Getting Started with the Web Client

The OmniVista Web Client for OmniVista provides users access to basic versions of the OmniVista Locator, Notifications, and Topology applications through a web application residing on the OmniVista Server. The following functionality is available:

- **Locator** - Search for an end station by either IP or MAC address and browse to locate an end station
- **Notifications** - View traps sent to OmniVista by the switches
- **Topology** - View information for all switches managed by OmniVista.

Note: OmniVista provides a **Web Services API** that customers can use to write their own web applications. Go to “Web Services API” on page 14 to view Web Services API document.

Requirements

The Web Client is installed on the OmniVista Server and can be accessed through the following web browsers:

- Windows - Internet Explorer Versions 6.0 and 7.0
- Linux - Firefox Versions 1.5, 2.0, and 3.0.
- Sun Solaris - Firefox Versions 1.5 and 2.0.

Your browser preferences (or options) should be set up as follows:

- Cookies should be enabled. Typically, this is the default.
- JavaScript must be enabled/supported.
- Java must be enabled.
- Style sheets must be enabled; that is, the colors, fonts, backgrounds, etc., of web pages should always be used (rather than user-configured settings).
- Checking for new versions of pages should be set to "Every time" your browser opens.

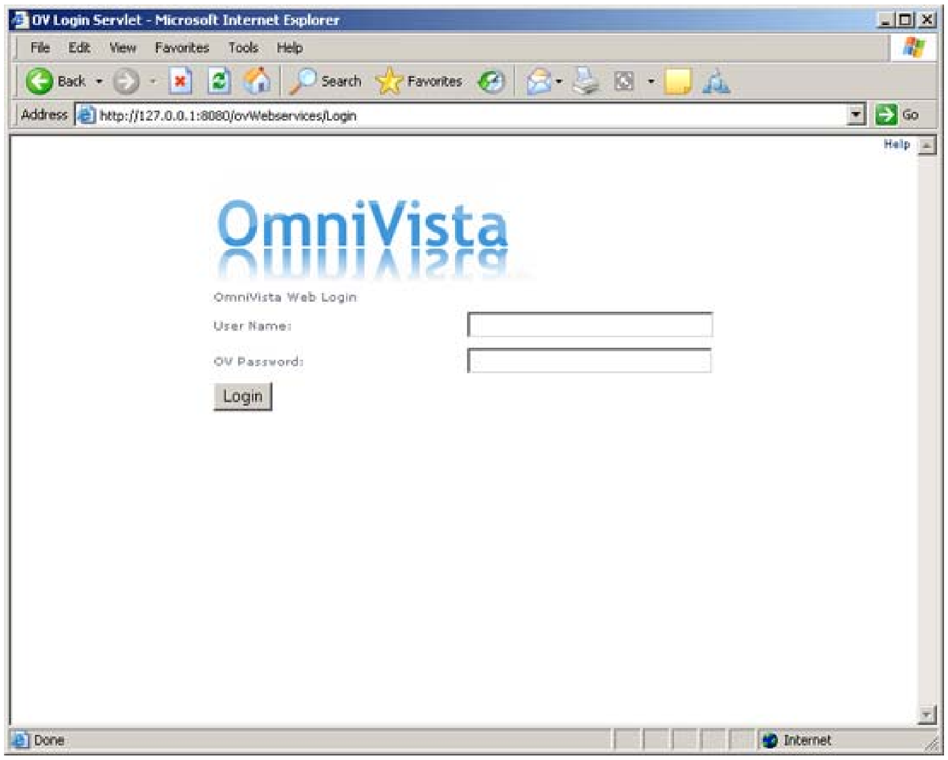
Note: Typically, many of these settings are configured as the default. Different browsers (and different versions of the same browser) may have different dialogs for these settings. Check your browser help pages if you need help.

Logging Into OmniVista Web Client

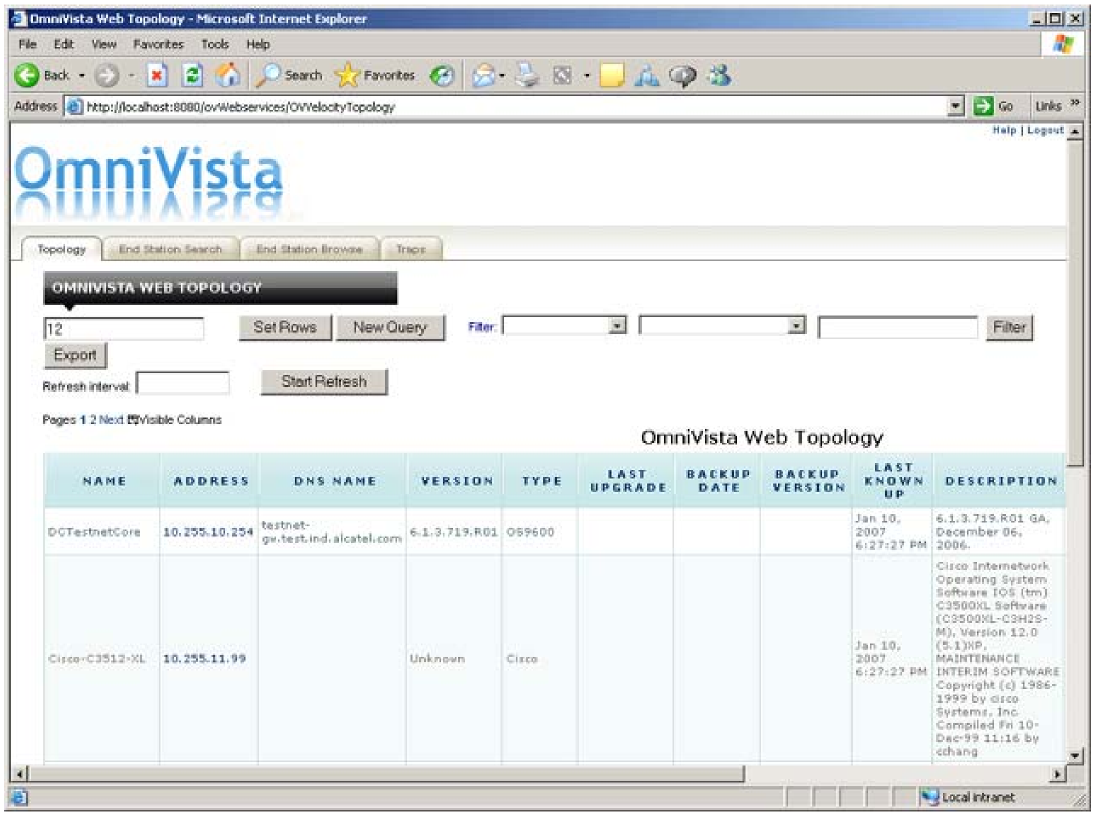
To access the OmniVista Web Client:

1. Open an Internet browser and enter **http://***OmniVista Server IP Address***:8080** in the address line, then press **ENTER**. The Login Screen will appear.

Note: If the client and server are installed on the same machine, you can enter **http://localhost:8080**.


OV Login Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://127.0.0.1:8080/ovWebservices/Login
Go
Help
OmniVista
OmniVista Web Login
User Name:
OV Password:
Login
Done
Internet

2. Enter the OmniVista user name and password and click **Login**. The following screen will appear.



OmniVista Web Topology

NAME	ADDRESS	DNS NAME	VERSION	TYPE	LAST UPGRADE	BACKUP DATE	BACKUP VERSION	LAST KNOWN UP	DESCRIPTION
DCTestnetCore	10.255.10.254	testnet-gw.test.ind.alcatel.com	6.1.3.719.R01	OS9600				Jan 10, 2007 6:27:27 PM	6.1.3.719.R01 GA, December 06, 2006.
Cisco-C3512-XL	10.255.11.99		Unknown	Cisco				Jan 10, 2007 6:27:27 PM	Cisco Internetwork Operating System Software IOS (tm) C3500XL Software (C3500XL-C3H2S-M), Version 12.0 (5.1)XP, MAINTENANCE INTERIM SOFTWARE Copyright (c) 1986-1999 by cisco Systems, Inc. Compiled Fri 10-Dec-99 11:16 by cchang

The Web Client Topology screen provides a listing and description of all discovered switches. From here, you can perform OmniVista Topology functions. You can also click on the End Station Search or End Station Browse tabs to access Locator functions; or click on the Traps tab to access Notifications Functions.

