
OPERATION_ID	String	Required	A unique ID for an operation that is either supported by the DSView software or by plug-ins. The plug-in can obtain the list of operations that are supported by the DSView software or by plug-ins using the getOperationList method of the Operation Service. This attribute is required when the OPERATION_CLASSIFICATION attribute is one of the following values: • CLASSIFICATION_OPERATION_GET_LIST • CLASSIFICATION_OPERATION_EXECUTE • CLASSIFICATION_OPERATION_DATA
OPERATION_ NAME	String	Optional	The name of the operation. The plug-in can display the operation name. The operation name is in the locale that is reported by either the DSView software or a plug-in. This attribute is required when the OPERATION_CLASSIFICATION attribute is one of the following values: • CLASSIFICATION_OPERATION_GET_LIST • CLASSIFICATION_OPERATION_DATA
CLIENT_ SESSION_ SESSION_ID	String	Required	A unique ID for a client session. This value is the client session ID of a user who has logged into the DSView software using the Client Session Service. The client session ID is used by the DSView software to determine if the logged in user has sufficient access rights to execute the operation on a given entity. This attribute is required when the OPERATION_CLASSIFICATION attribute is one of the following values: • CLASSIFICATION_OPERATION_GET_LIST • CLASSIFICATION_OPERATION_EXECUTE • CLASSIFICATION_OPERATION_DATA

Operation status map

The following table lists the operation status map classification attributes.

Table 4.79 Operation Status Map Classification Attributes

ATTRIBUTE	VALUE TYPE	REQUIREMENT	VALUE SET/RESTRICTIONS
CLASSIFICATION	String	Required	CLASSIFICATION_OPERATION_STATUS.
OPERATION_STATUS	String	Required	The status after executing an operation. The operation status can have one of the following values: • OPERATION_STATUS_SUCCESS • OPERATION_STATUS_FAILURE • OPERATION_STATUS_UNKNOWN_OPERATION • OPERATION_STATUS_INVALID_USER • OPERATION_STATUS_INSUFFICIENT_RIGHTS • OPERATION_STATUS_NO_POWER_ CONNECTION • OPERATION_STATUS_NO_IPMI_CONNECTION • OPERATION_STATUS_NOT_SUPPORTED

4.8 Customizing DSView Software Functionality

4.8.1 Unit name

The DSView™ and Rack Power Manager software display the unit name in the Unit Overview Screen. A plug-in can require the name to be read-only by updating the following property in the nmm.xml file.

Table 4.80 Unit Name Property in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctDBFieldName	read-only	This property indicates a unit name is read-only.

Example: Unit Element with a read-only name field

```
<element elementId="sam1000" classification="avctUnit"
 scope="avctLocal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctDBFieldName" value="read-only" />
 </properties>
```

4.8.2 Merging target devices with the same name

The DSView™ and Rack Power Manager software can merge target devices with the same name into a single target device during the Resync Unit Wizard or Add Unit Wizard. This checkbox appears on the Select Options screen of the wizards. If the plug-in disables the Select Options screen, it can define if merging target devices with the same name is enabled or disabled by updating the following property in the nmm.xml file.

Table 4.81 Property for Merging Target Devices in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctDiscoverAllowSameName	Enabled or Disabled	If the Select Options screen is hidden, this property specifies if merging target devices with the same name are enabled or disabled.

Example: Unit Element that enables merging target devices with identical names

```
<element elementId="sam1000" classification="avctUnit"
 scope="avctLocal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key=" avctDiscoverAllowSameName" value="enabled" />
 </properties>
```

4.8.3 Remove offline connections

The DSView™ and Rack Power Manager software can remove offline connections during the Resync Unit Wizard or Add Unit Wizard. This checkbox appears on the Select Options screen of the wizards. If the plug-in disables the Select Options screen, it can define if removing offline connections is enabled or disabled by updating the following property in the nmm.xml file.

Table 4.82 Property for Removing Offline Connections in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctRemoveOfflineConn	Enabled or Disabled	If the Select Options screen is hidden, this property specifies if removing offline connections is enabled or disabled.

