Testing Specific Protocols

This document provides a PDF version of the "How to Test Specific Protocols" topics from the Layer 4-7 Application (Avalanche Commander) Help. If you want a printed version of this information, you might find it more convenient to print this document instead of the individual Help topics. For each protocol it is assumed that you:

- Created a Device test if you want to emulate both clients and servers.
- Created an Application test if you want to emulate only clients.
- Completed entries on other tabs to define load, associations, and so on, that are required for all tests. For more information about other tabs, see Creating and Running an Advanced Test in the Help.

The following testing information is included in this document:

Testing Capture Replay

To complete Capture Replay information and run the test:

1. Learn about Avalanche Capture Replay testing.
2. Add a content file (if you want to supply commands this way).
3. Define Capture Replay information for the client.
4. Define Capture Replay information for the server. (Device test only)
5. Add an Actions list.
6. Calculate One-Way Latency.
7. Run the test and review results.

About Capture Replay Testing

Capture Replay testing helps you evaluate overall capacity handling, error handling, effectiveness of QoS mechanisms, and other capabilities. You can produce realistic traffic without a full implementation of the protocol by manually defining the commands to be sent, or recording a file of the protocol and replaying the communication. Your capture file must be libpcap-compatible. The following types of capture files are supported:

- PCAP files created using a packet capture program such as TCPDUMP or Wireshark
- CAP files created using Network Associates Sniffer 2.00x program

TIPS:

- If your capture file is in a rare or older format (such as Sun Snoop), you can open the file in a packet capture program, and use the **Save As...** option to save the file in a libpcap-compatible format.
- If your capture file for TCP does not contain data packets, you must manually specify the commands to be sent. (A TCP connection without data is not supported using the file method.)

NOTES:

- Capture Replay is a send and expect protocol. You must configure the flow of the session in both the client and the server consistently. That which is sent by the client is expected by the server, and vice-versa. The server expects an exact match, character for character. However, the client expects a match only with the first character.
- At the end of the session, specify close on both sides, with the value specified as **do** or **expect**. (UDP forces processing at the end of the session, so these commands can be omitted.)
- Capture Replay does not support IPv6 packets.

Adding a Content File

If you want to supply commands by using a PCAP or CAP file, you upload the file by using the Content Files tab. The PCAP file must be libpcap-compatible, such as that generated by Wireshark or TCPDUMP, or CAP files created using Network Associates Sniffer 2.00x program. You can use a file to supply client and server commands.

To add content files:

1. Click the **Content Files** tab.
2. Click the **Add** button. A directory dialog box appears. (For sample tests, sample streaming files are already loaded.)
3. Select the file, and then click the **Add** button. The file appears in the **Content Files** tab.

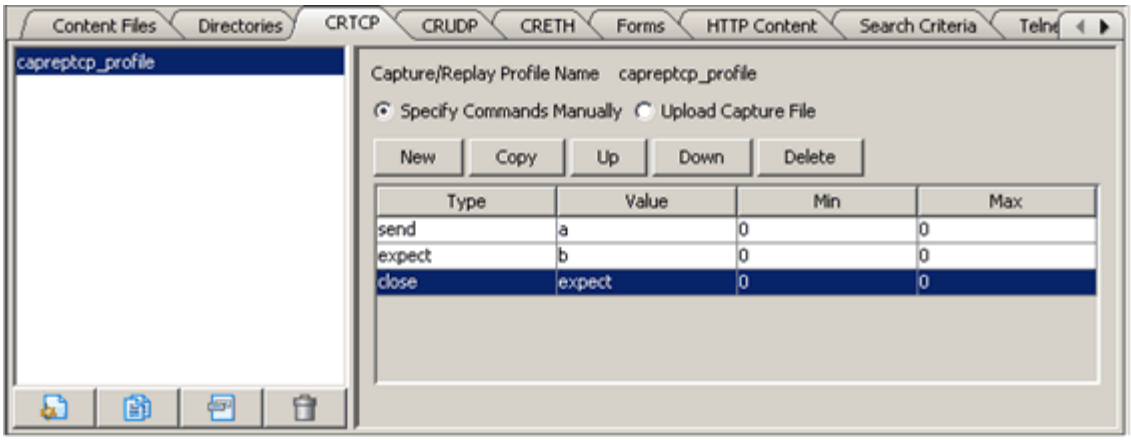
Defining Client Capture Replay Information

Define the commands that you want to use such as Send, Expect, and Wait, or select a file that defines the commands.

To configure the client:

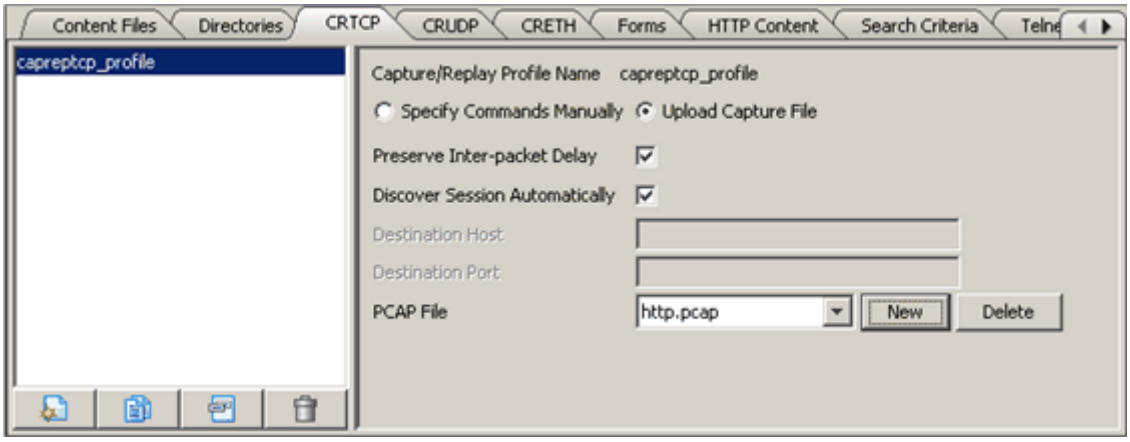
1. Click the **Client Actions** tab, and then the **CRTCP** or **CRUDP** tab (for TCP or UDP).
2. In the left pane of the **CRTCP** or **CRUDP** tab, select the Capture Replay profile that you want to use, or click the **New** button  and enter a name to create a new profile. You will reference the name of the Capture Replay profile when you create an Action list to test Capture Replay.
3. In the right pane of the **CRTCP** or **CRUDP** tab, select an editor type. Depending on the editor type that you select, different fields appear for you to define the commands.
4. If you select to enter commands manually, complete entries in the table that appears. For more information about commands, see Capture Replay Commands in the Help. If you select to upload a file, complete entries in the fields that appear. (You will identify the content file that you added previously.) For more information about fields, see Client Capture Replay Fields in the Help.

   The following shows a profile that uses commands.



NOTE: To use a forms database as the source of Send commands, select the forms_db command in the Type field. Enter the column number values (1, 2, 3, and so on, without a leading $ character) in the Value field. These values correspond to the column number in the forms database, starting from 1.

The following shows a profile that uses a file.



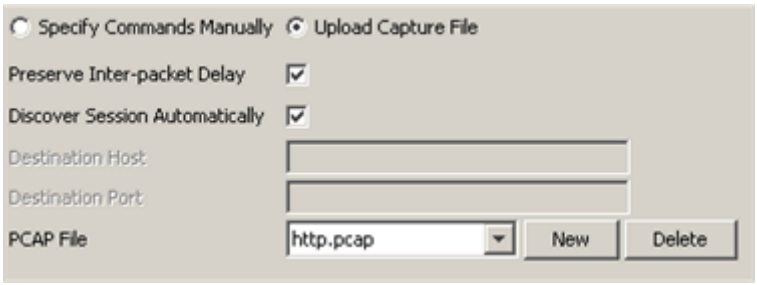
Defining Server Capture Replay Commands

Complete this section only if you are defining a Device test. You can enter server Capture Replay commands or reference a PCAP or CAP file that defines the commands. For more information about the fields, see Server Profiles Tab Capture Replay in the Help.

To configure a server profile for use with a PCAP or CAP file:

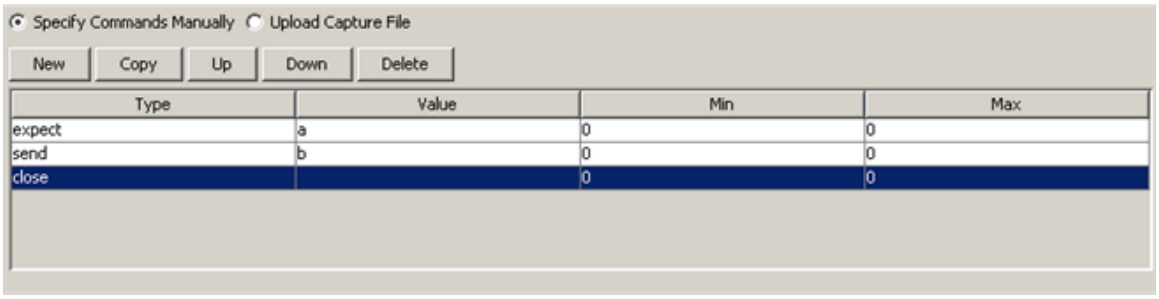
1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  and create a new profile.
3. Enter a description for the profile, and then select **CapRepTCP** or **CapRepUDP** as the server type based on the transport type (TCP or UDP, respectively) that you configured on the CRTCP or CRUDP tab of the Client Actions tab.
4. Enter the port where the server resides.

5. Select the **Upload Capture File** option.
6. Select options and enter information in the fields. (You will identify the content file that you added previously.)



To configure a server profile for use with commands:

1. Perform Steps 1-4 of the previous procedure.
2. Select the **Specify Commands Manually** option. The Capture Replay Command table appears.
3. Click the **New** button to add a new row to the table, and then enter information in the fields to define the commands.



Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

To create an Actions list:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter a description for the Actions list.
4. Enter information that specifies the IP address and port of the server, and the Capture Replay client profile name for TCP and UDP, respectively. See the following syntax and examples.

CAUTION: Do not include a trailing slash (/) after the IP address or an error will occur.

Syntax

```
CRTCP://ip address[:port] PROFILE=profile name [FORMS=forms
database name]

CRUDP://ip address[:port] PROFILE=profile name [FORMS=forms
database name]
```

- IP address:port number—IP address for the test. (Port is optional; if it is not included, then 2000 is assumed.)
- PROFILE=filename—Specifies the Capture Replay profile to use with this Action list. The profile name that you specify is the profile defined in the CRTCP or CRUDP tab of the Client Actions tab.
- FORMS=filename—If you use forms database values to provide Send command values in the profile, assign the name of the database here. Different Actions can use different forms databases, but only one forms database can be used on each separate Action.

5. Examples

```
CRTCP://192.168.42.11 PROFILE=capreptcp_profile
```

You can use a forms database as a source for commands, by specifying the forms_db command in the Type field on the client side in the Capture Replay Commands fields.

The following example uses a forms database:

```
CRUDP://192.168.42.11 PROFILE=caprepudp_profile
FORMS=dbfileabc
```

Calculating One-Way Latency

Before you configure a capture replay test to calculate one-way latency, you should first ensure that a basic session flow works properly.

To configure a basic session flow:

1. Specify the client commands as follows:

   send a

   expect b

   close do

2. Specify the server commands as follows:

   expect a

   send b

   close expect

3. Run the test, and ensure that the session flow is successful.

To calculate one-way latency:

1. Specify the client commands as follows:

send send_datetime

close do

2. Specify the server commands as follows:

   expect expect_datetime

   close expect

3. Run the test, and view the following Server Capture Replay real-time statistics:

   Minimum CapRepTCP (or CapRepUDP) One Way Latency

   Maximum CapRepTCP (or CapRepUDP) One Way Latency

   Average CapRepTCP (or CapRepUDP) One Way Latency

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run** Test  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing CIFSNG

NOTE: Avalanche provides a prototype CIFS (Common Internet File System) implementation denoted by "CIFS" in the Action list. This prototype is currently undocumented, and provides only a small subset of CIFS capabilities. The CIFSNG (Common Internet File System Next Generation) implementation provides more complete coverage of the CIFS protocol. If you are currently using the prototype CIFS, you cannot port to the new CIFSNG implementation directly. You will need to start from scratch.

IMPORTANT: When using an emulated server, real I/O is not actually performed in all cases, because of performance reasons. In general, operations that physically alter data on the emulated server are not performed (e.g., OPEN_NEW, COPY, DELETE, etc.). For example, when a client issues a write command, the emulated server will discard the data, but it will respond that the operation was successful. However, when necessary, the emulated server will perform real I/O, as in the case of a read command, so that it can return the actual file data to the client.

To complete CIFSNG-specific information and run the test:

1. Learn about Avalanche CIFSNG testing.
2. Define information for the client.
3. Add an Actions list.
4. Define the server profile. (Device test only)
5. Run the test and review results.

About CIFSNG Testing

The Common Internet File System (CIFS) is a file sharing protocol. Clients use this protocol to request file access services from servers over a network. CIFS is based on the Server Message Block (SMB) protocol, widely used by personal computers and workstations on a variety of operating systems. It is a key file sharing protocol due to its widespread distribution and enhancements for Internet authoring and file sharing.