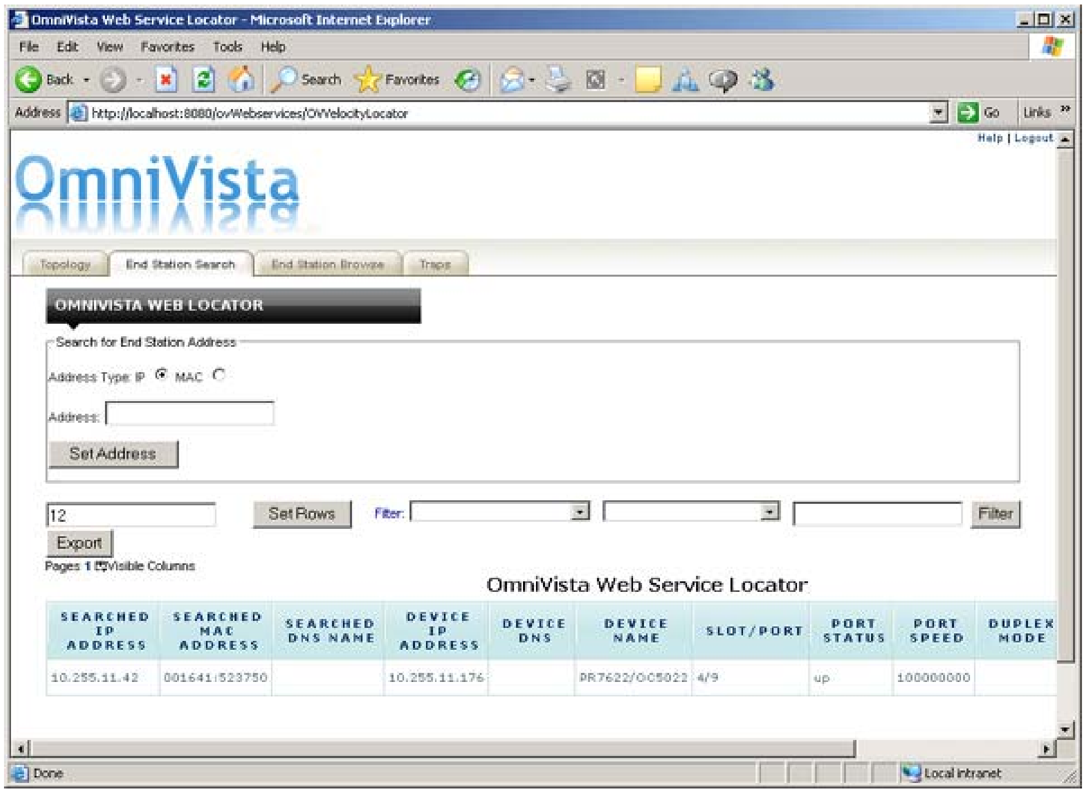
Locator

The OmniVista **Locator** application within the OmniVista Web Client is a search tool that enables you to search for end stations connected to switches in the network. The End Station Search tab enables you to locate a switch and that is directly connected to a user-specified end station. The End Station Browse tab enables you search in the "opposite direction". Instead of entering an end station's address to locate the switch and slot/port to which the end station is connected, you can search for and list ALL end stations connected to user-specified switch ports. The end stations are located by searching the historical database.

The search results display a list of devices in a table that provides basic information for each device. You can customize the table display, sort the information in the table by column, and create filters to view specific information. You can also export the information to a .CSV file.

End Station Search Tab

The **End Station Search** tab enables you to locate a switch and that is directly connected to a user-specified end station. You can enter the end station's IP address or MAC address to locate the switch and slot/port to which the end station is connected.



Search for End Station Address

Address Type: IP ● MAC ○

Address:

12

Pages 1 Visible Columns

OmniVista Web Service Locator

SEARCHED IP ADDRESS	SEARCHED MAC ADDRESS	SEARCHED DNS NAME	DEVICE IP ADDRESS	DEVICE DNS	DEVICE NAME	SLOT/PORT	PORT STATUS	PORT SPEED	DUPLEX MODE
10.255.11.42	001641:523750		10.255.11.176		PR7622/OC5022	4/9	up	100000000	

Although you can enter an end station's IP address or MAC address to locate the switch and slot/port that is directly connected to the end station, Locator actually searches for the end station's MAC address. If you enter an IP address, the first thing Locator does is find the corresponding MAC address. This MAC address is displayed in the **OmniVista Web Service Locator** table with a time stamp indicating when the information was obtained (last time the device was polled).

Searching for a Specific End Station

To search for a specific switch:

1. In the **Address Type** field, select IP to search by IP address or MAC to search by MAC address.

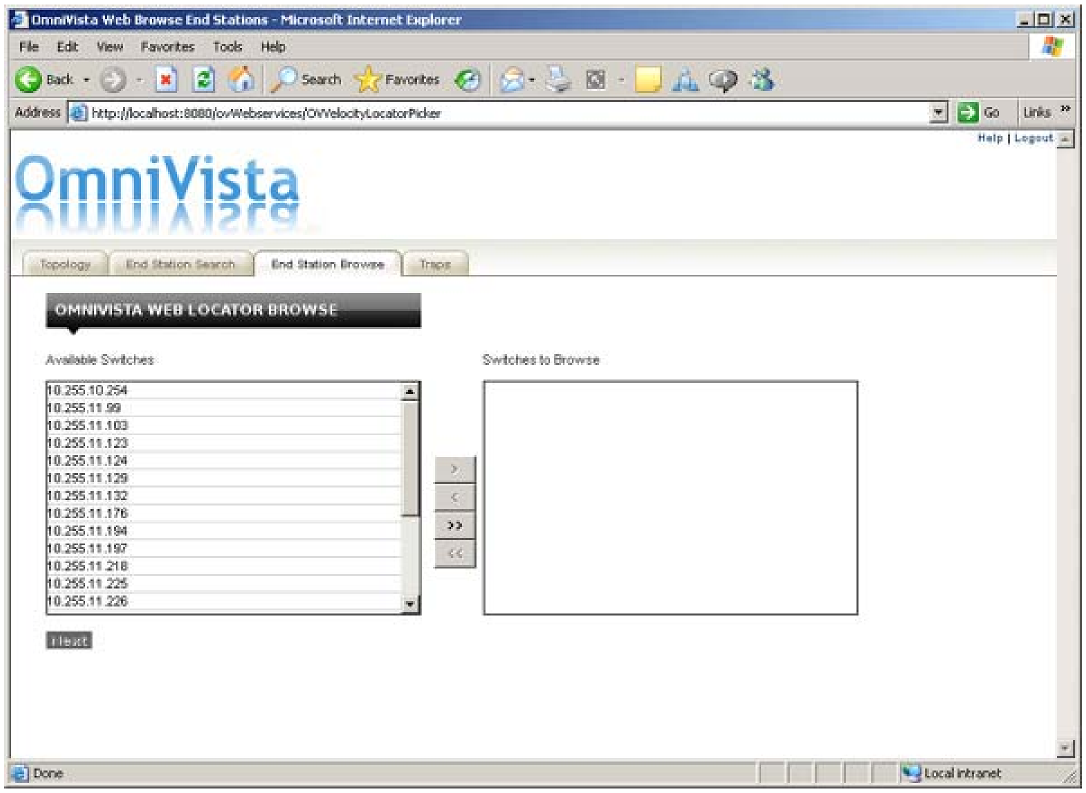
2. In the **Address** field, enter the IP or MAC address. If searching by MAC address, you must enter the address in 000000:000000 format.

3. Click the **Set Address** button.

The results will appear in the **OmniVista Web Service Locator** table. By default, the table is initially sorted in ascending order by IP address. You can sort the data by column heading (ascending/descending) or apply filters to the table to display specific information. You can also export the information to a .CSV file.

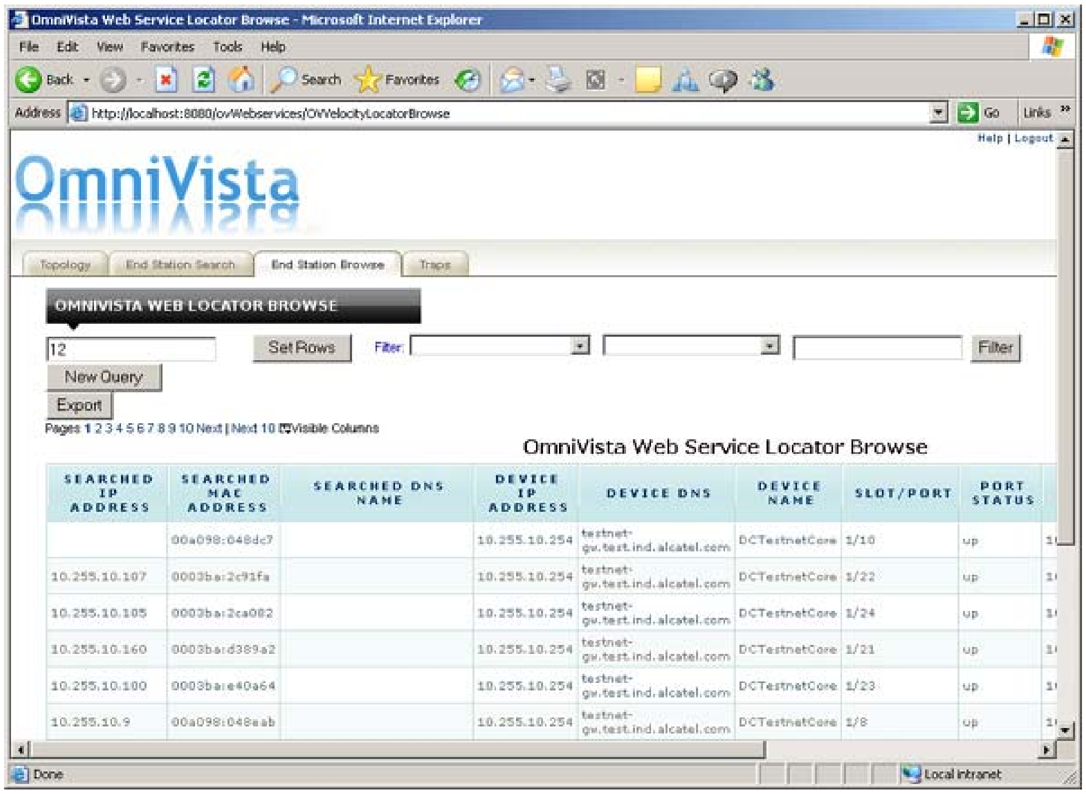
End Station Browse Tab

The **End Station Browse** tab enables you search in the "opposite direction" of the End Station Search tab. Instead of entering an end station's address to locate the switch and slot/port to which the end station is connected, you can search for and list ALL end stations connected to user-specified switch ports.