Example: Unit element that enables removing offline connections

```
<element elementId="sam1000" classification="avctUnit"
 scope="avctLocal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key=" avctRemoveOfflineConn" value="enabled" />
 </properties>
```

4.8.4 Unit network address

The DSView™ and Rack Power Manager software display the unit address in the Unit Network Properties screen. A plug-in can require the address to be read-only by updating the following property in the nmm.xml file.

Table 4.83 Unit Address Property in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctDBFieldAddress	read-only	This property indicates a unit address is read-only.

Example: Unit element with a read-only address field

```
<element elementId="sam1000" classification="avctUnit"
 scope="avctLocal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctDBFieldAddress" value="read-only" />
 </properties>
```

4.8.5 Default names for target devices

The DSView™ and Rack Power Manager software display the option “Allow target devices that contain default names to be added for these type of connection(s)” in the Add Unit and Resync wizards. A plug-in can specify to hide this option for a connection element that does not support default names for target devices. To hide the option, add the following property value to the connection element in the nmm.xml file.

Table 4.84 Property for Hiding Default Names Option in the DSView™ and Rack Power Manager software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_ CONNECTION	avctDefaultNames	false	The plug-in must set the value of this property to false and the scope of the connection element must be set to avctLocal.

In the following example, the connection element "samlink" has a local scope and does not support default names for target devices that are associated with the connection element.

Example: Connection element “samlink” with local scope

```
<element elementId="samlink"
 classification="avctConnection" scope="avctLocal"
 type="samlink"
label="unit.samlink.label"
navigationSetId="samlink"
 imageSetId="samlink">

 <properties>
 <property key="avctDefaultNames" value="false" />
 </properties>

 </element>
```

4.8.6 KVM profile

The DSView™ and Rack Power Manager software KVM Profile screen allow you to assign a KVM profile to a generic target device. By default, this screen is disabled for all plug-in defined unit types. You can enable a KVM Profile screen for a unit defined by the plug-in by adding the following property value in the nmm.xml file.

Table 4.85 Property for Enabling the KVM Profile Screen

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctKvmProfile	Enabled or Disabled	Enables or disables the KVM Profile screen for the specified unit. The default value is Disabled.

NOTE: Enabling the KVM Profile screen also enables the Profile section of the multi-unit properties operation (available from the Operations menu in a Units View screen).

Example: Unit element "genericd" with the KVM profile screen enabled

```
<element elementId="MyCard" classification="avctUnit"
 scope="avctLocal" type="MyCard"
 category="avctCategoryCard,avctCategoryTargetDevice"
 label="mycard.label" >

 <properties>
 <property key="avctKvmProfile" value="enabled" />
 </properties>

 </element>
```

4.8.7 Unit connections

Attach device to unit connection

The DSView™ and Rack Power Manager software allow you to attach a target unit to an available unit connection from the Units View screens. A plug-in can prevent a unit element from being attached to an available unit connection by updating the following property value in the nmm.xml file.

Table 4.86 Property for Attaching Devices to Unit Connections in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctUnsupportedOpList	avctAttachDevice	This property contains a comma-separated list of operations or tools.

Example: Unit element "sam1000" with the attach device operation disabled

```
<element elementId="sam1000" classification="avctUnit"
 scope="avctLocal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctUnsupportedOpList"
 value="avctAttachDevice" />
 </properties>

 </element>
```

Overwrite unit type and unit categories for attached target unit

When you attach a target unit to a connection, the plug-in can define both the unit type and unit categories for the target. The DSView™ and Rack Power Manager software updates both the unit type and unit categories for the target after the attach target unit operation is complete. The plug-in must provide the following extended properties for the unit connection.