Avalanche emulates multiple clients accessing files on a CIFS server. The server can be a real CIFS server, a cluster of real servers, and/or an emulated server. Avalanche gains access to shared folders through authentication, downloads and uploads real files through Action list commands, and collects relevant statistics.

As an alternative to real files, you can specify for Avalanche to generate file data using the on-the-fly feature. (See the on-the-fly example in the Actions list section.) These generated files are filled with the "s" character.

NOTE: See the Microsoft documents at http://search.microsoft.com/results.aspx?mkt=en-US&setlang=en-US&q=CIFS for an overview of CIFS. The Storage Networking Industry Association (SNIA) technical reference document, http://www.snia.org/tech_activities/CIFS/CIFS-TR-1p00_FINAL.pdf, contains further details on the CIFS protocol.

Defining Client Information

Define the client CIFSNG parameters that you want to use.

To configure the client:

1. Click the **Client** tab, and then the **Profiles** tab.
2. Select the client profile with which you want to work.
3. Click the **CIFSNG** tab, and complete entries to define the randomization and authentication parameters. For more information, see Client Profiles CIFSNG Fields in the Help.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

To add an Actions list for a CIFSNG test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter information that emulates clients accessing files on a CIFS server. Refer to the following syntax and examples.

Syntax

To specify the file structure for CIFSNG commands, you need to first upload the file directory using the Directories tab on the Client Actions tab. You then reference your named directory in the FSTREE action as follows:

```
FSTREE=directory_name
```

NOTES:

- The maximum size of this directory depends on the memory and storage resources of your hardware platform. For example, for Avalanche appliances, no more than 1 gigabyte is recommended.
- You should use the FSTREE keyword only once in your Actions list. If you use it multiple times, only the last one will apply to your test.

The following syntax establishes a TCP connection to a CIFS server:

```
cifsng://server_IP_address USER=user_name PASSWD=password
AUTH_TYPE=NTLM AUTHDOMAIN=SERVER
CONNECTION_TREE=\\server_IP_address\server_dir
```

- server_IP_address—The IP address of the server, such as 10.1.1.1
- USER—The user name used for authentication
- PASSWD—The password used for authentication
- AUTH_TYPE—Authentication type (NTLM, NTLMv2, NTLMv2_SESSION_SECURITY)
- AUTHDOMAIN—Authentication domain (any string)
- CONNECTION_TREE—The server IP address and directory or subdirectory tree on the server to which the client will connect.

Examples

```
FSTREE=cifs

cifsng://10.1.1.1 USER=cifs_server PASSWD=welcome AUTH_TYPE=NTLM
AUTHDOMAIN=SERVER CONNECTION_TREE=\\10.1.1.1\shared_dir
```

NOTE: Every OPEN command must have a corresponding CLOSE command.

- **Write Example (real server)**

  Open/create a new file (if it does not exist) called `simple_copy.txt`. Write 129000 bytes starting at byte 200 from `simple.txt` into `simple_copy.txt` starting at byte 100. Write `simple.txt` into `simple_copy.txt`. Close `simple_copy.txt`.

  ```
  CIFS_SUB_CMD OPEN_NEW FILENAME=simple_copy.txt

  CIFS_SUB_CMD WRITE FILENAME=simple_copy.txt OFFSET=100
  BYTE_SIZE=129000 INPUT_SOURCE=/cifs/simple.txt
  INPUT_OFFSET=200

  CIFS_SUB_CMD WRITE FILENAME=simple_copy.txt
  INPUT_SOURCE=/cifs/simple.txt

  CIFS_SUB_CMD CLOSE FILENAME=simple_copy.txt
  ```

- **Write Example (emulated server, files from directory)**

  NOTE: For this example, you select the following fields in the Server Profiles Tab CIFS_NG:

  - From Dir/File to enable the Serve Files from Directory field
  - Serve Files from Directory to upload the file directory named `server_dir`

  Open a file called `simple_copy.txt`. Write 129000 bytes starting at byte 200 from `simple.txt` into `simple_copy.txt` starting at byte 100. Write `simple.txt` into `simple_copy.txt`. Close `simple_copy.txt`.

  ```
  CIFS_SUB_CMD OPEN FILENAME=/server_dir/simple_copy.txt

  CIFS_SUB_CMD WRITE FILENAME=/server_dir/simple_copy.txt
  OFFSET=100 BYTE_SIZE=129000 INPUT_SOURCE=/cifs/simple.txt
  INPUT_OFFSET=200
  ```

```
CIFS_SUB_CMD WRITE FILENAME=/server_dir/simple_copy.txt
INPUT_SOURCE=/cifs/simple.txt

CIFS_SUB_CMD CLOSE FILENAME=/server_dir/simple_copy.txt
```

- **Write/On-the-Fly Example (real server)**

  Open/create a new file (if it does not exist) called `simple_longer_copy.txt`. Write `simple_longer.txt` into `simple_longer_copy.txt`. Write 50000 bytes starting at byte 50000 from `simple_longer.txt` into `simple_longer_copy.txt` starting at byte 10. Generate 128000 byes of on-the-fly data into `simple_longer_copy.txt`. Generate 100000 byes of on-the-fly data into `simple_longer_copy.txt`, randomize the offset, starting with 0, and increase it by 0 up to the file size. Close `simple_longer_copy.txt`.

  NOTE: When you generate file data using the on-the-fly feature, the maximum size of the combined generated files depends on the memory and storage resources of your hardware platform. For example, for Avalanche appliances, no more than 1 gigabyte is recommended.

```
CIFS_SUB_CMD OPEN_NEW FILENAME=simple_longer_copy.txt

CIFS_SUB_CMD WRITE FILENAME=simple_longer_copy.txt
INPUT_SOURCE=/cifs/simple_longer.txt

CIFS_SUB_CMD WRITE FILENAME=simple_longer_copy.txt OFFSET=10
BYTE_SIZE=50000 INPUT_SOURCE=/cifs/simple_longer.txt
INPUT_OFFSET=50000

CIFS_SUB_CMD WRITE FILENAME=simple_longer_copy.txt
BYTE_SIZE=128000 INPUT_SOURCE=ON_THE_FLY

CIFS_SUB_CMD WRITE FILENAME=simple_longer_copy.txt OFFSET=0
RANDOMIZE_OFFSET BYTE_SIZE=100000 INPUT_SOURCE=ON_THE_FLY

CIFS_SUB_CMD CLOSE FILENAME=simple_longer_copy.txt
```

- **Write/On-the-Fly Example (emulated server)**

  NOTES:

  - For this example, you select the On the fly field in the Server Profiles Tab CIFS_NG to generate file data using the *on-the-fly* feature.
  - When you generate file data using the on-the-fly feature, the maximum size of the combined generated files depends on the memory and storage resources of your hardware platform. For example, for Avalanche appliances, no more than 1 gigabyte is recommended.

Open a file called `simple_longer_copy.txt`. Write `simple_longer.txt` into `simple_longer_copy.txt`. Write 50000 bytes starting at byte 50000 from `simple_longer.txt` into `simple_longer_copy.txt` starting at byte 10. Generate 128000 byes of on-the-fly data into `simple_longer_copy.txt`. Generate 100000 byes of on-the-fly data into `simple_longer_copy.txt`, randomize the offset, starting with 0, and increase it by 0 up to the file size. Close `simple_longer_copy.txt`.

```
CIFS_SUB_CMD OPEN
FILENAME=/server_dir/simple_longer_copy.txt

CIFS_SUB_CMD WRITE
FILENAME=/server_dir/simple_longer_copy.txt
INPUT_SOURCE=/cifs/simple_longer.txt

CIFS_SUB_CMD WRITE
FILENAME=/server_dir/simple_longer_copy.txt OFFSET=10
BYTE_SIZE=50000 INPUT_SOURCE=/cifs/simple_longer.txt
INPUT_OFFSET=50000

CIFS_SUB_CMD WRITE
FILENAME=/server_dir/simple_longer_copy.txt BYTE_SIZE=128000
INPUT_SOURCE=ON_THE_FLY

CIFS_SUB_CMD WRITE
FILENAME=/server_dir/simple_longer_copy.txt OFFSET=0
RANDOMIZE_OFFSET BYTE_SIZE=100000 INPUT_SOURCE=ON_THE_FLY

CIFS_SUB_CMD CLOSE
FILENAME=/server_dir/simple_longer_copy.txt
```

- **Read Example (emulated client and server)**

  NOTE: This example assumes that the emulated server is configured with the `smalldir` named file hierarchy.

```
CIFSNG://172.30.3.51 USER=user_name PASSWD=user_password
AUTH_TYPE=NTLMv2 AUTHDOMAIN=your_domain
CONNECTION_TREE=Shared

CIFS_SUB_CMD OPEN FILENAME=/smalldir/smallfile.txt

CIFS_SUB_CMD READ FILENAME=/smalldir/smallfile.txt

CIFS_SUB_CMD CLOSE FILENAME=/smalldir/smallfile.txt
```

- **File Operations Example**

  Rename, move, copy, and delete files.

```
CIFS_SUB_CMD RENAME_FILE FILENAME=t.txt NEW_FILENAME=test.txt

CIFS_SUB_CMD MOVE_FILE FILENAME=test.txt
NEW_FILENAME=test.txt

CIFS_SUB_CMD COPY_FILE FILENAME=test.txt
NEW_FILENAME=test1234.txt

CIFS_SUB_CMD DELETE_FILE FILENAME=test.txt
```

- **Find File Example**

  Search for the first file that matches the file specification. Information level specifies to return both file and directory information.

```
CIFS_SUB_CMD FIND_FIRST FILENAME=\newfolder\f1\text.txt
INFO_LEVEL=FIND_FILE_BOTH_DIRECTORY_INFO
```

- **Query File Example**

  Get information about a named file or directory. Information levels specify to return basic, standard, extended attribute, and internal file information.

```
CIFS_SUB_CMD TRANS2_QUERY_PATH_INFO
FILENAME=\newfolder\f1\test.txt
INFO_LEVEL=QUERY_FILE_BASIC_INFO

CIFS_SUB_CMD TRANS2_QUERY_PATH_INFO
FILENAME=\newfolder\f1\test.txt
INFO_LEVEL=QUERY_FILE_STANDARD_INFO

CIFS_SUB_CMD TRANS2_QUERY_PATH_INFO
FILENAME=\newfolder\f1\test.txt INFO_LEVEL=QUERY_FILE_EA_INFO

CIFS_SUB_CMD TRANS2_QUERY_PATH_INFO
FILENAME=\newfolder\f1\test.txt
INFO_LEVEL=QUERY_FILE_INTERNAL_INFO
```

- **Query File System Example**

  Get file system information. Information levels specify to return volume, allocation, and size information.

```
CIFS_SUB_CMD QUERY_FS_INFO INFO_LEVEL=FS_VOLUME_INFO

CIFS_SUB_CMD QUERY_FS_INFO INFO_LEVEL=FS_INFO_ALLOCATION

CIFS_SUB_CMD QUERY_FS_INFO INFO_LEVEL=FS_SIZE_INFO
```

- **Query File/Read Example**

  Open file `test.txt`. Get basic, standard, extended attribute, and internal file information. Read 100 bytes, randomize the offset, starting with 0, and increase it by 0 up to the file size *minus* the requested number of bytes (100). Perform other read commands. Close `test.txt`.

```
CIFS_SUB_CMD OPEN FILENAME=\newfolder\f1\test.txt

CIFS_SUB_CMD TRANS2_QUERY_FILE_INFO
FILENAME=\newfolder\f1\test.txt
INFO_LEVEL=QUERY_FILE_BASIC_INFO

CIFS_SUB_CMD TRANS2_QUERY_FILE_INFO
FILENAME=\newfolder\f1\test.txt
INFO_LEVEL=QUERY_FILE_STANDARD_INFO

CIFS_SUB_CMD TRANS2_QUERY_FILE_INFO
FILENAME=\newfolder\f1\test.txt INFO_LEVEL=QUERY_FILE_EA_INFO

CIFS_SUB_CMD TRANS2_QUERY_FILE_INFO
FILENAME=\newfolder\f1\test.txt
INFO_LEVEL=QUERY_FILE_INTERNAL_INFO
```

```
CIFS_SUB_CMD READ FILENAME=\newfolder\f1\test.txt OFFSET=0
RANDOMIZE_OFFSET BYTE_SIZE=100

CIFS_SUB_CMD READ FILENAME=\newfolder\f1\test.txt

CIFS_SUB_CMD READ FILENAME=\newfolder\f1\test.txt OFFSET=2
BYTE_SIZE=10

CIFS_SUB_CMD CLOSE FILENAME=\newfolder\f1\test.txt
```

- **Directory Operations Example**

  Create a directory, check/verify that a path exists and is a directory, and delete a directory (must be empty).

```
CIFS_SUB_CMD CREATE_DIR
DIR_PATH=\newfolder\temporary_directory\

CIFS_SUB_CMD CHECK_DIR
DIR_PATH=\newfolder\temporary_directory\

CIFS_SUB_CMD DELETE_DIR
DIR_PATH=\newfolder\temporary_directory\
```

Using Loops and Timers

The CIFS loop action allows you to execute a set of CIFS actions multiple times. You can also introduce a delay between CIFS actions using the CIFS think time action.

NOTES:

- Nested loops are not permitted, and will result in an error.
- Every `OPEN` command must have a corresponding `CLOSE` command *within* the loop.