OmniVista Web Browse End Stations - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/ovWebservices/OVVelocityLocatorPicker
Go Links
Help | Logout
OmniVista
Topology
End Station Search
End Station Browse
Traps
OMNIVISTA WEB LOCATOR BROWSE
Available Switches
Switches to Browse
>
<
>>
<<
Next
Done
Local intranet

10.255.10.254
10.255.11.99
10.255.11.103
10.255.11.123
10.255.11.124
10.255.11.129
10.255.11.132
10.255.11.176
10.255.11.194
10.255.11.197
10.255.11.218
10.255.11.225
10.255.11.226

To browse for end stations select the switch(es) in which you are interested, then click the **Next** button. To select multiple switches, use the **Shift** or **Ctrl** keys. The results will appear in the **OmniVista Web Service Locator Browse** table. By default, the table is initially sorted in ascending order by IP address. You can sort the data by column heading (ascending/descending) or apply filters to the table to display specific information. You can also export the information to a .CSV file.


OmniVista Web Service Locator Browse - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/ovWebservices/OVVelocityLocatorBrowse
Go Links
Help | Logout
OmniVista
Topology
End Station Search
End Station Browse
Traps
OMNIVISTA WEB LOCATOR BROWSE
12
Set Rows
Filter:
Filter
New Query
Export
Pages 1 2 3 4 5 6 7 8 9 10 Next | Next 10 Visible Columns
Done
Local intranet

OmniVista Web Service Locator Browse

SEARCHED IP ADDRESS	SEARCHED MAC ADDRESS	SEARCHED DNS NAME	DEVICE IP ADDRESS	DEVICE DNS	DEVICE NAME	SLOT/PORT	PORT STATUS
	00a098:048dc7		10.255.10.254	testnet-gw.test.ind.alcatel.com	DCTestnetCore	1/10	up
10.255.10.107	0003ba:2c91fa		10.255.10.254	testnet-gw.test.ind.alcatel.com	DCTestnetCore	1/22	up
10.255.10.105	0003ba:2ca082		10.255.10.254	testnet-gw.test.ind.alcatel.com	DCTestnetCore	1/24	up
10.255.10.160	0003ba:d389a2		10.255.10.254	testnet-gw.test.ind.alcatel.com	DCTestnetCore	1/21	up
10.255.10.100	0003ba:e40a64		10.255.10.254	testnet-gw.test.ind.alcatel.com	DCTestnetCore	1/23	up
10.255.10.9	00a098:048eab		10.255.10.254	testnet-gw.test.ind.alcatel.com	DCTestnetCore	1/8	up

Table Display

You can set the number of rows that you want displayed on each page by entering the number of rows you want to display (e.g., 12), then clicking the **Set Rows** button. If a table spans several pages, click **Next** or a specific page number at the top left corner of the table to page through the table.

> **Note:** A maximum row setting of 100 is advised. Larger row settings can increase the time it takes to populate the table.

You can also configure the columns you want to display. Click on **Visible Columns** and select all of the columns you want to display. De-select any columns you want to hide. The new settings will remain in effect until you log out of the session. By default, all columns are displayed.

Column Definitions

Searched IP Address: The IP address of the end station connected to the selected device.

Searched MAC Address: The MAC address of the end station connected to the selected device.

Searched DNS Name: The DNS name of the end station connected to the selected device.

Device IP Address: The IP address of the device connected to the end station.

Device DNS: The DNS of the device connected to the end station.

Device Name: The name of the device connected to the end station.

Slot/Port: The slot/port of the device connected to the end station.

Port Speed: The port speed of the device connected to the end station.

Port Status: The port status of the device connected to the end station.

Duplex Mode: The duplex mode (half duplex, full duplex, or auto duplex) of the selected device connected to the end station.

VLAN ID: The VLAN ID associated with the device connected to the end station.

Timestamp: The time the information was gathered.

Sorting Information

You can sort the Locator Browse Table in ascending or descending order by clicking on a column heading.

Filtering Information

You can filter the information in the table to display specific switch information.

1. Select a column heading from the first **Filter** drop-down menu (e.g., Device IP Address, Device DNS).

2. Select an operator from the second **Filter** drop-down menu (e.g., Equals, Starts With).

3. Enter a value in the last **Filter** field (e.g., "9600").

For example, to view a list of devices in the Test Network, you would select "Device DNS", "Contains", and enter "testnet" in the last field).

4. When you have entered all of the filter criteria, click the **Filter** button. The results will be displayed in the Locator Table. To refine the filter, enter a new set of filtering criteria and click the **Refine Filter** button to display a list of devices matching both sets of filters.

> **Note:** You can display the results of previous filters by clicking the browser's **Back** button. The OmniVista Web Client will display up to eight (8) previous results in a single login session.

To start over click the **New Query** button.

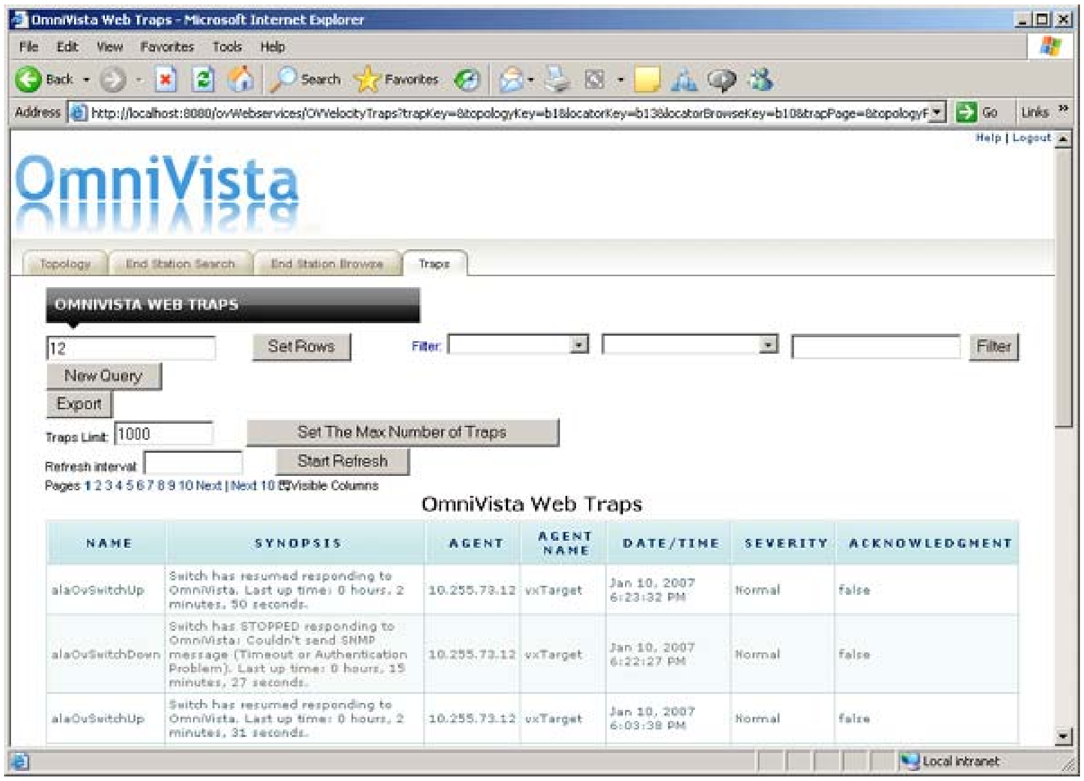
Exporting Information

To export the table to a .CSV file, use the browse function to locate the devices and populate the Locator Browse Table, filter the information (if applicable), then click the **Export** button. Click **Yes** at the Security prompt, then select the directory in which you want to save the file.

Notifications

The **Notifications** application within the OmniVista Web Client is used to monitor switch activity through the **OmniVista Web Traps** Table. The table, available under the **Traps** tab, displays information on all alarms and traps received by the OmniVista Server. You can customize the table display, sort the information in the table by column, and create filters to view specific information. You can also export the information to a .CSV file.