Table 4.87 Extended Unit Type and Unit Category Properties in the DSView Software

ELEMENT CLASSIFICATION	ATTRIBUTE	EXTENDED PROPERTY	DESCRIPTION
CLASSIFICATION_CONNECTION	CONNECTION_EXTENDED_PROPERTIES	avctTargetUnitType	The plug-in must set the value of this property to the unit type to be used for a unit to be attached to a unit connection that is defined by the plug-in.
CLASSIFICATION_CONNECTION	CONNECTION_EXTENDED_PROPERTIES	avctTargetUnitCategory	The plug-in must set the value of this property to the unit categories to be used for a unit to be attached to a unit connection that is defined by the plug-in. Each unit category must be separated by a comma delimiter.
CLASSIFICATION_CONNECTION	CONNECTION_EXTENDED_PROPERTIES	avctTargetUnitScope	The plug-in must set the value of this property to the scope that is defined for the unit type specified by the avctTargetUnitType extended property.

Resync unit connection

The DSView™ and Rack Power Manager software synchronize unit connections from the Resync Wizard, Auto Topology Update feature and Topology Update task. In addition, unit connections can be synchronized from a plug-in using the Update Topology feature of the repository service. At run time, a plug-in can update the unit connection element to prevent the synchronization of a unit connection. The plug-in must provide the following extended property.

Table 4.88 Property for Resyncing Unit Connections in the DSView Software

ELEMENT CLASSIFICATION	ATTRIBUTE	EXTENDED PROPERTY	DESCRIPTION
CLASSIFICATION_CONNECTION	CONNECTION_EXTENDED_PROPERTIES	avctResync	If the property value is false, the unit synchronization is prevented. If the property value is true or the property is not provided, unit synchronization is permitted.

Delete unit connection

When a unit is deleted from the Units View screen, the unit connection remains so that another unit can be attached to it. At run time, a plug-in can update the unit connection element to require the DSView™ and Rack Power Manager software to delete the unit connection. The plug-in must provide the following extended property.

Table 4.89 Extended Property for Deleting Unit Connections in the DSView Software

ELEMENT CLASSIFICATION	ATTRIBUTE	EXTENDED PROPERTY	DESCRIPTION
CLASSIFICATION_CONNECTION	CONNECTION_EXTENDED_PROPERTIES	avctDeleteConnection	If the property value is true, the unit connection is deleted. If the property value is false or the property is not provided, the unit connection is not deleted.

4.8.8 Operations and Tools

Pull names from appliance

The DSView™ and Rack Power Manager software display the Pull Names from Appliance operation in the Operations menu of the Units View screen. A plug-in can indicate that this operation should be hidden by updating the following property value in the nmm.xml file.

Table 4.90 Property for Hiding Pull Names Operation in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctUnsupportedOpList	avctPullNames	This property contains a comma-separated list of operations or tools.

Example: Unit element “sam1000” with the pull names from appliance operation disabled

```
<element elementId="sam1000" classification="avctUnit"
 scope="avctLocal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

<properties>
 <property key="avctUnsupportedOpList"
 value="avctPullNames" />
 </properties>

 </element>
```

Show versions

The DSView™ and Rack Power Manager software display the Show Versions operation in the Operations Menu of the Units View. A plug-in can indicate that this operation should be hidden by updating the following property value in the nmm.xml file.

Table 4.91 Property for Hiding Show Versions Operation in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctUnsupportedOpList	avctShowVersions	This property contains a comma-separated list of operations or tools.

Example: Unit element “sam1000” with the show versions operation disabled

```
<element elementId="sam1000" classification="avctUnit"
scope="avctLocal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctUnsupportedOpList"
 value="avctShowVersions" />
 </properties>

</element>
```

Resync

The DSView™ and Rack Power Manager software display the Resync button on the Unit Overview screen. A plug-in can indicate that this button should be hidden by updating the following property value in the nmm.xml file.

Table 4.92 Property for Hiding Resync Button in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctUnsupportedOpList	avctResync	This property contains a comma-separated list of operations or tools.

Example: Unit element “sam1000” without the Resync button

```
<element elementId="sam1000" classification="avctUnit"
 scope="avctLocal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctUnsupportedOpList"
value="avctResync" />
 </properties>

</element>
```

You can remove the support for Pull Names, Show Versions and Resync at the same time by listing each property value separated by a comma.

Example: Resync, pull names and show versions operations disabled

```
<properties>
<property key="avctUnsupportedOpList"
value="avctPullNames,avctShowVersions,avctResync" />
</properties>
```

Merge Target Devices

The DSView™ and Rack Power Manager software display the Merge Target Devices tool in the target device Overview Screen. A plug-in can indicate that a target device defined by the plug-in does not support merging by updating the following property value in the nmm.xml file.