Loop Syntax

```
CIFS_SUB_CMD CIFS_START_LOOP = number_of_loops
```

list of `CIFS_SUB_CMD` actions

```
CIFS_SUB_CMD CIFS_STOP_LOOP
```

- number_of_loops—the number of times you want to execute the list of CIFS_SUB_CMD actions contained in the loop. When the count is reached, the action list continues after the CIFS_STOP_LOOP action.

Timer Syntax

```
CIFS_SUB_CMD CIFS_THINK_TIME = number_of_seconds
```

- number_of_seconds—the time delay in seconds.

Example

Execute a loop 10 times, which requests file system information and pauses 1 second after each request.

```
CIFS_SUB_CMD CIFS_START_LOOP = 10

CIFS_SUB_CMD QUERY_FS_INFO INFO_LEVEL=FS_VOLUME_INFO

CIFS_SUB_CMD CIFS_THINK_TIME = 1

CIFS_SUB_CMD CIFS_STOP_LOOP
```

Using a Forms Database

The following example uses a forms database with CIFSNG actions to obtain filename *randomization*. The example shows how to iterate through a list of files, such that each simulated user accesses a different filename.

NOTE: The variable assignments (`theFilename`, `theSrc`, `theDest`, and `thedirname`) must precede the CIFSNG actions.

```
ASSIGN VARIABLE <theFilename Form_1.$1>
ASSIGN VARIABLE <theSrc Form_2.$1>
ASSIGN VARIABLE <theDest Form_2.$2>
ASSIGN VARIABLE <thedirname Form_1.$2>

CIFSNG://10.1.1.1 USER=cifs_server PASSWD=welcome AUTH_TYPE=NTLM
AUTHDOMAIN=SERVER CONNECTION_TREE=\\10.1.1.1\shared_dir
```

NOTE: The following shows how to use `thedirname` variable to create a directory and check/verify that a path exists and is a directory.

```
CIFS_SUB_CMD CREATE_DIR DIR_PATH_VAR=<APPLY thedirname>
CIFS_SUB_CMD CHECK_DIR DIR_PATH_VAR=<APPLY thedirname>
```

NOTES:

- The following shows file and directory information being obtained for `theFilename` variable, as well as basic and extended attribute information. The file is opened, read, and closed.
- Write and rename commands are used with `theSrc` and `theDest` variables.

```
CIFS_SUB_CMD FIND_FIRST FILENAME_VAR=<APPLY theFilename>
INFO_LEVEL=FIND_FILE_BOTH_DIRECTORY_INFO

CIFS_SUB_CMD TRANS2_QUERY_PATH_INFO FILENAME_VAR=<APPLY
theFilename> INFO_LEVEL=QUERY_FILE_BASIC_INFO

CIFS_SUB_CMD OPEN FILENAME_VAR=<APPLY theFilename>
CIFS_SUB_CMD TRANS2_QUERY_FILE_INFO FILENAME_VAR=<APPLY
theFilename> INFO_LEVEL=QUERY_FILE_EA_INFO
CIFS_SUB_CMD CLOSE FILENAME_VAR=<APPLY theFilename>

CIFS_SUB_CMD OPEN FILENAME_VAR=<APPLY theFilename>
CIFS_SUB_CMD READ FILENAME_VAR=<APPLY theFilename>
CIFS_SUB_CMD READ FILENAME_VAR=<APPLY theFilename> OFFSET=4
BYTE_SIZE=1000
CIFS_SUB_CMD WRITE FILENAME_VAR=<APPLY theSrc>
INPUT_SOURCE_VAR=<APPLY theDest>
```

```
CIFS_SUB_CMD CLOSE FILENAME_VAR=<APPLY theFilename>
CIFS_SUB_CMD RENAME_FILE FILENAME_VAR=<APPLY theSrc>
NEW_FILENAME_VAR=<APPLY theDest>
```

NOTE: The following shows how to delete `theDest` file and `thedirname` directory.

```
CIFS_SUB_CMD DELETE_FILE FILENAME_VAR=<APPLY theDest>
CIFS_SUB_CMD DELETE_DIR DIR_PATH_VAR=<APPLY thedirname>
```

Defining the Server Profile

Complete this section only if you are defining a Device test. Use the **Server Profiles** tab to configure information for an emulated CIFS server, which can respond to CIFS requests generated by Avalanche.

To define a CIFS server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **CIFS_NG** as the server type.
4. Enter the port where the server resides. The default port is 445.
5. In the Data Source pane, select to generate file data by creating data on the fly, or by using real files from a named file hierarchy.
6. In the Authentication and Data Randomization pane, you can select an authentication profile to use and configure randomization parameters for read operations. For more information about CIFS_NG fields, see Server Profiles Tab CIFS_NG in the Help.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** button to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing DHCP

To complete DHCP-specific information and run the test:

1. Learn about Avalanche DHCP testing.
2. Add an Actions list.
3. Define the server profile. (Device test only)
4. Run the test and review results.

About DHCP Testing

DHCP is a communications protocol that lets you automate the assignment of IP addresses in a network, as well as other networking parameters, such as DNS, static routes, and so on. It allows you to assign IP addresses from a central point and automatically reassign a new IP address when a computer is moved within the network, for example.

You can emulate a client requesting IP addresses from a DHCP server using an Actions list, and can also send options 60 (Vendor Class Identifier) and 82 (Agent Information Option). You configure the value of these options in the Actions list or in the Client Subnets DHCP tab:

- Option 60: Identification of the vendor's equipment, for example, a set-top box manufacturer and software version.
- Option 82: User identification, for example, a unique ID that is assigned to each user and sent to the service provider, so that it can validate to which services the user has subscribed.

You can emulate a DHCP server to define a pool of IP addresses that will be offered to the clients requesting IP addresses. For more information about the role of the server and detailed Server Profile field definitions, see Server Profiles Tab DHCP in the Help.

You can emulate a DHCP relay, such that Avalanche routes the relayed DHCP message to the target DHCP server. You emulate a DHCP relay using the following:

- Source and destination UDP ports both set to 67
- DHCP message sent unicast to the server
- Other DHCP Action list options

IMPORTANT:

- With DHCP relay emulation, Avalanche participates in the usual DHCP transaction scenarios, with one significant difference: When Avalanche receives DHCP_ACK from the server, it in **no way** changes the DHCP user's IP address, default route, etc. The DHCP user's role is simply to act as a DHCP relay; however, it will **not** relay the DHCP message from the server to some other DHCP client in front of the Avalanche relay. In other words, Avalanche emulating a DHCP relay via the Actions list is simply meant to *simulate* a DHCP relay. It is not an actual DHCP relay. You can run a test with a DHCP relay in the Actions list, along with various application protocols, but it will not necessarily assign these application protocols a DHCP address.
- Avalanche acting as a DHCP relay should be on the same network subnet as the server. Avalanche will attempt to ARP the target server IP's MAC address in order to proceed with the DHCP transaction.

NOTE: Refer to the following RFCs for more information about DHCP:

- RFC 2131 - Dynamic Host Configuration Protocol
- RFC 2132 - DHCP Options and BOOTP Vendor Extensions
- RFC 3046 - DHCP Relay Agent Information Option

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

To add an Actions list for a DHCP test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter information that simulates DHCP. Refer to the following syntax and examples.

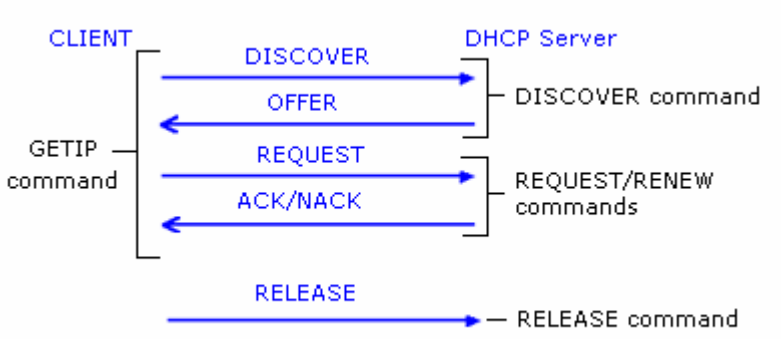
Syntax

```
dhcp:// <cmd> [reqip=<requested ipv4 addr> svrip=<server ipv4
addr>] [<dhcp parameters> reqopts=<dhcp options>]
```

- <cmd>: DISCOVER | REQUEST | RELEASE | GETIP | RENEW
- reqip=<requested ipv4 addr>: The IPv4 address that the client wants to use (required for REQUEST, but can be omitted for RELEASE or RENEW commands).

- svrip=<server ipv4 addr>: The IPv4 address of the DHCP server that the client wants to use (required for REQUEST, but can be omitted for RELEASE or RENEW commands).
- <dhcp parameters>: <dhcp parameters> <dhcp param> | <dhcp param>
- <dhcp param>: <dhcp-lval> '=' <action-variable> | <dhcp_unary_val>
- <dhcp-lval>: TIMEOUT | RETRIES | VENDOR | DHCPRLYS1 | DHCPRLYS2 | SVRIP | CMAC
- <dhcp_unary_val>: DEFOPTS | RELAY
- reqopts=<dhcp options>: <dhcp options> , <integer-value> | <integer-value>

The following figure illustrates a complete DHCP exchange:



Parameters and Options

Parameters and Options differ in that parameters can be set by a client. Some parameters, such as the Vendor ID (VENDOR) and DHCP Relay Sub-options 1 and 2 (DHCPRLYS1 and DHCPRLYS2), are actual DHCP options. The parameters in the Actions list syntax are consistent with those specified in the Client Subnets DHCP tab.

Values in an *Option List* simply define the *Parameter Request List* DHCP option, which is used by a client to query a server's capabilities (read-only).

Default Option List

The following table contains the *default* DHCP Option List. The client can create a *new* default DHCP Option List for a given DHCP server, to which a DHCP server can choose to respond or not.

NOTE: The default DHCP Option List is sent during a DISCOVER or GETIP command.

Option Code	Description
1	Subnet mask
3	Router (gateway IP address)
6	DNS servers
12	Hostname
15	Domain
28	Broadcast address
33	Static routes
42	NTP server
72	Web server
120	SIP server

DHCP Relay

You emulate a DHCP relay using the following syntax:

dhcp://<cmd> relay svrip=<server ip addr > cmac=<client MAC addr>

<cmd>: DISCOVER | REQUEST | RELEASE | GETIP | RENEW

<server ip addr>: <apply variable1> | string-value

<client MAC addr>: <apply variable2> | string-value

NOTES:

- The SVRIP keyword specifies the DHCP server to use.
- If you are using the CMAC parameter, you must specify the RELAY keyword. Otherwise, you will receive a parse error.

Examples (basic)

- dhcp://getip

  The client starts a complete DHCP exchange, and includes the default DHCP options listed in the table above.

- dhcp:// release

  The client releases its lease on the IP address. (The period over which an IP address is allocated to a client is referred to as a *lease*.)

Examples (advanced)

The following examples emulate portions of a DHCP exchange for use when you need more granularity than provided by the GETIP command alone:

- dhcp:// getip VENDOR="Spirent" DHCPRLYS1="3000" DHCPRLYS2="3333"

  The client starts a complete DHCP exchange, and sends a Vendor ID (option 60) of "Spirent" and an option 82 value pair of "3000","3333".

- dhcp:// getip reqopts=1,3,6,15,72

  The client starts a complete DHCP exchange and overrides the default DHCP Option List, querying a DHCP server for the following DHCP options: subnet mask, gateway, DNS, domain name, and Web server(s), respectively.

  NOTE: It is valid for a client request to specify a DHCP option that is already in the default list of options.

- dhcp:// getip defopts reqopts=7,13,122

  The client starts a complete DHCP exchange and requests the DHCP Option List, in addition to the options 7, 13 and 122, from a DHCP server. The server responds with the values it understands.

  NOTE: The following two commands are equivalent:

  - dhcp:// getip
  - dhcp:// getip defopts

- dhcp:// DISCOVER

  The client sends the initial DISCOVER command to the DHCP server, which is the beginning of the DHCP exchange. It also requests the default list of DHCP options from the server.

- dhcp:// request reqip="20.0.0.2" svrip="20.0.0.1"

  The client requests IP address 20.0.0.2 from DHCP server 20.0.0.1. The server simply allows or denies the request.

- dhcp:// renew

  The client requests to use the IP address and DHCP server IP address known from the last DISCOVER command. The server simply allows or denies the request.

NOTES:

- The following two commands are equivalent:
  1. dhcp:// request reqip="20.0.0.2" svrip="20.0.0.1"
  2. dhcp:// renew reqip="20.0.0.2" svrip="20.0.0.1"
- In the current release, the RENEW command sends a REQUEST message, and expects the DHCP server to send an ACK or NACK response. If you do not specify the requested IP address and server IP address, those IP addresses learned from a previous DISCOVER transaction will be used in the REQUEST message. If no such information exists, the RENEW command will fail.

- dhcp://getip relay svrip="1.2.3.4" cmac="00:11:22:33:44:55"

  Avalanche emulates a DHCP relay using server IP address 1.2.3.4 and client MAC address 00:11:22:33:44:55.