> **Note:** If the table is not displaying any notifications, it may be that none of your discovered switches have been configured to send traps to the OmniVista server.


OmniVista Web Traps - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/ovWebservices/OVVelocityTraps?trapKey=&topologyKey=b1&locatorKey=b13&locatorBrowseKey=b10&trapPage=&topologyF
Go Links
Help | Logout
OmniVista
Topology
End Station Search
End Station Browse
Traps
OMNIVISTA WEB TRAPS
Set Rows
Filter:
Filter
New Query
Export
Set The Max Number of Traps
Start Refresh
Pages: 1 2 3 4 5 6 7 8 9 10 Next | Next 10 Visible Columns
Local intranet

Traps Limit: 1000

Refresh interval:

OmniVista Web Traps

NAME	SYNOPSIS	AGENT	AGENT NAME	DATE/TIME	SEVERITY	ACKNOWLEDGMENT
alaOvSwitchUp	Switch has resumed responding to OmniVista. Last up time: 0 hours, 2 minutes, 50 seconds.	10.255.73.12	vxTarget	Jan 10, 2007 6:23:32 PM	Normal	false
alaOvSwitchDown	Switch has STOPPED responding to OmniVista: Couldn't send SNMP message (Timeout or Authentication Problem). Last up time: 0 hours, 15 minutes, 27 seconds.	10.255.73.12	vxTarget	Jan 10, 2007 6:22:27 PM	Normal	false
alaOvSwitchUp	Switch has resumed responding to OmniVista. Last up time: 0 hours, 2 minutes, 31 seconds.	10.255.73.12	vxTarget	Jan 10, 2007 6:03:38 PM	Normal	false

Table Display

You can set the number of rows that you want displayed on each page by entering the number of rows you want to display (e.g., 12), then clicking the **Set Rows** button. If a table spans several pages, click **Next** or a specific page number at the top left corner of the table to page through the table.

You can also configure the columns you want to display. Click on **Visible Columns** and select all of the columns you want to display. De-select any columns you want to hide. The new settings will remain in effect until you log out of the session. By default, all columns are displayed.

By default, up to 1,000 traps are displayed. However, you can configure the display by entering a number in the **Traps Limit** field, and clicking the **Set The Max Number of Traps** button. When the configured maximum number is reached, the newest traps overwrite the oldest traps.

The refresh function is disabled by default. To enable it, enter a value (in seconds) in the **Refresh Interval** field, and click the **Start Refresh** button (the refresh function will be enabled and the **Start Refresh** button will change to **Stop Refresh**). To disable the refresh function, click the **Stop Refresh** button. The default refresh interval is 30 seconds (this is also the minimum refresh interval).

Column Definitions

Name: The name of the trap as defined in the MIB.

Synopsis: A brief description of the trap.

Agent: The IP address of the switch that generated the trap.

Agent Name: The name of the switch that generated the trap.

Date/Time: The date and time the trap was received by the OmniVista server, using the OmniVista server's system clock.

Severity: The severity level assigned to the trap in the Notifications Application's Trap Definitions Window:

- Normal
- Warning
- Minor
- Major
- Critical.

Acknowledged: Indicates whether or not the trap has been acknowledged. "true indicates an acknowledged trap. "false" indicates that the trap that has yet been acknowledged, or the acknowledgement has been renounced.

Sorting Information

You can sort the **OmniVista Web Traps** Table in ascending or descending order by clicking on a column heading.

Filtering Information

You can filter the information in the table to display specific switch information.

1. Select a column heading from the first **Filter** drop-down menu (e.g., Name, Severity).

2. Select an operator from the second **Filter** drop-down menu (e.g., Equals, Starts With).

3. Enter a value in the last **Filter** field (e.g., "OS9600").

For example, to view a list of all traps with a Severity Level of "Major", you would select "Severity" in the first field, "Equals" in the second, then enter "Major" in the last field).

4. When you have entered all of the filter criteria, click the **Filter** button. The results will be displayed in the OmniVista Web Traps Table. To further refine the filter, enter a new set of filtering criteria and click the **Refine Filter** button to display a list of devices matching both sets of filters.

> **Note:** You can display the results of previous filters by clicking the browser's **Back** button. The OmniVista Web Client will display up to eight (8) previous results in a single login session.

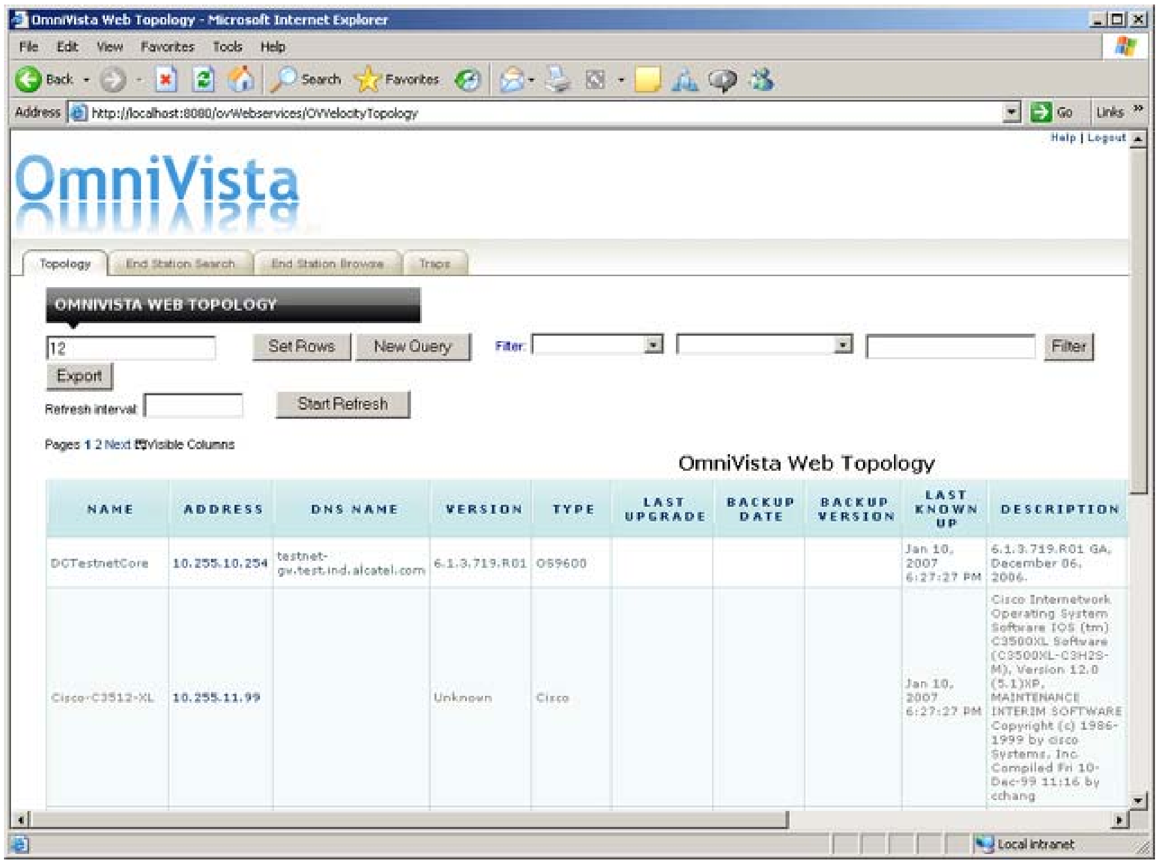
To start over and create a new filter, click the **New Query** button and repeat Steps 1 - 4.

Exporting Information

To export the table to a .CSV file, filter the information (if applicable), the click the **Export** button. Click **Yes** at the Security prompt, then select the directory in which you want to save the file.

Topology

The **Topology** application within the OmniVista Web Client is used to access the OmniVista "List of All Discovered Devices". The table, available under the **Topology** tab, provides basic information for all physical devices in the network, including all devices discovered by OmniVista, as well as any devices that were added manually. You can customize the table display, sort the information in the table by column, create filters to view specific information and export the information to a .CSV file. You can also access web-based management tools (e.g., WebView) for individual AOS switches listed in the Topology Table.



12

Pages 1 2 Next Visible Columns

OmniVista Web Topology

NAME	ADDRESS	DNS NAME	VERSION	TYPE	LAST UPGRADE	BACKUP DATE	BACKUP VERSION	LAST KNOWN UP	DESCRIPTION
DCTestnetCore	10.255.10.254	testnet-gw.test.ind.alcatel.com	6.1.3.719.R01	OS9600				Jan 10, 2007 6:27:27 PM	6.1.3.719.R01 GA, December 06, 2006.
Cisco-C3512-XL	10.255.11.99		Unknown	Cisco				Jan 10, 2007 6:27:27 PM	Cisco Internetwork Operating System Software IOS (tm) C3500XL Software (C3500XL-C3H2S-M), Version 12.0 (5.1)XP, MAINTENANCE INTERIM SOFTWARE Copyright (c) 1986-1999 by cisco Systems, Inc. Compiled Fri 10-Dec-99 11:16 by cchang

Table Display

You can set the number of rows that you want displayed on each page by entering the number of rows you want to display (e.g., 12), then clicking the **Set Rows** button. If a table spans several pages, click **Next** or a specific page number at the top left corner of the table to page through the table.