Table 4.93 Property for Hiding Merge Target Device Tool in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_ UNIT	avctUnsupportedOpList	avctMergeTargetDevice	This property contains a comma-separated list of operations or tools.

Example: Unit element “sam1000” with the merge target devices operation disabled

```
<element elementId="sam1000" classification="avctUnit"
 scope="avctLocal" category="avctCategorySoftware,avctCategoryTargetDevice"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctUnsupportedOpList"
 value="avctMergeTargetDevice" />
 </properties>

 </element>
```

Remove offline connections

The DSView™ and Rack Power Manager software display the Remove offline connections operation during the Resync Unit Wizard in the Select Options Screen. A plug-in can indicate that this operation is unsupported by updating the following property value in the nmm.xml file.

Table 4.94 Removing Support for Remove Offline Connections Tool in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_ UNIT	avctUnsupportedOpList	avctRemoveOfflineConn	This property contains a comma-separated list of operations or tools.

Example: Unit element “sam1000” with the remove offline connections operation disabled

```
<element elementId="sam1000" classification="avctUnit"
 scope=" avctGlobal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctUnsupportedOpList"
 value="avctRemoveOfflineConn" />
 </properties>

 </element>
```

Allow target devices with the same name to be merged into a single target device

The DSView™ and Rack Power Manager software display the Allow target devices with the same name to be merged into a single target device operation during the Add Unit Wizard and the Resync Unit Wizard in the Select Options Screen. A plug-in can indicate this operation is unsupported by updating the following property value in the nmm.xml file.

Table 4.95 Removing Support for Merge Target Devices Tool in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_ UNIT	avctUnsupportedOpList	avctDiscoverAllowSameName	This property contains a comma-separated list of operations or tools.

Example: Unit element “sam1000” with the allow target devices operation disabled

```
<element elementId="sam1000" classification="avctUnit"
 scope=" avctGlobal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctUnsupportedOpList"
 value="avctDiscoverAllowSameName" />
 </properties>

 </element>
```

Resync Unit Wizard Select Options screen

The Resync Unit Wizard displays the Select Options screen by default. A plug-in can require that the Select Options screen is not displayed in the Resync Unit Wizard by updating the following property value in the nmm.xml file.

Table 4.96 Hiding the Select Options screen in the Resync Wizard

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_ UNIT	avctUnsupportedOpList	avctResyncSelectOptions	This property contains a comma-separated list of operations or tools.

NOTE: If the Select Options screen is disabled in the Resync Unit Wizard, the DSView™ and Rack Power Manager software uses the values defined in the avctRemoveOfflineConn and avctDiscoverAllowSameName properties, if defined. If a property is not defined for these values, the DSView™ and Rack Power Manager software uses the globally defined options for the wizard.

Example: Unit element “sam1000” with the Resync Select options operation disabled

```
<element elementId="sam1000" classification="avctUnit"
 scope=" avctGlobal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctUnsupportedOpList"
 value="avctResyncSelectOptionsPage" />
 </properties>

 </element>
```

Add Unit Wizard Select Options screen

The Add Unit Wizard displays the Select Options screen by default. A plug-in can require that the Select Options screen is not displayed in the Add Unit Wizard by updating the following property value in the nmm.xml file.

Table 4.97 Hiding the Select Options Screen in the Add Unit Wizard

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctUnsupportedOpList	avctAddUnitSelectOptions	This property contains a comma-separated list of operations or tools.

NOTE: If the Select Options screen is disabled in the Add Unit Wizard, the DSView™ and Rack Power Manager software uses the values defined in the avctDiscoverAllowSameName property, if defined. If a property is not defined for this value, the DSView™ and Rack Power Manager software uses the globally defined options for the wizard.