- dhcp://getip relay svrip=<apply mysvr> cmac=<apply mycmac>

  Avalanche emulates a DHCP relay using variables for the server IP address and client MAC address, defined as follows:

  assign variable <mysvr "1.2.3.4">

  assign variable <mycmac "11:22:33:44:55:66">

Using a Forms Database for DHCP Parameters

The following example shows how to define various DHCP parameters using variables from a forms database:

```
assign variable <dhcpto DhcpParams.$1> ($1 is the first column)

assign variable <dhcpretries DhcpParams.$2> ($2 is the second column)

assign variable <dhcpvid DhcpParams.$3> ($3 is the third column)

assign variable <rlys1 DhcpParams.$4> ($4 is the fourth column)

assign variable <rlys2 DhcpParams.$5> ($5 is the fifth column)

dhcp://getip timeout=<apply dhcpto> retries=<apply dhcpretries> vendor=<apply
dhcpvid> dhcprlys1=<apply rlys1> dhcprlys2=<apply rlys2>
```

NOTE: Alternatively, you can use the Client Subnets DHCP tab, Use This Forms DB field, instead of assigning the variables for DHCPRLYS1 and DHCPRLYS2.

The following shows a sample line from the DhcpParams forms database used in this example:

```
1000,3,SpirentAvalanche,VLANDhcp1,snet42
```

The following parameters are defined from the forms database columns:

- timeout - column 1=1000
- retries - column 2=3
- vendor - column 3=SpirentAvalanche

- dhcprlys1 - column 4=VLANDhcp1
- dhcprlys2 - column 5=snet42

Accessing DHCP Option Values within an Action List

When a DHCP exchange completes successfully, it stores certain option values that you can access using an Action list. The following values are accessible, provided you requested these DHCP options from the server:

- Primary DNS server - IPv4 address
- Secondary DNS server - IPv4 address
- Primary SIP server - IPv4 address or FQDN

You use the following variable names in an Action list to access the values above:

- SPIDNS1 - Primary DNS server
- SPIDNS2 - Secondary DNS server
- SPISIP1 - Primary SIP server

You can use the following variable names in an Action list to access other information:

- SPIGUID - A global, monotonically increasing counter that increases by one each time a SimUser is born. The initial value is 1. You can set the variable to another initial value using the ASSIGN VARIABLE statement. Subsequent ASSIGN statements will not have any effect on the variable.
- SPIUSRIP - The SimUser's current IP address.

Examples

- DNS:// <apply SPIDNS1> A sip.spirentcom.com

  Resolves the DNS name sip.spirentcom.com by sending the IP address specified in the variable SPIDNS1 to the DNS server.

- DNS://192.168.43.42 A <apply SPISIP1> <apply mysip>

  Resolves the DNS name stored in the variable SPISIP1 and stores the SIP server's IP address in the variable mysip.

Limitations

- The period over which an IP address is allocated to a client is referred to as a *lease*. Once a client obtains a lease from the DHCP server, it is the client's responsibility to relinquish the lease, as there is no lease expiration. You must explicitly invoke the RELEASE command when you want to relinquish the DHCP lease.
- The DHCP RENEW/REBIND timers are not supported in the current release.
- For a given test, you cannot enable DHCP in the Client Subnets DHCP tab, and specify a DHCP:// action in its Action list. This is an invalid configuration.

- You should disable Gratuitous ARP in the Ports tab, if you are using DHCP:// in the Action list.
- Some DHCP servers cannot release and renew as fast as the client can do a DHCP request. In these cases, you should insert a sleep or think time between groups of actions, as in the following example:

```
DHCP://GETIP RETRIES=5 TIMEOUT=5000

1 get http://192.168.1.1/index.html

DHCP://release

sleep 2000 (gives the DHCP server time to refresh its database; longer or
shorter times may apply)

DHCP://GETIP RETRIES=5 TIMEOUT=5000

1 get http://192.168.1.1/index.html

DHCP://release
```

Defining the Server Profile

Complete this section only if you are defining a Device test. Use the **Server Profiles** tab to configure information for a DHCP server.

To define a server profile:

7. Click the **Server** tab, and then the **Profiles** tab.
8. Click the **New** button  to create a new profile.
9. Enter a description for the profile, and then select **DHCP** as the server type.
10. Enter information in the DHCP pane. For more information about DHCP fields, see Server Profiles Tab DHCP in the Help.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing DNS

To complete DNS-specific information and run the test:

1. Learn about Avalanche DNS testing.
2. Define information for the client.
3. Add an Actions list.
4. Define the server profile. (Device test only)
5. Run the test and review results.

About DNS Testing

With the Layer 4-7 Application, you can generate DNS queries requesting responses from an emulated DNS Server. This testing allows you to assess the capacity and functionality of DNS-aware devices, such as firewalls and DNS server infrastructures.

DNS support allows:

- Enterprises and Service Providers to test a DNS infrastructure using an emulated client for load generation. The emulated client generates DNS queries and sends them to the DNS infrastructure.
- Network Equipment Manufacturers to test the DNS correctness and capacity of their DNS-aware devices. The emulated client generates DNS queries that pass through an intermediate device, and an emulated DNS server responds to those queries.
- Service Providers and Enterprises to evaluate network equipment during the buying process and before it is released to a production environment.

Defining Client Information

Define the client DNS parameters that you want to use.

To configure the client:

1. Click the **Client** tab, and then the **Profiles** tab.
2. Select the client profile with which you want to work.
3. In the **DNS** pane of the **DNS/MM4/SIP/Streaming** tab, complete entries to define the DNS parameters. For more information, see Client Profiles DNS Fields in the Help.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom. Unlike Avalanche HTTP syntax, you don't specify levels for DNS because they are not relevant for DNS testing.

TIP: You can use the Create New DNS Host File window to create DNS host files that map host names to the IP addresses of URLs in an Action list.

To add an Actions list for a DNS test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter information that simulates a DNS client sending different types of DNS name resolution requests to a DNS server. Refer to the following syntax and examples.

Syntax

```
DNS query_type server_IP_address name_to_resolve
[<RECURSIVE>] [<EXPECTS IP_address>]
[Secondary_server_ip_address]
```

NOTE: You can use the parameters <RECURSIVE> and <EXPECTS IP_address> with all DNS query types except PTR.

- query_type—The DNS query type, such as A (Address) or PTR (pointer). See Defining the Server Profile later in this topic, for a complete list of query types.
- server_IP_address—The DNS server IP address, such as 10.1.2.3.
- name_to_resolve—The web site name to be resolved, such as www.somewebsite.com.
- <RECURSIVE> (optional)—Enables the Recursive flag in the request message. It requests that the DNS server completely resolve the DNS name before responding, instead of supplying only the first step in name resolution. The DNS server can be configured to allow or deny recursive requests. Some DNS servers are configured to deny recursive queries to avoid burdening the DNS server.
- <EXPECTS IP_address> (optional)—Identifies the IP address expected for the DNS name specified, such as <EXPECTS 10.1.1.1>.
- Secondary_server_ip_address (optional)—A secondary DNS server IP address that is accessed if the web site name is not resolved by the first server.

Examples

- Address query:

```
DNS A 10.1.2.3 www.somewebsite.com

DNS A 10.1.2.3 www.somewebsite.com <RECURSIVE>

DNS A 10.1.2.3 www.somewebsite.com <EXPECTS 10.1.1.1>
```

- Optional secondary DNS support feature:

```
DNS A 192.168.42.11 somewebsite.com 192.168.42.12
```

  This results in accessing 192.168.42.11 and attempting to resolve somewebsite.com. If this fails, 192.168.42.12 (secondary DNS server) will be accessed, and another attempt made to resolve somewebsite.com.

  TIP: See Additional Examples, later in this topic for other example syntax and explanation.

  NOTE: Direct access of a forms database is not supported by the DNS structure. However, you can use the ASSIGN variable and forms database to build requests with a DNS protocol. For more information, see Using Form Databases with DNS, later in this topic.

Defining the Server Profile

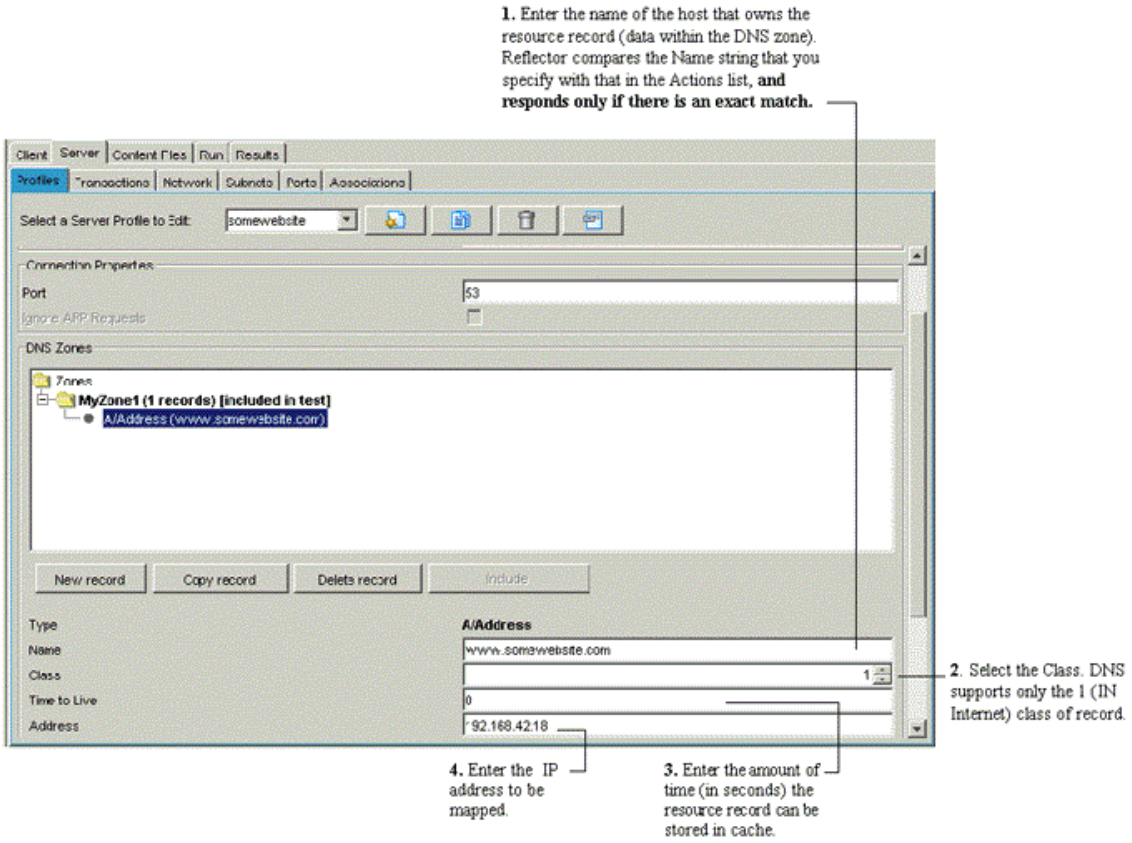
Complete this section only if you are defining a Device test.

To define a DNS server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **DNS** as the server type.
4. Enter the port where the server resides. The default port is 53.
5. Click the **New Zone** button. Zones are the equivalent of DNS zones. They help you organize the different DNS queries that you create. (If you have more than one zone, you must close an open zone before you can add a new one.)
6. Click the **New Record** button, and then select a record type (DNS query) from the drop-down list that appears. Avalanche supports the following types of DNS queries:
   - Address (A). Maps domain names to IPv4 addresses.
   - Address (AAAA). Maps domain names to IPv6 addresses.
   - Canonical Name (CNAME). DNS equivalent of an alias or symbolic link.
   - Mail Exchange (MX). Marks a server that has been designated as the mail server for a specific domain.

- Name Server (NS). Marks the beginning of a DNS zone and supplies the domain name of a name server for that zone. Typically you see it at the top of a zone, just after the SOA, and at the start of a subzone, where an NS and a paired A are all that is required to perform delegation.
- Pointer (PTR). Similar to CNAME in format, however, CNAME specifies an alias, while a PTR points to another location in the domain name space. The most important use of PTRs is to construct the in-addr.arpa domain, used to convert IP addresses to DNS names (the reverse of the normal process).
- Start of Authority (SOA). Marks the beginning of a DNS zone, and is typically seen as the first record in a name server for that domain

7. Complete entries in the fields that appear to support your record type. For definitions of all fields, see the Server Profiles Tab DNS Help. The following shows an example of an address query.



Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

NOTE: In the resulting statistics, each attempt is considered as attempted, and successful or unsuccessful. If the attempt was resolved on the first DNS server, the result is one attempted and one successful request. If the attempt was not resolved on the first DNS server, but resolved on the second DNS server, the result is two attempted, one successful, and one unsuccessful request. If both attempts fail, the result is two attempted and two unsuccessful requests.

Additional Examples

The following provides additional Action List examples for testing against a DNS server and for testing between simulated clients and servers.

Testing Against a DNS Server

The following provides examples for testing against a DNS server.