Note: A maximum row setting of 100 is advised. Larger row settings can increase the time it takes to populate the table.

You can also configure the columns you want to display. Click on **Visible Columns** and select all of the columns you want to display. De-select any columns you want to hide. The new settings will remain in effect until you log out of the session. By default, all columns are displayed.

The refresh function is disabled by default. To enable it, enter a value (in seconds) in the **Refresh Interval** field, and click the **Start Refresh** button (the refresh function will be enabled and the **Start Refresh** button will change to **Stop Refresh**). To disable the refresh function, click the **Stop Refresh** button. The default refresh interval is 30 seconds (this is also the minimum refresh interval).

Column Definitions

Address: The IP address of the device.

DNS Name: The name of the device, if applicable.

Version: The version number of the device firmware. Version numbers are not displayed for certain non-XOS devices.

Type: The type of the device chassis.

Last Upgrade Status: The status of the last firmware upgrade on the switch:

- "Successful" - Successful BMF and Image upgrade performed.
- "Successful (BMF)" - Successful BMF upgrade performed.
- "Successful (Image)" - Successful Image upgrade is performed.
- "Failed (BMF, Image)" - BMF and Image upgrade failed.
- "Failed (BMF)" - BMF upgrade failed.
- "Failed (Image)" - Image upgrade failed.

In all "Failed" cases, "Reload From Working" will be disabled on the switch until a successful upgrade is performed.

Backup Date: The date that the device's configuration and/or image files were last backed-up to the OmniVista server.

Backup Version: The firmware version of the configuration and/or image files that were last backed-up to the OmniVista server.

Last Known Up At: The date and time when the last poll was initiated on the device.

Description: A description of the device, usually the vendor name and model.

Status: This field displays the operational status of the device.

- **Up** - Device is up and responding to polls.
- **Down** - Device is down and not responding to polls.
- **Warning** - Device has sent at least one warning or critical trap and is thus in the warning state.

Traps: This field indicates the status of trap configuration for the device.

- **On** - Traps are enabled.
- **Off** - Traps are disabled.
- **Not Configurable** - Traps for this device are not configurable from OmniVista. (Note that traps may have been configured for such devices outside of OmniVista.)
- **Unknown** - OmniVista does not know the status of trap configuration on this switch. OmniVista will read the switch's trap configuration when traps are configured for the switch via the Configure Traps Wizard.

Seen By: This field lists the Security Groups that are allowed to view the device. (The Security Groups that are allowed to view a device can be defined when devices are auto-discovered, added manually, or edited.) The default Security Groups shipped with OmniVista are as follows:

- **Default Group** - This group has read-only access to switches in the list of All Discovered Devices that are configured to grant access to this group.
- **Writers Group** - This group has both read and write access to switches in the list of All Discovered Devices that are configured to grant access to this group. However, members of this group cannot run auto-discovery nor can they manually add, delete, or modify entries in the list of All Discovered Devices.
- **Network Administrators Group** - This group has full administrative access rights to all switches on the network. Members of this group can run autodiscovery and can manually add, delete, and modify entries in the list of All Discovered Devices. Members of this group also have full read and right access to entries in the Audit application and the Control Panel application. Members of this group can do everything EXCEPT make changes to Security Groups.
- **Administrators Group** - This group has all administrative access rights granted to the Network Administrators group AND full administrative rights to make changes to Security Groups.

Note that other Security Group names may display in this field if custom Security Groups were created. Refer to help for the Security application Users and Groups for further information on Security Groups.

Running From: For AOS devices, this field indicates whether the switch is running from the certified directory or from the working directory. This field is blank for all other devices. For AOS devices, the directory structure that stores the switch's image and configuration files in flash memory is divided into two parts:

- The Certified Directory contains files that have been certified by an authorized user as the default configuration files for the switch. When the switch reboots, it will automatically load its configuration files from the certified directory if the switch detects a difference between the certified directory and the working directory. (Note that you can specifically command a switch to reboot from either directory).
- The Working Directory contains files that may or may not have been altered from those in the certified directory. The working directory is a holding place for new files to be tested before committing the files to the certified directory. You can save configuration

changes to the working directory. You cannot save configuration changes directly to the certified directory.

Note that the files in the certified directory and in the working directory may be different from the running configuration of the switch, which is contained in RAM. The running configuration is the current operating parameters of the switch, which are originally loaded from the certified or working directory but may have been modified through CLI commands, WebView commands, or OmniVista. Modifications made to the running configuration must be saved to the working directory (or lost). The working directory can then be copied to the certified directory if and when desired.

Changes: For AOS devices, this field indicates the state of changes made to the switch's configuration. This field is blank for all other devices. This field can display the following values:

- Unsaved. Changes have been made to the running configuration of the switch that have not been saved to the working directory.
- Uncertified. Changes have been saved to the working directory, but the working directory hasn't been copied to the certified directory. The working directory and the certified directory are thus different.
- Blank. When this field is blank for an AOS device, the implication is that OmniVista knows of no unsaved configuration changes and assumes that the working and certified directories in flash memory are identical.

OmniVista is now capable of tracking AOS configuration changes made through CLI commands or WebView, and so will reflect configuration changes made outside of OmniVista through these two interfaces in the Changes field. Information in the Changes field will be accurate as long as OmniVista has polled the switch since the last change was made (through any interface).

Note that it is possible a switch could be in a state where it is both Unsaved and Uncertified. In this situation Unsaved displays in the Changes field. Whenever an AOS device is in the Unsaved or Uncertified state, a blue exclamation mark displays on its icon ( ).

Sorting Information

You can sort the Discovered Devices Table in ascending or descending order by clicking on a column heading.

Filtering Information

You can filter the information in the table to display specific switch information.

1. Select a column heading from the first **Filter** drop-down menu (e.g., Address, DNS Name).

2. Select an operator from the second **Filter** drop-down menu (e.g., Equals, Starts With).

3. Enter a value in the last **Filter** field (e.g., "9600").

For example, to view a list of 9600 devices, you would select "Type", "Equals", and enter "9600" in the last field).

4. When you have entered all of the filter criteria, click the **Filter** button. The results will be displayed in the OmniVista Web Topology Table. To further refine the filter, enter a new set of filtering criteria and click the **Refine Filter** button to display a list of devices matching both sets of filters.

> **Note:** You can display the results of previous filters by clicking the browser's **Back** button. The OmniVista Web Client will display up to eight (8) previous results in a single login session.

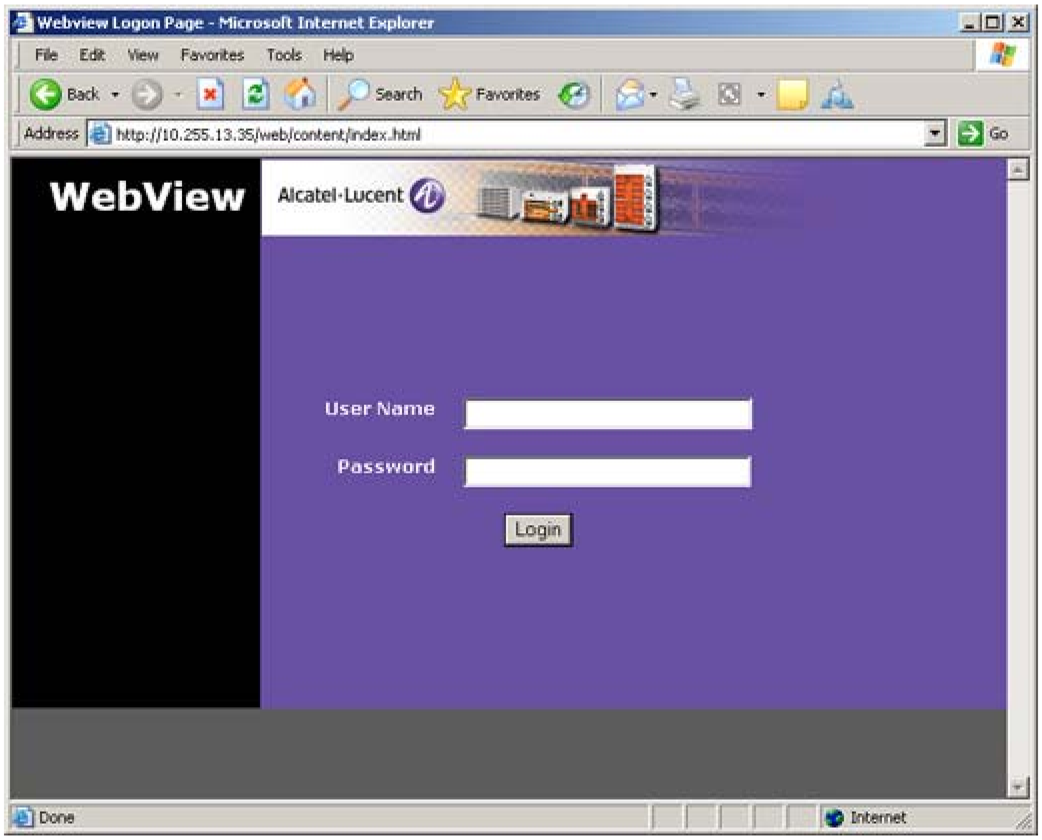
To start over and create a new filter, click the **New Query** button and repeat Steps 1 - 4.