Example: Unit element “sam1000” with the add unit select options screen disabled

```
<element elementId="sam1000" classification="avctUnit"
 scope=" avctGlobal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctUnsupportedOpList"
 value="avctAddUnitSelectOptionsPage" />
 </properties>

 </element>
```

Add Unit Select Procedure Screen

The DSView™ and Rack Power Manager software display the Add Unit Select Procedure screen. If the Add Unit Select Procedure screen is not displayed, the DSView™ and Rack Power Manager software defaults to the procedure for discovering a single unit. A plug-in can indicate that this screen should not be displayed in the Add Unit Wizard by updating the following property value in the nmm.xml file.

Table 4.98 Property for Hiding Add Unit Select Procedure Screen in the DSView Software

ELEMENT CLASSIFICATION	PROPERTY KEY	PROPERTY VALUE	DESCRIPTION
CLASSIFICATION_UNIT	avctUnsupportedOpList	avctAddUnitSelectProcPage	This property contains a comma-separated list of operations or tools.

Example: Unit element “sam1000” with the add unit select procedure screen disabled

```
<element elementId="sam1000" classification="avctUnit"
 scope="avctLocal"
 type="sam1000" label="unit.sam1000.label"
 navigationSetId="sam1000" imageSetId="sam1000">

 <properties>
 <property key="avctUnsupportedOpList"
 value="avctAddUnitSelectProcPage" />
 </properties>

 </element>
```

Push Names To Appliance

The DSView™ and Rack Power Manager software display the Push Names To Appliance operation in the Operations Menu of the Units View. To indicate that this operation should be hidden, remove the pushNames method from the ConfigManagementInterface in the supportedInterfaces section in the nmm.xml file.

Reboot

The DSView™ and Rack Power Manager software display the Reboot operation in the Operations Menu and the Reboot button in the Unit Overview screen. A plug-in can indicate that this operation/button should be hidden using the following steps.

To hide the Reboot operation/button:

1. In the nmm.xml file, remove the access right that allows you to reboot a unit.

Example: Unit element reboot access right

```
<rights>
 <right name="avctUnitReboot" />
 </rights>
```

2. In the class that implements the Operation interface, indicate that the reboot operation is not supported.

Example: Reboot operation not supported by the plug-in

```
public boolean isOperationSupported( String szOperationId ) throws DSViewException
 {
 if ( szOperationId.equals( DSViewConstantsInterface.OPERATION_REBOOT ) )
 {
 return ( false );
 }
 return ( false );
 }
```

Upgrade Firmware

The DSView™ and Rack Power Manager software display the Upgrade Firmware button in the Unit Overview screen. A plug-in can indicate that this button should be hidden using the following steps.

To hide the Upgrade Firmware button:

1. In the nmm.xml file, remove any elements for files whose element category is avctFirmware.

Example: avctFirmware element

```
<element elementId="myFirmwareType" classification="avctFile"
 scope="avctGlobal" category="avctFirmware" type="localType1"
 family="34" oem="0" label="file.myFirmwareType.label">
 </element>
```

2. In the nmm.xml file, remove any supported files that are referenced by elements whose category is avctFirmware.

Example: avctFirmware element with a supported file

```
<supportedFileTypes>
<fileType type="localType1" scope="avctGlobal" />
 </supportedFileTypes>
```

Save and Restore Configuration

The DSView™ and Rack Power Manager software display the Save Configuration and Restore Configuration button in the Unit Overview screen. A plug-in can indicate that these buttons should be hidden using the following steps.

To hide the Save Configuration and Restore Configuration buttons:

1. In the nmm.xml file, remove any elements for files whose element category is avctConfig.

Example: avctConfig element

```
<element elementId="myConfigType" classification="avctFile"
 scope="avctLocal" category="avctConfig" type="localType3"
 label="file.myConfigType.label">
 </element>
```

2. In the nmm.xml file, remove any supported files that are referenced by elements whose category is avctConfig.

Example: avctConfig element with a supported file

```
<supportedFileTypes>
 <fileType type="localType3" />
 </supportedFileTypes>
```

NOTE: If support for the Save Configuration and Restore Configuration buttons is removed, you must also remove support for the Pull Names and Push Names operations.

Save and Restore User Database

The DSView™ and Rack Power Manager software display the Save User Database and Restore User Database buttons in the Unit Overview screen. A plug-in can indicate that these tools should be hidden using the following steps.