A Record

- Contacts the DNS server 192.168.42.11 to resolve the IP address for www.somewebsite.com. If the DNS server fails to respond, or no A record exists for www.somewebsite.com, the transaction fails.

  ```
  DNS A 192.168.42.11 www.somewebsite.com
  ```

- Contacts the DNS server 192.168.42.11 to resolve the IP address for www.somewebsite.com, requesting the DNS server to completely resolve the IP address for www.somewebsite.com. If the DNS server fails to respond, no A record exists for www.somewebsite.com, or the DNS server refuses recursive requests, the transaction fails. Note that recursion is disabled on some DNS servers, so this test works only when the DNS server supports recursive requests.

  ```
  DNS A 192.168.42.11 www.somewebsite.com <RECURSIVE>
  ```

- Contacts the DNS server 192.168.42.11 to resolve the IP address for www.somewebsite.com, expecting DNS to return the IP address 192.168.100.10 as the address for www.somewebsite.com. If the DNS Server fails to respond, no A record exists for www.somewebsite.com or any other address is returned, the transaction fails.

  ```
  DNS A 192.168.42.11 www.somewebsite.com <EXPECTS
  192.168.100.10>
  ```

- Contacts the DNS server 192.168.42.11 to resolve the IP address for www.somewebsite.com. If the DNS server fails to respond or no A record exists for www.somewebsite.com on the first DNS server, a second request is made to DNS server 192.168.42.12. If the secondary DNS server fails to respond or no A record exists for www.somewebsite.com on the second DNS server, the transaction fails.

```
DNS A 192.168.42.11 www.somewebsite.com 192.168.42.12
```

AAAA Record

When using the AAAA query, an IPv6 address is expected for the DNS name. The following example contacts the DNS server 10.1.2.3 to resolve the IP address for www.somewebsite.com, expecting DNS to return the IP address 2108::0200:FF:FE00:8301 as the address for www.somewebsite.com. If the DNS Server fails to respond, no AAAA record exists for www.somewebsite.com or any other address is returned, the transaction fails.

```
DNS AAAA 10.1.2.3 www.somewebsite.com <EXPECTS
2108::0200:FF:FE00:8301>
```

PTR Record

Contacts the DNS server 192.168.42.11 to resolve the DNS host name belonging to the IP address 192.168.100.10. If the DNS server fails to respond, or no PTR record exists on the DNS server for 192.168.100.10, the transaction fails.

```
DNS PTR 192.168.42.11 192.168.100.10
```

The following example uses IPv6 addresses:

```
DNS PTR 2000::0200:FF:FE00:0201 2222::0200:FF:FE00:101
```

CNAME Record

Contacts the DNS server 192.168.42.11 to resolve any alternative host name(s) for server1.somewebsite.com. If the DNS server fails to respond, or no CNAME record(s) exists for server1.somewebsite.com, the transaction fails.

```
DNS CNAME 192.168.42.11 server1.somewebsite.com
```

MX Record

Contacts the DNS server 192.168.42.11 to resolve the IP address for the mail server at somewebsite.com. If the DNS server fails to respond, or no MX address entry exists for somewebsite.com, the transaction fails.

```
DNS MX 192.168.42.11 somewebsite.com
```

NS Record

Contacts the DNS server 192.168.42.11 to resolve the IP address for the name server at somewebsite.com. If the DNS server fails to respond, or no NS address entry exists for somewebsite.com, the transaction fails.

```
DNS NS 192.168.42.11 somewebsite.com
```

SOA Record

Contacts the DNS server 192.168.42.11 to resolve the IP address for the DNS server which functions as the Start Of Authority for somewebsite.com. If the DNS server fails to respond, or no SOA address entry exists for somewebsite.com, the transaction fails.

```
DNS SOA 192.168.42.11 somewebsite.com
```

Testing Between Simulated Clients and Servers

The following commands show example syntax for testing between simulated clients and servers. Avalanche performs a literal request and response using the DNS protocol. DNS testing verifies if the devices under test are capable of passing or blocking DNS requests.

A Record

- Contacts the DNS server 192.168.42.11 to resolve the IP address for www.somewebsite.com. If the DNS server fails to respond, or no A record exists on the DNS server for www.somewebsite.com, the transaction fails.

  ```
  DNS A 192.168.42.11 www.somewebsite.com
  ```

- Contacts the DNS server 192.168.42.11 to resolve the IP address for www.somewebsite.com, requesting the DNS server to completely resolve the IP address for www.somewebsite.com. If the DNS server fails to respond, or no A record exists on the DNS server for www.somewebsite.com, the transaction fails. Note that recursion is irrelevant to testing between simulated clients and servers, because the simulated server always returns the full IP address.

  ```
  DNS A 192.168.42.11 www.somewebsite.com <RECURSIVE>
  ```

- Contacts the DNS server 192.168.42.11 to resolve the IP address for www.somewebsite.com, expecting the DNS server to return the IP address 192.168.100.10 as the address for www.somewebsite.com. If the DNS Server fails to respond, no A record exists on the DNS server for www.somewebsite.com, or any other address is returned, the transaction fails.

  ```
  DNS A 192.168.42.11 www.somewebsite.com <EXPECTS
  192.168.100.10>
  ```

- Contacts the DNS server 192.168.42.11 to resolve the IP address for www.somewebsite.com. If the DNS server fails to respond or no A record exists on the DNS server for www.somewebsite.com on the first DNS server, a second request is made to DNS server 192.168.42.12. If the secondary DNS server fails to respond or no A record exists on the DNS server for www.somewebsite.com on the second DNS server, the transaction fails.

  ```
  DNS A 192.168.42.11 www.somewebsite.com 192.168.42.12
  ```

PTR Record

Contacts the DNS server 192.168.42.11 to resolve the DNS host name belonging to the IP address 192.168.100.10. If the DNS server fails to

respond, or no PTR record exists on the DNS server for 192.168.100.10, the transaction fails.

```
DNS PTR 192.168.42.11 192.168.100.10
```

The following example uses IPv6 addresses:

```
DNS PTR 2000::0200:FF:FE00:0201 2222::0200:FF:FE00:101
```

CNAME Record

Contacts the DNS server 192.168.42.11 to resolve any alternative host name(s) for server1.somewebsite.com. If the DNS server fails to respond, or no CNAME record(s) exists on the DNS server for server1.somewebsite.com, the transaction fails.

```
DNS CNAME 192.168.42.11 server1.somewebsite.com
```

MX Record

Contacts the DNS server 192.168.42.11 to resolve the IP address for the mail server at somewebsite.com. If the DNS server fails to respond, or no MX address entry exists on the DNS server for somewebsite.com, the transaction fails.

```
DNS MX 192.168.42.11 somewebsite.com
```

NS Record

Contacts the DNS server 192.168.42.11 to resolve the IP address for the name server at somewebsite.com. If the DNS server fails to respond, or no NS address entry exists on the DNS server for somewebsite.com, the transaction fails.

```
DNS NS 192.168.42.11 somewebsite.com
```

SOA Record

Contacts the DNS server 192.168.42.11 to resolve the IP address for the DNS server which functions as the Start Of Authority for somewebsite.com. If the DNS server fails to respond, or no SOA address entry exists on the DNS server for somewebsite.com, the transaction fails.

```
DNS SOA 192.168.42.11 somewebsite.com
```

Using Forms Databases with DNS

Direct access of a forms database is not supported by the DNS structure. You can, however, use the ASSIGN variable and forms database to build requests with a DNS protocol. The following provides an example of assigning a forms database:

```
ASSIGN VARIABLE <Rec dnsdb.$1>

ASSIGN VARIABLE <Lookup dnsdb.$2>

ASSIGN VARIABLE <Expect dnsdb.$3>
```

You can then use the variables in an Action list by using the APPLY action:

```
DNS <APPLY Rec> 192.168.44.11 <APPLY Lookup> <EXPECT <APPLY
Expect>>
```

An example forms database supporting the previous Action list is as follows:

```
A, www.somewebsite1.com,1.1.1.1

A, www.somewebsite3.com,3.3.3.3

A, www.somewebsite4.com,4.4.4.4

A, www.somewebsite5.com,5.5.5.5

A, www.somewebsite6.com,6.6.6.6

A, www.somewebsite7.com,7.7.7.7

A, www.somewebsite8.com,8.8.8.8

A, www.somewebsite9.com,9.9.9.9

A, www.somewebsite10.com,10.10.10.10
```

Testing Ethernet Capture Replay

To complete Ethernet Capture Replay information and run the test:

1. Learn about Avalanche Ethernet Capture Replay testing.
2. Define Capture Replay Ethernet information for the client.
3. Define Capture Replay Ethernet information for the server. (Device test only)
4. Add an Actions list.
5. Run the test and review results.

About Ethernet Capture Replay Testing

Ethernet Capture Replay allows you to replay a trace file captured on the network between two devices at Layer 2. It plays back everything between the two devices, that is, two unique MAC addresses. It allows you to upload the trace file, and filter the conversation based on the MAC addresses. This functionality is useful when troubleshooting a network crash. If you have a trace file captured during the network crash, you can use this file to reproduce the exact crash scenario, from a transmission/packet level, for troubleshooting the network.

Defining Client Capture Replay Ethernet Information

Define the Ethernet content file that you want to use.

To configure the client:

1. Click the **Client Actions** tab, and then the **CRETH** tab.
2. In the left pane of the **CRETH** tab, select the Capture Replay Ethernet profile that you want to use, or click the **New** button  and enter a name to create a new profile. You will reference the name of the Capture Replay Ethernet profile when you create an Action list to test Capture Replay Ethernet.
3. In the right pane of the **CRETH** tab, select the name of the Ethernet content file that you want to use from the **Eth Content File** drop-down menu for the selected Capture Replay Ethernet profile, or click the **New** button to create a new Ethernet content file. Clicking the New button launches the PCAP Wizard. This wizard guides you through the steps to create a new Ethernet content file. It converts your selected trace file into an intermediate format (XML) for use by Avalanche. (If you are not using the PCAP Wizard, you can skip the remaining steps in this section.)
4. In the PCAP Wizard, click the **Browse** button to navigate to and select the trace file that you want to use, and then click **Next**.
5. Enter the name of the output XML file, and click **Next**. (If you want to keep the default name, just click **Next**.) The wizard displays the frame-by-frame content of the trace file.

6. You can filter the trace to show only the frames between two unique MAC addresses by selecting the **Filter** checkbox. You can then choose the MAC addresses by clicking on any frame. (If the trace is already filtered, this will have no effect.)

   TIP: For complex file captures, it is recommended that you *prefilter* the packets you want to play back prior to importing them, using an external capture editor such as ClearSite ™.

7. Click **Finish** to close the wizard. The resulting XML file appears in the Eth Content File drop-down menu, and the wizard stores this file as a content file.

Defining Server Capture Replay Ethernet Information

Complete this section only if you are defining a Device test. Use the **Server Profiles** tab to configure information for a CapRepEth server.

To define a CapRepEth server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  and create a new profile.
3. Enter a description for the profile, and then select **CapRepEth** as the server type.
4. Enter the port where the server resides.
5. Select options and enter information in the fields.
6. Select the name of the Ethernet content file that you want to use from the **Eth Content File** drop-down menu, or click the **New** button to create a new Ethernet content file. Clicking the New button launches the PCAP Wizard (described above in the client section). You can also use the **Delete** button to delete the selected Ethernet content file.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

To create an Actions list:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter a description for the Actions list.
4. Enter information that specifies the IP address and port of the server, as well as the Capture Replay Ethernet profile to use. See the following syntax and example.

CAUTION: Do not include a trailing slash (/) after the IP address or an error will occur.

Syntax

```
CRETH://ip address[:port] PROFILE=profile name
```

- ip address[:port]—IP address and port of the server. (Port is optional; if it is not included, then 2000 is assumed.)
- PROFILE=profile name—Specifies the Capture Replay Ethernet profile to use with this Action list. The profile name that you specify is the profile defined in the CRETH tab of the Client Actions tab.

5. Example

```
CRETH://192.168.42.11 PROFILE=caprepeth_profile
```

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run** Test  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing FTP

To complete FTP-specific information and run the test:

1. Learn about Avalanche FTP testing.
2. Add an Actions list.
3. Use FTP PUT command.
4. Use Active FTP.
5. Define the server profile. (Device test only)
6. Upload your own contents on the server.
7. Run the test and review results.

About FTP Testing

Avalanche uses FTP to perform a simple file transfer from a specified server. You can emulate an FTP server, establishing a connection with a client, returning various files in binary mode or ASCII mode, and then terminating the session. You can also use APPLY with an FTP action list to apply a previously defined variable value.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom. You can create an Action list for Application tests or for Device tests.

For each FTP Action list entry, Avalanche establishes a connection, requests a file transfer, and then terminates the connection when it has completed.

To create an Actions list for an FTP application test (client emulation only):

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter a description for the Actions list.
4. Enter information that simulates an FTP transfer. For each FTP entry, enter 1 preceding the URI (to signify a top-level retrieval) and then the action. Refer to the following syntax and examples.

Syntax - Basic

```
1 ftp://server_IP_address /filename <USER=username
PASSWD=password> <MODE=BINARY> <BURST_SIZE=burst_bytes>
<BURST_INTERVAL=burst_interval_msecs>

or

1 ftp://server_IP_address /filename <USER_VAR=<APPLY
user_variable> PASSWD_VAR=<APPLY password_variable>> <MODE=
BINARY> <BURST_SIZE=burst_bytes>
<BURST_INTERVAL=burst_interval_msecs>
```

- server_IP_address—The address of the server, such as 10.1.79.1
- filename—The file that you are transferring, such as abcfile.txt
- USER—The user name used for authentication
- PASSWD—The password used for authentication
- MODE—BINARY or ASCII
- BURST_SIZE—The number of bytes of data to generate in each client request
- BURST_INTERVAL—The number of milliseconds to wait in between client requests
- USER_VAR—A user variable that contains the user name for authentication
- PASSWD_VAR—A password variable that contains the password for authentication

Examples - Basic

- USER and PASSWD:

  ```
  1 ftp://10.1.79.1/abcfile.txt <USER=ann PASSWD=ann@somewebsite.com>
  <MODE=BINARY>
  ```

- USER variable and PASSWD variable:

  ```
  1 ftp://10.1.79.1/abcfile.txt <USER_VAR=<APPLY groupnames>
  PASSWD_VAR=<APPLY passwords>> <MODE=BINARY>
  ```

  NOTE: To use the APPLY command you must first ASSIGN a run-time value to the variable.