Exporting Information

To export the table to a .CSV file, filter the information (if applicable), the click the **Export** button. Click **Yes** at the Security prompt, then select the directory in which you want to save the file.

Web-Based Management

You can also access web-based management tools (e.g., WebView) for individual AOS switches listed in the Topology Table by clicking on the switch's IP address in the Topology Table. If a switch has web-based management capabilities, the login screen for the switch will appear. A WebView Login Screen is shown below.


Webview Logon Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://10.255.13.35/web/content/index.html
Go
WebView
Alcatel·Lucent
User Name
Password
Login
Done
Internet

Web Services API

Software Interfaces

OmniVista provides a Web Services API that customers can use to write their own web applications. The service is available both by HTTP and by secure HTTPS, and uses the port(s) that the customer specified at installation for the web server that runs the OmniVista web client. At installation, OmniVista also gives the user the option of whether or not to allow HTTP (as opposed to HTTPS) connections to the web service; however, this setting can be modified after installation.

The web service delegates much of the work to a private OmniVista Server back end, over on non-public API, which listens on a port configured by the customer at installation.

The Web Service Design Language (WSDL) description of OmniVista's API is available to the user at "/axis/services/OVWeb1?wsdl" on the included Tomcat web server, e.g. "http://localhost:8080/axis/services/OVWeb1?wsdl ".

The Web Service API is described in a more human-readable form than WSDL in the following sections, using a Java-like syntax. Despite the use of the Java-like syntax in what follows, the actual web service is platform-independent, callable from a variety of web-development languages (e.g., Java, PHP, perl). The following sections describe the operations that can be performed, followed by the data types used and returned by those operations.

Common Behaviors

Web Service

- Any of the function calls listed below, which attempt to delegate work to the private back end server, return a Fault if they are unable to contact the private back end.
- The web service requires that the first call of a session be the login call, which defines a login session.
- The web service requires that all non-login calls to the web service identify the login session that they belong to by including a session cookie (see login).
- Callers to the web service are only allowed to read or modify the data belonging to the specified login session.
- The web service supports up to 6 active login sessions (logged-in web sessions which have not yet logged out nor expired due to inactivity).
- Login sessions expire after 30 minutes of inactivity, where activity is defined as making any function call within that login session.
- The web service displays at least as many results (e.g., number of Locator Net Forward results) as are supported by the traditional OmniVista "heavy" client.
- The getXxx() calls silently limit the number of returned rows to 10,000. Note that this does not limit the number of rows returned by the queryXxx() calls; it is simply a "page size" that limits the number of results returned at once.

ResultSets

Many OmniVista Web Service functions return some flavor of ResultSet. Each ResultSet includes a uniqueId, which is passed in subsequent function calls to operate on that ResultSet.

The web service supports the creation and simultaneous existence of a minimum of 8 ResultSets of each type (e.g., 8 TrapResultSets, 8 SwitchResultSets, etc) within each login session. When the number of ResultSets of a given type exceeds the, OmniVista may automatically dispose of the least-recently used ResultSet of that type, where a "use" is defined as "passing the uniqueId of a ResultSet to a function that supports that ResultSet ID type as an input."

Any function that accepts a ResultSet as an input, other than the disposeXxxResults() functions, returns a Fault if no such ResultSet of the specified type exists.

Function: login

```
void login(
    Base64 userName,
    Base64 password
    ) throws Fault
```

When this function is called, OmniVista shall attempt to create a login session for the given OmniVista user, using the given OmniVista password.

If the userName/password combination is incorrect, the function shall return a Fault.

If the maximum number of active login sessions has been reached, the function shall return a Fault.

If the login succeeds, the response headers shall include a set-cookie header identifying the login session ID, similar to the following example:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=65761E9ABEC4B23F0E66BA3AC3ED14D9; Path=/axis
Content-Type: text/xml;charset=iso-8859-1
Transfer-Encoding: chunked
Date: Tue, 26 Sep 2006 18:06:56 GMT
```

All subsequent calls that intend to execute within the newly-created login session must include this cookie in the request headers, similar to the following example:

```
POST /axis/services/OVWeb1 HTTP/1.1
Host: localhost:8080
User-Agent: NuSOAP/0.7.2 (1.94)
Connection: Keep-Alive
Content-Type: text/xml; charset=ISO-8859-1
SOAPAction: ""
Content-Length: 554
Cookie: JSESSIONID=65761E9ABEC4B23F0E66BA3AC3ED14D9;
```

The above cookie handling is done automatically, or nearly automatically, by many web service libraries, such as Java's AXIS, or PHP's NuSOAP.

OmniVista shall consider any call whose Cookie request header correctly identifies an existing login session to be part of that login session.

Function: logout

```
   void logout() throws Fault
```

When this function is called, OmniVista shall dispose of the current login session, defined as the login session identified by the cookie request header (for a description of cookie handling, see login() above).

On disposal of the login session, OmniVista shall also dispose of any cached query results that belong to that login session.

Function: querySwitches

```
   SwitchResultSet querySwitches(
       FilterObj []    filters,
       SortObj []      sorters,
       long            maxResults
       ) throws Fault
```

When this function is called, OmniVista shall return a filtered and sorted SwitchResultSet of up to maxResults switches.

If more than maxResults switches are available, OmniVista shall return the first maxResults switches which pass the given filters, if any.

If sorters is null, or the array of sorters is empty, the returned results shall be in ascending order of switch IP address.

The returned SwitchResultSet can be passed to getSwitchData() or to any of the other functions below which accept a SwitchResultSet.uniqueId – for example, to retrieve some of the data from the result set, or to resort the results, or to further filter the results.

Function: getSwitchData

```
   SwitchData [] getSwitchData(
       String          switchResultSet.uniqueId,
       long            offset,
       long            count
       ) throws Fault
```

When this function is called, OmniVista shall return an array of up to count SwitchData objects from the SwitchResultSet identified by switchResultSet.uniqueId, starting at the given zero-based row offset.

If no data is available in the SwitchResultSet starting at offset, then OmniVista shall return an empty array of SwitchData.

Function: sortSwitchResults

```
   SwitchResultSet sortSwitchResults(
       String          switchResultSet.uniqueId,
       SortObj []      sorters
       )throws Fault
```

When this function is called, OmniVista shall return a new SwitchResultSet with the identical contents as the input switchResultSet, but sorted in the order specified by sorters.

If sorters is null, or the array of sorters is empty, the returned results shall be in the same order as the input switchResultSet.

The input switchResultSet shall remain unchanged.

Function: refineSwitchResults

```
SwitchResultSet refineSwitchResults(
    String          switchResultSet.uniqueId,
    FilterObj []    moreFilters
    ) throws Fault
```

When this function is called, OmniVista shall return a SwitchResultSet with the identical contents as the input switchResultSet, but further filtered by the filters specified by moreFilters.

The input switchResultSet shall remain unchanged.

Function: disposeSwitchResults

```
void disposeSwitchResults(
    String          switchResultSet.uniqueId
    ) throws Fault
```

When this function is called, OmniVista shall dispose of the SwitchResultSet specified by the given uniqueId.

The function shall return normally even if no such SwitchResultSet exists (since that SwitchResultSet might have been automatically expired).

Function: queryTraps

```
TrapResultSet queryTraps(
    FilterObj []    filters,
    SortObj []      sorters,
    long            maxResults
    ) throws Fault
```

When this function is called, OmniVista shall return a filtered and sorted TrapResultSet of up to maxResults traps.

The traps returned shall be from OmniVista’s “post-absorption” stream (which is the same stream that is displayed on the traditional OmniVista “heavy” client).

If more than maxResults traps are available, OmniVista shall return the maxResults most recent traps which pass the given filters, if any.

If sorters is null, or the array of sorters is empty, the returned results shall be in descending order of the trap date/time (newest-first).

The returned TrapResultSet can be passed to getTrapData() or to any of the other functions below which accept a TrapResultSet.uniqueId – for example, to retrieve some of the data from the result set, or to resort the results, or to further filter the results.

Function: getTrapData

```
TrapData [] getTrapData(
    String          trapResultSet.uniqueId,
    long            offset,
    long            count
    ) throws Fault
```

When this function is called, OmniVista shall return an array of up to count TrapData objects from the TrapResultSet identified by trapResultSet.uniqueId, starting at the given zero-based row offset.

If no data is available in the TrapResultSet starting at offset, then OmniVista shall return an empty array of TrapData.

Function: sortTrapResults

```
TrapResultSet sortTrapResults(
    String          trapResultSet.uniqueId,
    SortObj []      sorters
    )throws Fault
```

When this function is called, OmniVista shall return a new TrapResultSet with the identical contents as the input trapResultSet, but sorted in the order specified by sorters.

If sorters is null, or the array of sorters is empty, the returned results shall be in the same order as the input trapResultSet.

The input trapResultSet shall remain unchanged.

Function: refineTrapResults