To hide the Save User Database and Restore User Database buttons:

1. In the nmm.xml file, remove any elements for files whose element category is avctUserConfig.

Example: avctUserConfig element

```
<element elementId="myUserConfigType1" classification="avctFile"
 scope="avctLocal" category="avctUserConfig" type="localType2"
label="file.myUserConfigType.label">
 </element>
```

2. In the nmm.xml file, remove any supported files that are referenced by elements whose category is avctUserConfig.

Example: avctUserConfig element with a supported file

```
<supportedFileTypes>
 <fileType type="localType2" />
 </supportedFileTypes>
```

NOTE: If support for the Save User Database and Restore User Database buttons is removed, you must also remove support for the Pull Names and Push Names operations.

Save configuration template and appliance replacement

The DSView™ and Rack Power Manager software display the Save Configuration Template and Appliance Replacement tools in the Unit Overview screen. A plug-in can indicate that these tools should be hidden using the following steps.

To hide the Save Configuration Template and Appliance Replacement tools:

1. In the nmm.xml file, remove any elements for files whose element category is avctConfigTemplate.

Example: avctConfigTemplate element

```
<element elementId="myConfigTemplateType" classification="avctFile"
 scope="avctLocal" category="avctConfigTemplate" type="localType4"
 label="file.myConfigTemplateType.label">
 </element>
```

2. In the nmm.xml file, remove any supported files that are referenced by elements whose category is avctConfigTemplate.

Example: avctConfigTemplate element with a supported file

```
<supportedFileTypes>
 <fileType type="localType4" />
 </supportedFileTypes>
```

NOTE: If support for the Save User Database and Restore User Database buttons is removed, you must also remove support for the Pull Names and Push Names operations.

4.9 Plug-in JAR File

A plug-in image is packaged in a standard JAR file. The JAR files are loaded into the DSView software, then if all licensing and signature requirements are met, the plug-in functionality is activated.

The following table describes the structure of a plug-in JAR file. No additional files can be stored in or be members of the JAR file.

Table 4.99 JAR File Structure

FILE/DIRECTORY	TYPE	DESCRIPTION
nmm.xml	File	Plug-in definition file that defines the functionality and capabilities of the plug-in.
validation.xml	File	Plug-in validation file that defines the basic prompt level validations for the plug-in screens.
jars	Directory	Contains all JAR files needed by the plug-in.
jars/*.jar	File(s)	JAR files needed by the plug-in. These files contain the implementations of the plug-in interfaces, plus other needed implementations.
res	Directory	Contains all resource property files that contain properties referenced by the nmm.xml file.
res/*.properties	File(s)	Property files needed by the plug-in; must follow local naming conventions.
images	Directory	Contains image files referenced by the nmm.xml file.
images/*.gif	File(s)	Image files needed by the plug-in.
help	Directory	Contains help files associated with the plug-in. The help files must be fully self-contained. They cannot reference other plug-in resources, such as the images directory. All images needed for the help should be included in this directory/subdirectories, with appropriate references. Subdirectories can be provided to support other languages. The en directory is required for U.S. English.
help/en	Directory	Contains all help files for U.S. English.
help/en/index.html	File	Entry point for help on the plug-in.
META-INF	Directory	Standard JAR directory for containing the manifest file.
META-INF/MANIFEST.MF	File	Manifest file of the plug-in JAR.
swf	Directory	Directory containing all Adobe Flash swf files that are referenced by the nmm.xml file.
swf/*.swf	File(s)	Adobe swf files (Flash movies) needed by the plug-in for Flex screens used in the Flex Screen decks.

4.10 Access Rights

The following tables contain an overview of the access rights supported by the DSView software. An X indicates the role is granted that access right. A dash (-) indicates the role is not granted that access right.