- ASCII mode:

  1 ftp://10.1.79.1/1Kb <USER=anonymous PASSWD=anonymous> <MODE=ASCII>

- BURST:

  1 ftp://10.1.79.1/abcfile.txt <USER=anonymous PASSWD=anonymous>
  <MODE=BINARY> <BURST_SIZE = 1024> <BURST_INTERVAL=3>

To create an Actions list for an FTP Device test:

With the FTP server emulation in a Device test, you have access to a virtual file system that consists of any size file. The size is determined by the URL's postfix in the FTP Action list. For example

- 1b: a file with a size of 1 byte
- 10b: a file with a size of 10 bytes
- 100b: a file with a size of a 100 bytes
- 1k: a file with a size of 1 Kbytes
- 10k: a file with a size of 10 Kbytes
- 100k: a file with a size of 100 Kbytes
- 1m: a file with a size of 1 Mbyte
- 10m: a file with a size of 10 Mbytes
- 100m: a file with a size of a 100 Mbytes
- 1g: a file with a size of 1 Gigabyte.

Syntax

```
1 ftp://server_IP_address/file_size
```

- server_IP_address—The address of the server, such as 10.1.79.1
- file_size—The size of the file, such as 1b for 1 byte

Example

For a file size of 1 Gigabyte

```
1 ftp://192.168.42.11/1g
```

The above example uses an IPv4 address. The following example uses an IPv6 address:

```
1 ftp://[2106::0200:FF:FE00:8101]/1g
```

Using FTP PUT Command

You can send data from the client to the server using the FTP PUT command. You can either specify the file size or a specific file to send to the server.

To send a file from the client, you need to first upload the file directory. If you have already uploaded a directory for the server, as explained in the section, "Uploading Contents on the Server," you can use the files from the same directory. If you want to use a different directory for the files to be sent by the client, use the Directories tab on the Client Actions tab.

Syntax

NOTE: You should use the FSTREE keyword only once in your Actions list. If you use it multiple times, only the last one will apply to your test.

```
FSTREE=directory_name

1 ftp://server_IP_address <PUT=filename>

or

1 ftp://server_IP_address <PUT=filename SIZE=size>
```

- directory_name—The named file directory that contains the file to send from the client
- server_IP_address—The address of the server, such as 10.1.79.1
- filename—The file to send from the client
- size—The number of bytes to send from the file

Examples

- A client logs in using a guest username and password, and then sends the file Text.txt from the named file directory NEW:

  ```
  FSTREE=NEW

  1 ftp://192.168.41.11 <PUT = Text.txt>
  ```

- A client logs in using a specific username and password. The client then sends a PORT command to use Active FTP mode, so that the server opens the data connection. Once the server opens the data connection, the client sends the file Text.txt from the named file directory NEW:

  ```
  FSTREE=NEW

  1 ftp://192.168.41.11 <USER=user1 PASSWD=pass1> <PUT =
  Text.txt> <ACTIVE_FTP_MODE = YES>
  ```

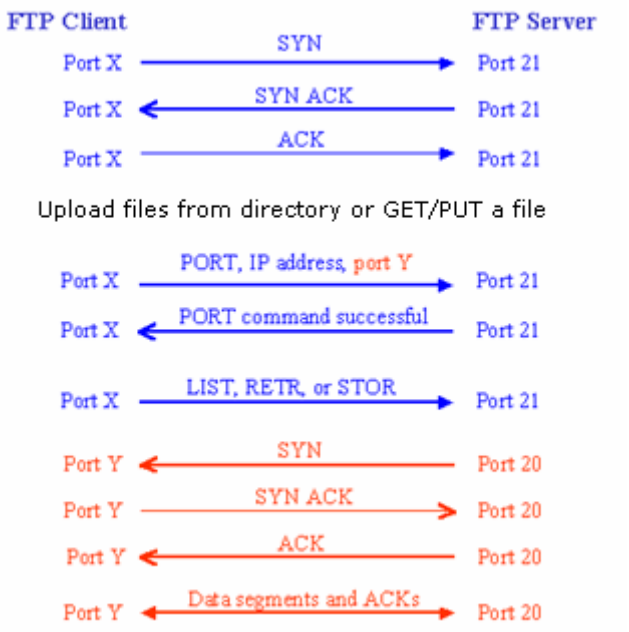
Using Active FTP

In Active FTP mode, the client connects from a random unprivileged port (N > 1023) to the FTP server's command port 21. The client then starts listening to port N+1, and sends the FTP command PORT N+1 to the FTP server. The server then connects back to the client's specified data port from its local data port 20.

From the server-side firewall's standpoint, to support Active FTP mode, the following communication channels need to be opened:

- FTP server's port 21 from anywhere (client initiates connection)
- FTP server's port 21 to ports > 1023 (server responds to client's control port)
- FTP server's port 20 to ports > 1023 (server initiates data connection to client's data port)

- FTP server's port 20 from ports > 1023 (Client sends ACKs to server's data port)



To enable Active FTP mode on the client side, add the following to your FTP action:

```
<ACTIVE_FTP_MODE = YES>
```

Examples

- Log in to an FTP server using the guest username and password, and then GET the file Text.txt from the server in Active FTP mode (that is, the server opens the data connection):

  ```
  1 ftp://10.10.10.201 <GET = Text.txt> <ACTIVE_FTP_MODE = YES>
  ```

- Log in to an FTP server using a specific username and password, and then PUT the file Text.txt on the server in Active FTP mode (that is, the server opens the data connection):

  ```
  1 ftp://10.10.10.201 <USER=user1 PASSWD=pass1> <PUT =
  Text.txt> <ACTIVE_FTP_MODE = YES>
  ```

- Active GET

  ```
  FSTREE=FILES

  1 ftp://126.1.1.10/ <USER=anonymous PASSWD=anonymous>
  <GET=1Kb> <ACTIVE_FTP_MODE = YES>
  ```

- Active PUT

```
FSTREE=FILES

1 ftp://126.1.1.10/ <USER=anonymous PASSWD=anonymous>
<PUT=1Kb> <ACTIVE_FTP_MODE = YES>
```

Defining the Server Profile

Complete this section only if you are defining a Device test.

To create an FTP Test:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the server profile, and then select **FTP** as the server type.
4. Enter the FTP server port. The default is 21.
5. Enter **Burst** parameters for controlling bandwidth.

IMPORTANT: If the TCP Inactivity Timer in the Server Network tab (or Client Network tab) is less than two seconds, the FTP control connection could be aborted. Otherwise, the FTP application has a work-around to send a Keep Alive packet over the control connection. The data connection will not be idle, and therefore, is unaffected by the TCP Inactivity Timer.

Uploading Contents on the Server

You can upload your own contents on the server, and then issue a GET command from the client to download the contents from the server. You can upload as many directories as you want, but you can select only one directory per server.

CAUTION: Both the size and number of files that you upload can adversely affect performance, especially during the test preparation phase. While the load generators can support up to 1 GB of files, less than 4 MB are recommended.

To upload your own contents on the server:

1. In the Server Profiles FTP tab, click the **New** button  in the Options pane. A directory dialog box appears.
2. Click the **Browse** button, navigate to and select the directory that you want to use, and then click **Open**.
3. You can change the **Directory Name**, and then click **OK**.

   NOTE: The directory name (and filenames) must contain only alpha-numeric characters and underscores. Spaces, other special characters, and URL encoding are not allowed. The directory name is case sensitive.
4. Use the other buttons in the Options pane to copy, edit, or delete a named directory.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing HTTP

To complete information and run the test:

1. Learn about Avalanche HTTP testing.
2. Add an Actions list.
3. Define information for the client.
4. Define information for the server. (Device test only)
5. Define server transaction objects. (Device test only)
6. Run the test and review results.

TIP: See HTTP example test scenarios in the Help for the client and server behavior that is a result of various client/server parameter settings.

About HTTP Testing

Avalanche uses HTTP to request a series of objects from a server, or to submit forms databases by using embedded strings. To accomplish this, an HTTP Action list uses three elements: Level, Method, and URI. Sometimes the URI is appended with an embedded string that affects client-server exchanges. In addition to using an Action list, Avalanche provides a series of parameters that you can use to configure the client profile. For example, you can set user behavior, the transport mode, browser emulation, and SSL configuration.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom. You can create an Action list for Application tests or for Device tests. In addition to defining HTTP Level, Method, and URI, Avalanche supports additional HTTP actions that you can use to perform a variety of other tasks such as searching and assigning variables. Actions include:

- HTTP Match and Match Not
- HTTP Stop
- HTTP Search
- HTTP Assign Variables
- HTTP Apply
- HTTP Dynamic URL
- HTTP Set Runtime Cookie Action
- HTTP Additional Headers with Variables
- HTTP URL Encode
- HTTP Response Body Decompress

- HTTP Pipelining

For more information about these actions including syntax and examples, see the associated topic in the Help.

To create an Actions list:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter a description for the Actions list.
4. Enter information in the Action list as defined by the following syntax.

Syntax

The URI states the protocol, the server, and the requested object. HTTP Action list entries must state the protocol (http://) on each line.

```
1 or 2 method HTTP:// server ip address/requested object
```

Examples

- GET method:

```
1 GET http://10.10.10.10/

1 GET http://192.168.44.1/Transaction_1

1 GET http://www.somewebsite.com/

2 GET http://www.somewebsite.com/images/about.gif
```

- HEAD method:

```
1 HEAD http://www.somewebsite.com/support/support.shtml

2 HEAD http://www.somewebsite.com/images/logo1.gif
```

- POST method:

```
1 POST http://www.somewebsite.com/ username=<filename.$1>
password=<filename.$2>

NEXT ROW <filename>

1 POST http://www.somewebsite.com/cgi-bin/ username=<filename.$1>
password=<purchase.$1>
```

Defining Client Information

Define the client HTTP parameters that you want to use. See the Help for more information about the fields in a Client Profile pane.

TIP: If you want to use client default values, you can skip the following procedure.

To configure the client:

1. Click the **Client** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile or select the profile with which you want to work.
3. Enter a description for the server profile, and then select **HTTP** as the server type.
4. In the User Behavior pane, enter information to define a profile that simulates how a user behaves, such as the amount of time a user spends on a page at a site, or to simulate users logging in to a site. You also use this pane to create and associate different files with a test. For example, you can:
    - create search criteria files for use with an HTTP Search, Match, or Match Not action.
    - create HTTP Body content files that you can send by using an HTTP POST command.
    - create forms databases that you can use to send information during a test. You can use a forms database to simulate users submitting GET and POST requests to an Internet site.
    - create DNS host mapping files for mapping host names to IP addresses in an Action list.
5. Complete entries in the Browser Emulation and Protocol panes to set parameters that determine how simulated users communicate with the targeted network.

    If you select a standard browser such as Microsoft Internet Explorer or Netscape Navigator, default protocol entries appear in the Protocol Level and Browser Header panes.

    If you select User Defined browser, the fields in the Browser Header pane become available so that you can define header strings.
6. To test Basic or Proxy Authentication, complete entries in the Authorization pane.

Defining the Server Profile

Complete this section only if you are defining a Device test and if you want to change default values.

To define a server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **HTTP** as the server type.

4. Enter the port where the server resides. The default port is 80.
5. Make selections in the Connection Properties pane, such as connection termination. You can select an item from the Transaction Profile drop-down menu as the object that the server responds with by default, if no object is included in the request on the Client Actions list. (See the next section "Defining Server Transaction Objects.")
6. Complete entries in the Server Emulation pane, such as server type, transaction profile, and protocol level.
7. To enable cookie support, select **Cookies**, and enter the settings that you want to use.

Defining Server Transaction Objects

Complete this section only if you are defining a Device test. You use the **Server Transactions** tab to configure the HTTP/HTTPS server transactions for tests. To specify an alternate (or custom) Transaction Profile in a URL list, see Customizing HTTP/HTTPS Responses in the Help.

To define a transaction object:

1. Click the **Server** tab, and then the **Transactions** tab.

   Select the default transaction from the Transaction Profile drop-down menu.
2. Select the options on the Server Transactions tab that you want to use. The data body types and body sizes are especially important for your test.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing IMAP4

To complete IMAP4-specific information and run the test:

1. Learn about IMAP4 testing.
2. Add an Actions list.
3. Define the client profile.
4. Define the server profile. (Device test only)
5. Run the test and review results.

About IMAP4 Testing

An IMAP4 test uses Avalanche to simulate a client logging into a server, executing supported commands, fetching various mail messages from an IMAP4 server, and then closing the session. An emulated IMAP4 server can respond to IMAP4 commands generated by Avalanche for IMAP4 traffic, allowing you to stress test network devices that are aware of mail traffic.