```
TrapResultSet refineTrapResults(
    String          trapResultSet.uniqueId,
    FilterObj []    moreFilters
    ) throws Fault
```

When this function is called, OmniVista shall return a TrapResultSet with the identical contents as the input trapResultSet, but further filtered by the filters specified by moreFilters.

The input trapResultSet shall remain unchanged.

Function: disposeTrapResults

```
void disposeTrapResults(
    String          trapResultSet.uniqueId
    ) throws Fault
```

When this function is called, OmniVista shall dispose of the TrapResultSet specified by the given uniqueId.

The function shall return normally even if no such TrapResultSet exists (since that TrapResultSet might have been automatically expired).

Function: queryLocatorBrowse

```
LocatorResultSet queryLocatorBrowse(
    String []       switchList,
    FilterObj []    netFwdFilters,
    SortObj []      netFwdSorters,
    long            maxNetFwdResults
    ) throws Fault
```

When this function is called, OmniVista shall return a filtered and sorted LocatorResultSet of up to maxNetFwdResults NetForwardData rows.

If switchList is non-null and non-empty, switchList shall be validated to be valid DNS or numeric IP addresses.

If switchList is non-null and non-empty, OmniVista shall limit the returned results to only those records that match one of the given switchIp addresses.

If more than maxNetFwdResults NetForwardData rows are available, OmniVista shall return the first maxNetFwdResults results which pass the given filters, if any.

If netFwdSorters is null, or the array of netFwdSorters is empty, the returned results shall be in ascending order of switch IP address.

The returned LocatorResultSet can be passed to getNetFwdData() or to any of the other functions below which accept a LocatorResultSet.uniqueId – for example, to retrieve some of the data from the result set, or to resort the results, or to further filter the results.

When executing this command, OmniVista shall use the currently-available Reverse DNS information, as opposed to the behavior on the traditional OmniVista heavy client, which begins with the currently-available Reverse DNS information and then trickles in updates. A future version may add support for trickle-back Reverse DNS updates).

Function: queryLocatorSearchByMac

```
LocatorResultSet queryLocatorSearchByMac(
    String          macAddress,
    FilterObj []    arpFilters,
    SortObj []      arpSorters,
    long            maxArpResults,
    FilterObj []    netFwdFilters,
    SortObj []      netFwdSorters,
    long            maxNetFwdResults
    ) throws Fault
```

When this function is called, OmniVista shall return search the OmniVista Locator database for records pertinent to the input macAddress, and return a filtered and sorted LocatorResultSet of up to maxArpResults ArpData rows and maxNetFwdResults NetForwardData rows.

OmniVista shall validate that macAddress is a well-formed MAC address, and throw a Fault if it is not.

If more than the requested number of rows are available, OmniVista shall return the first results which pass the given filters, if any.

If arpSorters is null, or the array of arpSorters is empty, the returned ArpData rows shall be in ascending order of IP address.

If netFwdSorters is null, or the array of netFwdSorters is empty, the returned results shall be in ascending order of switch IP address.

The returned LocatorResultSet can be passed to getArpData(), getNetFwdData() or to any of the other functions below which accept a LocatorResultSet.uniqueId – for example, to retrieve some of the data from the result set, or to resort the results, or to further filter the results.

When executing this command, OmniVista shall use the currently-available Reverse DNS information, as opposed to the behavior on the traditional OmniVista heavy client, which begins with the currently-available Reverse DNS information and then trickles in updates. A future version may add support for trickle-back Reverse DNS updates).

Function: queryLocatorSearchByIp

```
LocatorResultSet queryLocatorSearchByIp(
    String          ipAddress,
    FilterObj []    arpFilters,
    SortObj []      arpSorters,
    long            maxArpResults,
    FilterObj []    netFwdFilters,
    SortObj []      netFwdSorters,
    long            maxNetFwdResults
    ) throws Fault
```

When this function is called, OmniVista shall return search the OmniVista Locator database for records pertinent to the input ipAddress, and return a filtered and sorted LocatorResultSet of up to maxArpResults ArpData rows and maxNetFwdResults NetForwardData rows.

OmniVista shall validate that ipAddress is a well-formed DNS or numeric IP address, and throw a Fault if it is not.

If ipAddress is the address of a switch, rather than that of an end-station, the web service shall nonetheless return whatever data is available in the Locator database (as opposed to the traditional OmniVista “heavy” client, which pops up a message saying that this is the address of a switch).

If more than the requested number of rows are available, OmniVista shall return the first results which pass the given filters, if any.

If arpSorters is null, or the array of arpSorters is empty, the returned ArpData rows shall be in ascending order of IP address.

If netFwdSorters is null, or the array of netFwdSorters is empty, the returned results shall be in ascending order of switch IP address.

The returned LocatorResultSet can be passed to getArpData(), getNetFwdData() or to any of the other functions below which accept a LocatorResultSet.uniqueId – for example, to retrieve some of the data from the result set, or to resort the results, or to further filter the results.

When executing this command, OmniVista shall use the currently-available Reverse DNS information, as opposed to the behavior on the traditional OmniVista heavy client, which begins with the currently-available Reverse DNS information and then trickles in updates. A future version may add support for trickle-back Reverse DNS updates).

Function: getArpData

```
ArpData [] getArpData(
    String          locatorResultSet.uniqueId,
    long            offset,
    long            count
    ) throws Fault
```

When this function is called, OmniVista shall return an array of up to count ArpData objects from the LocatorResultSet identified by locatorResultSet.uniqueId, starting at the given zero-based row offseta.

If no data is available in the LocatorResultSet starting at offset, then OmniVista shall return an empty array of ArpData.

Function: getNetFwdData

```
NetForwardData [] getNetFwdData(
    String          locatorResultSet.uniqueId,
    long            offset,
    long            count
    ) throws Fault
```

When this function is called, OmniVista shall return an array of up to count NetForwardData objects from the LocatorResultSet identified by locatorResultSet.uniqueId, starting at the given zero-based row offset.

If no data is available in the LocatorResultSet starting at offset, then OmniVista shall return an empty array of NetForwardData.

Function: sortLocatorResults

```
LocatorResultSet sortLocatorResults(
    String          locatorResultSet.uniqueId,
    SortObj []      arpSorters,
    SortObj []      netFwdSorters,
    )throws Fault
```

When this function is called, OmniVista shall return a new LocatorResultSet with the identical contents as the input locatorResultSet, but sorted in the order specified by arpSorters. and netFwdSorters.

If arpSorters or netFwdSorters is null, or an array of sorters is empty, then the associated data in the returned results shall be in the same order as the input locatorResultSet.

The input locatorResultSet shall remain unchanged.

Function: refineLocatorResults

```
LocatorResultSet refineLocatorResults(
    String          locatorResultSet.uniqueId,
    FilterObj []    moreArpFilters
    FilterObj []    moreNetFwdFilters
    ) throws Fault
```

When this function is called, OmniVista shall return a LocatorResultSet with the identical contents as the input locatorResultSet, but further filtered by the filters specified by moreArpFilters and moreNetFwdFilters.

The input locatorResultSet shall remain unchanged.

Function: disposeLocatorResults

```
void disposeLocatorResults(
    String          locatorResultSet.uniqueId
    ) throws Fault
```

When this function is called, OmniVista shall dispose of the LocatorResultSet specified by the given uniqueId.

The function shall return normally even if no such LocatorResultSet exists (since that LocatorResultSet might have been automatically expired).

Data: ArpData

```
class ArpData
{
    // In the following, [Col] means:
    //    “Same as Col in Locator Initial Lookup Table.”
    String   ipAddress; // [IP Address]
    String   macAddress; // [MAC Address]
    String   dnsName; // [DNS Lookup]
    LongDate timeStamp; // [Time Stamp]
}
```

This data type represents a single row of ARP (Address Resolution Protocol) data collected from a switch.

It contains an associated IP/MAC address pair, the date and time when that address pair was read from a switch, and the reverse DNS name known by OmniVista for that IP address at that date/time.

Since OmniVista collects most reverse DNS data lazily, the dnsName is not guaranteed to be up-to-date as of date/time timeStamp.

When sorting or filtering ArpData, OmniVista shall accept a FilterObj.keyName which specifies any field of ArpData, such as ArpData.ipAddress, ArpData.macAddress, etc.

ArpData is returned by queryLocatorSearchByMac() and queryLocatorSearchByIp()

Data: Base64

Base-64 encoded data is a standard type used in web services to represent data which may contain characters that are meaningful to XML, such as “<”, “>”, etc.

The OmniVista web service shall* use Base64 when the data being transmitted may contain such characters.

*NOTE that we may change from Base64-encoding for some fields, such as login/password, to URL-encoding, if this is much more convenient for the user.

Data: FilterObj

```
class FilterObj
{
    // If true, we want objects that match, else we want those that
don’t.
    boolean     wantMatch;