Table 4.100 System Level Access Rights

SYSTEM LEVEL RIGHTS	DSV ADMIN	USER ADMIN	APP ADMIN	USER	AUDITOR	NONE/EVERYONE
avctDomainSystemSettingsEdit	X	-	-	-	-	-
avctDomainSystemsettingsView	X	-	-	-	-	-
avctDomainSiteDeptLocEdit	X	-	X	-	-	-
avctDomainBackup	X	-	-	-	-	-
avctDomainUserEdit	X	X	-	-	-	-
avctDomainUserView	X	X	-	-	-	-
avctDomainUserGroupEdit	X	X	-	-	-	-
avctDomainUserGroupView	X	X	-	-	-	-
avctDomainAuthServersEdit	X	X	-	-	-	-
avctDomainAuthServersView	X	X	-	-	-	-
avctDomainAuditLogEdit	X	-	-	-	X	-
avctDomainAuditLogView	X	-	-	-	X	-
avctDomainUnitEdit	X	-	X	-	-	-
avctDomainUnitView	X	X	X	X	-	-
avctDomainUnitGroupEdit	X	X	X	-	-	-
avctDomainUnitGroupView	X	X	X	-	-	-
avctDomainFirmwareEdit	X	-	X	-	-	-
avctDomainFiremwareView	X	-	X	-	-	-
avctDomainTasksEdit	X	X	X	X	-	-
avctDomainTasksView	X	X	X	X	-	-
avctDomainTasksViewAll	X	-	-	-	-	-
avctDomainAclEdit	X	X	-	-	-	-
avctDomainAclView	X	X	-	-	-	-
avctDomainConfiguratinFileEdit	X	-	X	-	-	-
avctDomainConfigurationFileView	X	-	X	-	-	-
avctDomainUserFileEdit	X	X	X	-	-	-
avctDomainUserFileView	X	X	X	-	-	-

Table 4.101 Default Unassigned Unit Group Rights

SYSTEM LEVEL RIGHTS	DSV ADMIN	USER ADMIN	APP ADMIN	USER	AUDITOR	NONE/EVERYONE
avctUnitView *	X	X	X	-	-	-
avctUnitReboot	X	X	X	-	-	-
avctUnitDisconnectSession	X	X	X	-	-	-
avctUnitFlash	X	-	X	-	-	-
avctUnitConfigure	X	-	X	-	-	-
avtUnitAdministerLocalAccts	X	X	X	-	-	-
avctUnitEstablishTdSession	X	X	X	-	-	-
avctUnitControlTdPower	X	X	X	-	-	-
avctUnitEstablishVmSession	X	X	X	-	-	-
avctUnitEstablishReservedVmSession	X	X	X	-	-	-
avctUnitViewDataLogging	X	X	X	-	-	-
The avctUnitView right is a special right. It is determined dynamically for units. Essentially, you have the avctUnitView right if you have the avctDomainUnitEdit, avctDomainAclEdit or any other avctUnit right.						

Table 4.102 Rights for Units Nodes

NODE	READ-ONLY RIGHT	READ-WRITE RIGHT
Units tab	avctDomainUnitView	avctDomainUnitEdit
Overview	avctUnitView	avctDomainUnitEdit or avctUnitConfigure
Properties	avctUnitView	avctDomainUnitEdit or avctUnitConfigure
Access rights	avctDomainAclView	avctDomainAclEdit
Appliance settings *	avctUnitView	avctUnitConfigure
OSCAR auth settings	avctUnitVeiw	avctUnitAdministerLocalAccts
CLI auth settings	avctUnitView	avctUnitAdministerLocalAccts
Local accounts	avctUnitView	avctUnitAdministerLocalAccts
Connections	avctUnitView	avctDomainUnitEdit
Save/restore configuration database	-	avctUnitConfigure
Save/restore user database	-	avctUnitAdministerLocalAccts
Reboot	-	avctUnitReboot
Upgrade firmware	-	avctUnitFlash
Resync **	-	avctUnitConfigure and avctDomainUnitEdit
* Appliance settings include all nodes under appliances other than OSCAR, CLI and local accounts. ** Resync requires both avctUnitConfigure and avctDomainUnitEdit rights because you can delete units during a Resync Operation.		

This page intentionally left blank.

VertivCo.com | Vertiv Headquarters, 1050 Dearborn Drive, Columbus, OH, 43085, USA

590-1796-501C