The Action list describes the transactions and controls their sequence of execution. By default, IMAP4 uses port 143 for transport, and levels are not relevant to the IMAP4 protocol.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

To add an Actions list for an IMAP4 test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.

Enter information that simulates an IMAP4 client sending information. Refer to the following syntax and examples.

Syntax

imap://IP_Address:port USER=string PASSWD=string commands

NOTE: If you are using a Device test, the USER and PASSWD strings must be the same.

If using a forms database to supply username and password

imap://IP_Address:port USER=filename.$1 PASSWD=filename.$2 commands

- IP_Address-Identifies the IP address of the test server.
- port-The port where the server resides.

- USER-Specifies the user's name. For a Device test, this must match mailbox name configured in Server Profiles tab. If using a forms database, specifies the filename.
- PASSWD-Specifies the user's password. For a Device test, this must match mailbox name configured in Server Profiles tab. If using a forms database, specifies the filename.
- command-Avalanche supports the following commands:

  NOTE: The LSUB command is not currently supported.

  - CAPABILITY-Requests a listing of capabilities that the server supports.
  - NOOP-Does nothing.
  - SELECT = folder-Selects the mailbox specified by folder, so that messages in the mailbox can be accessed. Folder is a string specifying a pre-existing folder.
  - CHECK-Requests a checkpoint of the currently selected mailbox.
  - CLOSE-Permanently removes all messages that have the \Deleted flag set from the currently selected mailbox, and returns to the authenticated state from the selected state. No untagged EXPUNGE responses are sent.
  - EXPUNGE-Permanently removes all messages that have the \Deleted flag set from the currently selected mailbox.
  - STATUS = folder-Requests the status of the mailbox specified by folder. Folder is a string specifying a pre-existing folder.
  - LIST-Returns a subset of names from the complete set of all names available to the client. Equivalent to issuing IMAP command: LIST "" "*".
  - FETCH_BODY = sequence-Retrieves message bodies associated with messages specified by sequence. Sequence is a string that has the same format as IMAP sequence. For example: FETCH_BODY="1,3,6:10,14:*"
  - FETCH_HEADERS = sequence-Retrieves headers associated with messages specified by sequence.
  - STORE = sequence-Marks messages specified by sequence as deleted (sets \Deleted flag).
  - LOGOUT-Informs the server that the client is done with the connection.

Examples

- Action list line entry for IMAP4

  imap://192.168.44.1 USER=admin PASSWD=admin NOOP LIST CAPABILITY STATUS = INBOX SELECT = INBOX CHECK EXPUNGE CLOSE LOGOUT

- Action list for forms database access for IMAP4

  imap://192.168.44.1 USER=imapdb.$1 PASSWD=imapdb.$2 NOOP LIST CAPABILITY STATUS = INBOX SELECT = INBOX CHECK EXPUNGE CLOSE LOGOUT

  NOTE: If you are using a Device test, the USER and PASSWD names must be the same, and must match the mailbox name configured in the Server Profiles tab.

  An example forms database supporting the previous Action is as follows:

```
username1,username1

username2,username2

username3,username3

username4,username4

username5,username5

username6,username6

username7,username7

username8,username8

username9,username9

username10,username10
```

Defining the Server Profile

Complete this section only if you are defining a Device test. Use the **Server Profiles** tab to configure information for an IMAP4 server.

Use a server profile to specify the type of messages sent to a mailbox and the mailbox name. The USER and PASSWD names identified in the Actions list must match the mailbox name you identify in the Server Profiles tab.

To define a server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **IMAP4** as the server type.
4. Enter the port where the server resides. The default port is 143.

5. In the IMAP4 Server Emulation Pane, click the **New** button  to specify a new mailbox name.
6. Enter a unique, alphanumeric mailbox name. The only special character you can use is underscore (_). The mailbox name and password associated with the mailbox are the same.
7. In the IMAP4 Server Emulation Pane, click the **New** button below the table to add folder information for the selected mailbox in the User Mailbox drop-down menu. For more information about IMAP4 fields, see Server Profiles Tab IMAP4 in the Help.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

NOTE: The statistical data for IMAP4 tests are described in IMAP4 Statistics in the Help.

Testing MM4

To complete MM4-specific information and run the test:

1. Learn about MM4 testing.
2. Add an Actions list.
3. Define the client profile.
4. Define the server profile. (Device test only)
5. Run the test and review results.

About MM4 Testing

With Multimedia Messaging Service (MMS), cell phone providers can send and receive multimedia messages. MM4 provides communication between the Multimedia Messaging Service Relays/Servers, also referred to as MMS Centers (MMSCs). The sender is the originator MMSC, and the receiver is the recipient MMSC.

By configuring an MM4 test, you can use Avalanche to emulate an originator MMS Relay/Server, simulating the forwarding of MM4 messages to another MMS Relay/Server, as well as to emulate a recipient MMS Relay/Server. For more information about the role of the client and server, see Emulating MM4 in the Help.

NOTE: This topic uses simulated clients and servers to explain the MM4 test, however, you can alternatively use your own client and server devices.

Adding an Actions List

An Action list defines the addresses, requests, and message for the test. The format of the MM4 Action list is similar to an SMTP Action list. You identify an envelope (FROM and TO information), a subject, and a message.

However, MM4 Action lists allow for three additional optional parameters—ACK_FORWARD, DELIVERY_REPORT, and READ_REPLY—to specify response and request messages.

Syntax

```
mm4://mmsc_ip_address_relay_server FROM=<sender_address>
TO=<recipient_address> [SUBJECT=<subject>] DATA=<FIXED size> [ACK_FORWARD][
DELIVERY_REPORT] [READ_REPLY]
```

NOTE: Use a DATA or MM4_BODY_FILE tag to define the message that you want to send. You cannot use DATA and BODY tags at the same time, however you must use one to indicate the data for the message. You can use MM4_ATTACH_FILE optionally with MM4_BODY_FILE.

- mmsc_ip_address_relay_server—Identifies the IP address of the intended MMS Relay/Server. (required)
- FROM=sender_address—Specifies the sender's (originator's) email address and domain name. (required)

- TO=recipient_address—Specifies the recipient (terminator's) email address and domain name. You can identify multiple addresses, separating each address with a comma or a space, such as Joe@somewebsite.com, Mary@somewebsite.com, or Joe@somewebsite.com Mary@somewebsite.com. (required)
- SUBJECT=subject—Text that specifies the subject information. The subject content cannot contain spaces unless you include them within quotation marks, such as "". If you don't enter a subject, a randomly generated subject is used for the test. (optional)
- DATA-Specifies the content of the message sent. (One of the following is required.)
    - DATA=<FIXED size> An auto-generated message body that is the size that you specify. This length does not include the message header.
    - DATA=<RANDOM=UNIFORM min_size max_size> An auto-generated message body that is randomly generated to be between the minimum and maximum sizes that you specify.
    - MM4_BODY_FILE=<"filename" "content type"> The filename and content type of the body that is included as the MIME (Multipurpose Internet Mail Extensions) part that represents the body of the message, such as the body of an email message. Format the contents of the file that you specify for the MM4_BODY_FILE as ASCII text, since it is used as is.
    - MM4_ATTACH_FILE=<"filename" "content type"> (You can use this tag optionally with MM4_BODY_FILE.) The filename and content type of an attachment that is included in the MIME message, such as an attachment that you would add to an email message. The contents of the file that you specify for an MM4_ATTACH_FILE will be base64-encoded. You can use the parameter multiple times to attach multiple files. You can specify any content type, such as plain text or image. If you use this parameter, you must also use the MM4_BODY_FILE parameter.

      IMPORTANT: You must add the files that you specify as BODY or ATTACH files as content files in the Content Files Tab before you run the test.
- ACK_FORWARD—Requests that the recipient send a forward response (MM4_forward.RES). (optional)
- DELIVERY_REPORT—Requests that the recipient send a delivery_reportrequest (MM4_delivery_report.REQ) (optional)
- READ_REPLY — Requests that the recipient send a read_reply_reportrequest (MM4_read_reply_report.REQ). (optional)

NOTES: If you include the three previous parameters in an Action list, you can additionally control the percentage of requests and responses from the server by setting values in the MM4 percentage fields on the Server Profiles tab. Using these fields you can choose to deny whatever percentage of the requests/responses you want. For example, to deny all requests, you can set the value to 100% denial. For a diagram showing example requests and responses sent between a client and server, see Emulating MM4.

Use the MM4 fields in the Client and Server Profile tabs to set minimum and maximum time-out values for the MM4 transactions and session.

Examples

The following example sends an MM4_forward_REQ message with a 100-byte body from one user to another user. The recipient is requested to respond with an MM4_forward.RES, MM4_delivery_report.REQ, and MM4_read_reply_report.REQ messages.

```
mm4://1.2.3.4 FROM=<user1@somewebsite1.com>
TO=<user2@somewebsite2.com> SUBJECT=<my_subject> DATA=<FIXED
100> ACK_FORWARD DELIVERY_REPORT READ_REPLY
```

The following example sends an MM4_forward_REQ message with a body file and three attached files from one user to another user. The recipient is requested to respond with an MM4_forward.RES. Other requests and responses are not required.

```
mm4://1.2.3.4 FROM=<user1@somewebsite1.com>
TO=<user2@somewebsite2.com> SUBJECT=<my_subject>
MM4_BODY_FILE=<"mymsg" "plain/text">

MM4_ATTACH_FILE=<"my_attacheddoc" "plain/text">

MM4_ATTACH_FILE=<"my_picture" "image/jpg">

MM4_ATTACH_FILE=<"my_audio" "audio/amr">

ACK_FORWARD
```

Defining the Client Profile

Use the MM4 pane of the **DNS/MM4/SIP/Streaming** tab on the Client Profiles tab to define information to emulate an originator MMS Relay/Server, simulating the forwarding of MM4 messages to another MMS Relay/Server. Each MM4 message is sent by using a separate SMTP session on a client and a server.

To define a client profile:

1. Click the **Client** tab, and then the **Profiles** tab.
2. Select the client profile with which you want to work.
3. In the MM4 pane of the **DNS/MM4/SIP/Streaming** tab, enter the originator port, response timeout, and session timeout. See the Client Profiles MM4 Help for detailed definitions.

Defining the Server Profile

Use the MM4 fields on the Server Profile tab to emulate a recipient MMS Relay/Server. Each MM4 message is sent by using a separate SMTP session on a client and a server. For detailed definitions of the server fields, see Server Profiles Tab MM4 in the Help.

To define a server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **MM4** as the server type.
4. Enter the port where the server resides. The default port is 25.
5. Enter information to define the MM4 settings, including the following:
    - MM4 emulation to define timeouts and percentage of report request denied and acknowledged.
    - Request status to define the percentage of each report status response that is randomly sent by the server as an MM4_forward.RES.
    - Message status to define the percentage of each request status that is randomly sent as an MM4_delivery_report.REQ.
    - Read status to define the percentage of each read status that is randomly sent by the server as an MM4_read_reply_report.REQ.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing POP3

To complete POP3-specific information and run the test:

1. Learn about Avalanche POP3 testing.
2. Add an Actions list.
3. Define the server profile. (Device test only)
4. Run the test and review results.

About POP3 Testing

A POP3 test uses Avalanche to simulate a client logging into a server, executing supported commands, fetching various mail messages from a POP3 server, and then closing the session. An emulated POP3 server can respond to POP3 commands generated by Avalanche for POP3 traffic, allowing you to stress test network devices that are aware of mail traffic.

The Action list describes the transactions and controls their sequence of execution. By default, POP3 uses port 110 for transport, and levels are not relevant to the POP3 protocol.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

To add an Actions list for a POP3 test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter information that simulates a POP3 client sending information. Refer to the following syntax and examples.

Syntax

```
pop3://IP_Address USER=string PASSWD=string commands
```

If using a forms database to supply username and password

```
pop3://IP_Address USER=filename.$1 PASSWD=filename.$2
commands
```

- IP_Address-Identifies the web address of the test server.
- USER-Specifies the user's name. Must match mailbox name configured in Server Profiles tab. If using a forms database, specifies the filename.
- PASSWD-Specifies the user's password. Must match mailbox name configured in Server Profiles tab. If using a forms database, specifies the filename.
- commands-Avalanche supports the following commands:

- RETR-Retrieves all the messages from the server. Precede this command with the CHECK command.
- RETR=integer-Retrieves the message number specified by the integer from the server.

  NOTE: If you are using an emulated server, you may receive some unsuccessful sessions when you specify a specific message number to retrieve. The emulated server randomizes the actual size of the mailbox, and if the specific message is larger than the mailbox, the session may fail.
- CHECK, CHECK = integer-Checks the number of messages on the server. If an integer follows the command, Avalanche checks that the integer is equal to the number of messages on the server. This is equivalent to the POP3 STAT command.
- LIST-Issues the equivalent of the POP3 LIST command. The server should issue a multi-line response for each message on the server.
- LIST=UIDL-Issues the equivalent of the POP3 UIDL command. The server should issue a multi-line response for each message's unique identifier.
- DEL (or DELETE)-Deletes all the messages on the server. Precede this command with the CHECK command. If used in conjunction with the RETR command, the RETR command should precede the DEL command.