      // An integer representing one of the following operators:
      // ==, !=, <, <=, >, >=, startsWith, endsWith, contains.
    int         operator;
    // Identifies the datum that we’re filtering on, e.g.
“TrapData.name”
    String      keyName;
    // The value that we’re comparing against.
    Base64      value;
}
```

OmniVista shall allow keyNames which are appropriate to the data being filtered, e.g. when filtering TrapData, use keyNames that begin with “TrapData”.

When filtering, if operator does not contain a meaningful value, a Fault shall be returned.

When filtering, if a keyName is inappropriate to the data being filtered, a Fault shall be returned.

An array of FilterObj objects shall be considered to specify an AND of those filters.

If a function such as queryTraps() is passed a null or empty array of FilterObj objects, the returned results shall be unfiltered.

When filtering on any dotted numeric IP address value (i.e. a value which the web service returns as a dotted numeric IP address), OmniVista shall expect a value that is a String containing a dotted numeric IP address, rather than a DNS name.

Data: LocatorResultSet

```
class LocatorResultSet extends ResultSet
{
    long            numArpResults;
    FilterObj []    arpFilters;
    SortObj []      arpSorters;
    long            numNetFwdResults;
    FilterObj []    netFwdFilters;
    SortObj []      netFwdSorters;
}
```

This data type represents the results of a Locator query.

The functions that expect a LocatorResultSet.uniqueId shall return a Fault if passed the uniqueId of another kind of ResultSet.

The numArpResults field contains the number of rows of ArpData that were returned by the query.

The arpFilters and arpSorters fields identify the arpFilters and arpSorters, if any, that were used to create the LocatorResultSet.

The xxxNetFwdXxx fields are similar, but apply to any NetForwardData returned by the query.

Data: LongDate

```
class LongDate
{
    long            msSinceEpochGMT;
}
```

This data type represents a date/time. LongDate are a documentation convention here; they will actually be coded in the WSDL identically to the long data type.

They always contain a date/time, expressed as the (positive or negative) number of milliseconds since midnight, January 1, 1970, Greenwich Mean Time, which is the same convention as java.util.Date, and is easily converted to other computer date/time standards.

Data: NetForwardData

```
class NetForwardData
{
    // In the following, [Col] means:
    //    “Same as Col in Locator Search Results Table.”
    String    endstationIpAddress;  // [Searched IP Address]
    String    endstationMacAddress; // [Searched MAC Address]
    String    endstationDnsName;    // [Searched DNS Name]
    String    switchIpAddress;      // [Device IP Address]
    String    switchDnsName;        // [Device DNS]
    Base64    switchSysName;        // [Device Name]
    int       slot;                 // “Slot” of [Slot/Port]
    int       port;                 // “Port” or [Slot/Port]
    int       ifIndex;              // SNMP ifIndex assoc. with
Slot/Port.
    String    portStatus;           // [Port Status]
    int       portSpeed;            // [Port Speed]
    String    portDuplexMode;       // [Duplex Mode]
    int       vlanId;               // [VLAN ID]
    LongDate  timeStamp;            // [Timestamp]
}
```

This data type represents a single row in the Locator NetForward table.

When sorting or filtering NetForwardData, OmniVista shall accept a FilterObj.keyName which specifies any field of NetForwardData, such as "NetForwardData.switchIpAddress", "NetForwardData.slot", etc.

NetForwardData is returned by queryLocatorBrowse(), queryLocatorSearchByMac(), and queryLocatorSearchByIp().

As a convenience, operations on NetForwardData accepts the comparison/sort key "NetForwardData.slotPort". When included in a sort request, "NetForwardData.slotPort".shall select "NetForwardData.slot" as the major key, and "NetForwardData.port" as the minor key. When included in a filter request, "NetForwardData.slotPort" shall return results consistent with an operation on the concatenation of slot, "/", port.

Data: ResultSet

```
class ResultSet
{
    String          uniqueId;
}
```

This data type represents the results of a query.

The uniqueId field identifies this ResultSet, and can be passed to functions to operate on the ResultSet..

Data: SnmpVar

```
class SnmpVar
 {
    String      name; // The name of an SNMP variable, e.g. in a
trap.
```

```
        Base64      value; // The value of that variable.
    }
```

This data type represents an SNMP Variable, such as those which are optionally included in TrapData [which see].

Data: SortObj

```
    class SortObj
    {
        // Identifies the datum that we’re sorting on, e.g.
    “TrapData.name”
        String      keyName;
        // If true, we want a low-to-high sort order, else high-to-low.
        boolean     ascending;
    }
```

OmniVista shall allow keyNames which are appropriate to the data being sorted, e.g. when sorting TrapData, use keyNames that begin with “TrapData”.

When sorting, if a keyName is inappropriate to the data being sorted, a Fault shall be returned.

An array of SortObj objects shall be considered to specify a sort order, major key first.

Data: Switch Data

```
    class SwitchData
    {
        String []   ipAddresses;       // All switch’s IP addresses.
        String      sysObjectId;       //                SNMP
    sysObjectID
    // In the following, [Col] means, “Same as column Col in Switches
    table.”
        Base64      sysName;           // [Name]         SNMP sysName
        String      ipAddress;         // [Address]      "main" IP
    address
        String      dnsName;           // [DNS Name]     Reverse DNS
    name
        Base64      type;              // [Type]         e.g. "Omni-5WX"
        Base64      version;           // [Version]      e.g.
    "5.1.6.32.R03"
        String      lastUpgradeStatus; // [Last Upgrade Status]
        LongDate    backupDate;        // [Backup Date]
        Base64      backupVersion;     // [Backup Version]
        LongDate    lastKnownUp;       // [Last Known Up At]
        Base64      description;       // [Description]  SNMP sysDescr
        String      upDownStatus;      // [Status] "Down", "Warning",
    "Up"
        String      trapsConfigured;   // [Traps] Configured to receive
    traps?
        String []   seenByGroups;      // [Seen By]      null if all can
    see.
        String      runningFrom;       // [Running From] AOS boot
    directory.
        String      changesSaved;      // [Changes]      AOS
    saved/certified.
        LongDate    discovered;        // [Discovered]   Date of first
    contact
    }
```

This data type represents a switch.

When sorting or filtering SwitchData, OmniVista shall accept a FilterObj.keyName which specifies any field of SwitchData, such as SwitchData.ipAddress.

When filtering SwitchData, if the name of an array field such as "SwitchData.ipAddresses" is passed in FilterObj.keyName, the filter shall be considered to mach if any of the elements of the array match.

When sorting SwitchData, if the name of an array field such as "SwitchData.seenByGroups" is passed in FilterObj.keyName, the SwitchData shall be sorted using each element of the array, starting from the first element.

Data: SwitchResultSet

```
class SwitchResultSet extends ResultSet
{
    long            numResults;
    FilterObj []    filters;
    SortObj []      sorters;
}
```

This data type represents the results of a switch query.

The functions that expect a SwitchResultSet.uniqueId shall return a Fault if passed the uniqueId of another kind of ResultSet.

The numResults field contains the number of rows of data that were returned by the query.

The filters and sorters fields identify the filters and sorters that were used to create the SwitchResultSet.

Data: TrapData

```
class TrapData
{
    long            instanceId;     // Uniquely ID's this trap
instance.
    int             snmpVersion;    // 1, 2, or 3
    LongDate        date;           // date/time that the trap
occurred.
    Base64          synopsis;
    boolean         acked;
    String          name;
    String          severity;
    String          agentIp;
    Base64          agentSysName;
    String          sourceIp;
    String          trapOID;
    SnmpVar []      snmpVars;
    String          enterprise;     // Valid for SNMP version 1 only
    String          enterpriseOID;  // Valid for SNMP version 1 only
    int             generic;        // Valid for SNMP version 1 only
```

```
        int             specific;       // Valid for SNMP version 1 only
    }
```

This data type represents a single instance of a trap notification received by the OmniVista Server.

When filtering TrapData, OmniVista shall accept a FilterObj.keyName which specifies a leaf field of TrapData, such as "TrapData.synopsis", "TrapData.severity", etc.

The Web Service shall accept of FilterObj.keyName in the form of "TrapData.snmpVars[snmpVariableName].value" to filter on the values of SNMP Variables in a trap. For example, to filter on SNMP Variable 'rndErrorDesc':, it will accept "TrapData.snmpVars[rndErrorDesc].value".

When sorting TrapData, OmniVista shall accept a SortObj.keyName which specifies a top-level leaf field of TrapData, such as "TrapData.synopsis", "TrapData.severity", etc. This means that OmniVista will not sort TrapData on "TrapData.snmpVars.name" nor "TrapData.snmpVars.value".

Data: TrapResultSet

```
    class TrapResultSet extends ResultSet
    {
        long            numResults;
        FilterObj []    filters;
        SortObj []      sorters;
    }
```

The functions that expect a TrapResultSet.uniqueId shall return a Fault if passed the uniqueId of another kind of ResultSet.

The numResults field contains the number of rows of data that were returned by the query.

The filters and sorters fields identify the filters and sorters that were used to create the ResultSet.