Examples

Action list line entries for POP3

```
POP3://192.168.44.11 USER=admin PASSWD=admin CHECK LIST

POP3://www.somewebsite.com USER=admin PASSWD=admin CHECK LIST
```

- Action list for forms database access for POP3

  ```
  POP3://192.168.44.11 USER=pop3db.$1 PASSWD=pop3db.$2 CHECK
  RETRIEVE
  ```

  NOTE: USER and PASSWD names must be the same and must match the mailbox name configured in the Server Profiles tab.

  An example forms database supporting the previous Action is as follows:

  ```
  username1,username1

  username2,username2

  username3,username3

  username4,username4

  username5,username5

  username6,username6
  ```

```
username7,username7

username8,username8

username9,username9

username10,username10
```

- An IPv4 address

  ```
  POP3://192.168.44.11 USER=admin PASSWD=admin CHECK LIST
  ```

- An IPv6 address

  ```
  POP3://[2107::0200:FF:FE00:8202] USER=admin PASSWD=admin
  CHECK LIST
  ```

Defining the Server Profile

Complete this section only if you are defining a Device test. Use a server profile to specify the type of messages sent to a mailbox and the mailbox name. The USER and PASSWD names identified in the Actions list must match the mailbox name you identify in the Server Profiles tab.

To define a POP3 server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **POP3** as the server type.
4. Enter the port where the server resides. The default port is 110.
5. Click the **Add** button to add a new mailbox.
6. Enter a unique, alphanumeric mailbox name. The only special characters you can use are dash (-), underscore (_), and period (.). The mailbox name and password associated with the mailbox are the same.
7. Select either **Programmed Response** or **Use Selected Files** as the mailbox type, and then complete configuration information:

   Programmed Response specifies a range of messages and message lengths. Each time there is a connection to a mailbox, Avalanche sends a random number of simulated POP3 messages within the specified range and size.

   Use Selected Files specifies POP3 message files. Each time you connect to the mailbox, Avalanche sends a POP3 message file that you have created and identified. Do not include the following characters in a POP3 message file: <, >, \, or ;.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

NOTE: The statistical data for POP3 tests are described in POP3 Statistics in the Help.

Testing RTMP

To complete RTMP-specific information and run the test:

1. Learn about Avalanche RTMP testing.
2. Define information for the client.
3. Add an Actions list.
4. Run the test and review results.

About RTMP Testing

Real Time Messaging Protocol (RTMP) is a proprietary protocol developed by Adobe Systems for streaming audio, video, and data over the Internet, between a Flash player and a server. Avalanche emulates various clients accessing Flash or media files on a real media server. Essentially, a simulated user requests a specific server to transmit a specific file. Avalanche receives the streamed content, controls how the content is streamed through Action list commands, and then terminates the session.

NOTE: The Avalanche RTMP implementation supports Flash Video (FLV) and MP3 streaming file types.

Avalanche supports the following variations of RTMP:

- RTMP (plain) which works on top of TCP and uses port number 1935
- RTMPT which is encapsulated within HTTP requests to traverse firewalls

Defining Client Information

Define the client RTMP parameters that you want to use.

To configure the client:

1. Click the **Client** tab, and then the **Profiles** tab.
2. Select the client profile with which you want to work.
3. In the **RTMP** pane of the **DNS/MM4/SIP/Streaming/RTMP** tab, specify the session teardown timer and indicate whether or not the streaming session is live. For more information, see Client Profiles RTMP Fields in the Help.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

NOTE: The Ramp Down phase time in the load profile must have enough time to allow all the streams to finish playing; otherwise, incomplete sessions are counted as failures. See the Loads Tab in the Help for more information. It is recommended that the Ramp Down phase is as least as long as the time required for the longest streaming file that you are requesting.

To add an Actions list for an RTMP test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter information that simulates retrieving streaming media. Refer to the following syntax and examples.

Syntax

```
RTMP://server_IP_address[:port] <APP = streaming_app> <STREAM =
filename> [<PLAY msecs> | <PAUSE msecs>]*

RTMPT://server_IP_address[:port] <APP = streaming_app> <STREAM =
filename> [<PLAY msecs> | <PAUSE msecs>]*
```

- Server IP address (port is optional)—The address of the server, such as 10.1.79.1, or a URL
- Streaming app—The type of streaming application (such as VOD or LIVE) or the application's full path
- Filename—The name of the streaming file or the file's full path

  NOTE: Avalanche supports Flash Video (FLV) and MP3 streaming file types. However, you do not include the file extensions (`.flv` and `.mp3`) in the syntax.

  - MP3 syntax:

    ```
    <STREAM = mp3:filename>
    ```

  - FLV syntax (no prefix infers FLV):

    ```
    <STREAM = filename>
    ```

- The optional commands include the following:

  NOTE: Avalanche supports positive, integer values for these commands.

  - PLAY-Plays the streaming file. Use this command first in the sequence of commands, and specify the number of milliseconds (msecs) to play the streaming file. You can specify multiple PLAY command sequences in the RTMP action, as indicated by the asterisk (*).
  - PAUSE-Pauses the streaming file for the number of milliseconds (msecs) that you specify.

Examples

- Using an actual streaming server and video on demand (VOD) application, plays the streaming file, `SAMPLE.FLV`, for 10 seconds:

  ```
  RTMP://192.168.1.1 <APP = VOD> <STREAM = SAMPLES/VOD/SAMPLE>
  <PLAY 10000>
  ```

- Using an actual streaming server and live video (LIVE) application, plays the streaming file, `liveShow.flv`, for 10 seconds, pauses for 15 seconds, plays for 20 seconds, pauses for 15 seconds, and plays for 10 seconds:

  ```
  RTMP://192.168.1.1 <APP = LIVE> <STREAM = liveShow> <PLAY
  10000> <PAUSE 15000> <PLAY 20000> <PAUSE 15000> <PLAY 10000>
  ```

- Using an actual streaming server and video on demand (VOD) application, plays the streaming file, `videoSample.mp3`, for 60 seconds:

  ```
  RTMP://10.1.1.1 <APP = vod> <STREAM = mp3:videoSample> <PLAY
  60000>
  ```

- Using an actual streaming server and video on demand (VOD) application, plays the streaming file, `audioSample.mp3`, for 5 seconds, pauses for 4 seconds, and plays for 7 seconds:

  ```
  RTMPT://10.1.1.1 <APP = vod> <STREAM = mp3:audioSample>
  <PLAY 5000> <PAUSE 4000> <PLAY 7000>
  ```

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** button to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing RTSP/RTP Streaming

To complete RTSP/RTP-specific information and run the test:

1. Learn about Avalanche RTSP/RTP testing.
2. Define information for the client.
3. Add content files. (Device test only)
4. Add an Actions list.
5. Define the server profile. (Device test only)
6. Run the test and review results.

About RTSP/RTP Testing

Avalanche emulates various clients that use RTSP/RTP to retrieve streaming media files. Essentially, a simulated user requests a specific server to transmit a specific file. An emulated streaming server can establish a connection with a client, return QuickTime streaming media, Windows Media, or MPEG files, and then terminate the session.

NOTE: Currently, Avalanche supports RTCP only for a QuickTime server.

Defining Client Information

Define the client streaming parameters that you want to use.

1. Click the **Client** tab, and then the **Profiles** tab.
2. Select the client profile with which you want to work.
3. In the **Streaming** pane of the **DNS/MM4/SIP/Streaming** tab, select the **Transport** mode. For more information, see Client Profiles Streaming Protocol Fields in the Help.

Adding Content Files

If you run a test against an actual streaming server, you do not need to add a content file. For use with an emulated streaming server, load the specified file as a content file before you start the test. The available files include the following:

- cawsample_36k_60s_va.mov.caw: A 60-second stream file encoded at a 36K rate that has both video and audio channels.
- cawsample_80k_378s_vo.mov.caw: A 378-second stream file encoded at an 80k rate that has only a video channel.
- cawsample_160k_60s_va.mov.caw: A 60-second stream file encoded at a 160k rate that has both video and audio channels.
- cawsample_250k_60s_va.mov.caw: A 60-second stream file encoded at a 250k rate that has both video and audio channels.

NOTES:

- The sample file names above are annotated, and have different encoding rates and time spans. (The emulated streaming server will NOT work with regular .mov files.)
- If you are doing VQA testing, MPEG-2 TS files are also supported for use with an emulated streaming server (only for VQA analysis).
- Complete the steps in this section only if you are defining a Device test.

To add content files:

1. Click the **Content Files** tab.
2. Click the **Add** button. A directory dialog box appears. By default, it lists files that appear in the default Content file directory. When you first install the Avalanche software, this directory contains the sample streaming files described previously.
3. Select the file, and then click the **Add** button. The file appears in the **Content Files** tab.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom. An RTSP Action list uses two elements: Level and URI. Enter a 1 preceding the URI to signify a top-level retrieval.

NOTE: The Ramp Down phase time in the load profile must have enough time to allow all the streams to finish playing; otherwise, incomplete sessions are counted as failures. See the Loads Tab in the Help for more information. It is recommended that the Ramp Down phase is as least as long as the time required for the longest streaming file that you are requesting.

To add an Actions list for an RTSP/RTP test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter information that simulates retrieving streaming media. Refer to the following syntax and examples.

Syntax - Basic

The following shows the basic syntax for the RTSP action:

```
1 RTSP://server_IP_address[:port]/directory/filename
```

```
1 RTSP://server_IP_address[:port]/filename
```

- Server IP address (port is optional)—The address of the server, such as 10.1.79.1, or a URL

- Directory—The directory where the file is stored. Define this information if you are testing against an actual server, using an Application test.

  NOTE: Most commercially available streaming servers have a default directory for content files. If the file you are requesting is in that directory, you do not need to add the full path to the Action list.

- Filename—The name of the streaming file

Examples - Basic

- Using an actual server

  ```
  1 RTSP://10.1.79.1/movies/thriller.mov
  ```

  ```
  1 RTSP://10.1.79.1/movies/drama.mov
  ```

- Using an emulated server

  ```
  1 RTSP://10.1.79.2/cawsample_36k_60s_va.mov
  ```

IMPORTANT: When using an emulated server, the actual file name (as shown in the Content Files tab) includes the extension .caw, such as cawsample_36k_60s_va.mov.caw. However, when requesting a file in an Actions list, do not include the .caw extension. For example, identify the file as cawsample_36k_60s_va.mov.

Syntax - With Optional Commands

The following syntax shows optional commands that you can use with the RTSP action:

```
1 RTSP://server_IP_address[:port]/filename [PLAY secs {PAUSE secs |
FF secs | RW secs | SEEK secs offset}]*
```

The optional commands include the following:

- PLAY-Plays the streaming file. Use this command first in the sequence of commands, and specify the number of seconds (secs) to wait before performing the other commands. You can specify multiple PLAY command sequences in the RTSP action, as indicated by the asterisk (*).

- PAUSE-Pauses the streaming file for the number of seconds (secs) that you specify.

- FF-Fast forwards the streaming file for the number of seconds (secs) that you specify.

- RW-Rewinds the streaming file for the number of seconds (secs) that you specify.

- SEEK-Positions the streaming file based on the offset parameter. You specify this offset in seconds from the beginning of the stream. You also specify the number of seconds (secs) parameter to indicate the amount of time to continue receiving the stream after the SEEK command.

NOTES:

- Avalanche supports these commands for QuickTime, RealNetworks, and Microsoft, with the exception of MSRTSP over HTTP as the transport. However, Avalanche supports these commands for MSRTSP over TCP and UDP.
- A live streaming server (that is, video not coming from a pre-recorded file) often ignores or generates errors when requested to rewind, pause, fast forward, or seek. In this situation, Avalanche attempts to recover by restarting the stream whenever possible. However, this may or may not be successful. Therefore, it is recommended that you avoid using these four commands against a live stream.
- You must specify the PLAY command when using a live streaming server, except for MSRTSP.
- The PAUSE and SEEK commands are supported on the server side, but FF and RW are not.

Example - With Optional Commands

Play the streaming file for 10 seconds and pause for 3 seconds; play for 5 seconds, position the file at 30 seconds, and continue receiving the stream for 5 seconds; play for 4 seconds and fast forward for 10 seconds:

```
1 RTSP://10.1.79.2/cawsample_36k_60s_va.mov PLAY 10 PAUSE 3 PLAY 5
SEEK 5 30 PLAY 4 FF 10
```

Additional Streaming Settings

The following syntax shows additional streaming settings that you can use with the RTSP action.

- Vendor

  ```
  1 RTSP://10.1.79.2/some_url <VENDOR "Microsoft">
  ```

  The VENDOR streaming setting allows you to specify the vendor that the Avalanche streaming client should emulate. Options are not case sensitive, and include the following:

  - `"Microsoft"`
  - `"Apple"`
  - `"QuickTime"` (currently, handled the same as "Apple")
  - `"Real"`
  - `"SeaChange"`

Avalanche currently supports emulation of RTSP clients for Microsoft, Apple, RealNetworks, and SeaChange vendors, which you specify as described above. When you specify VENDOR, Avalanche does **not** use the file extension at the end of the URL to determine the vendor to emulate. Therefore, you can specify any file extension (or none) in the URL. However, if you do not specify VENDOR, Avalanche uses the file extension at the end of the URL to determine the vendor to emulate (.asf/.wmv for Microsoft, .mov/.mpg for Apple, .rm/.rmvb for RealNetworks, and .mp2t/.mpeg2-ts for SeaChange). If Avalanche encounters an unrecognized extension (or no extension), it generates an error.

NOTES:

- When using a live streaming server, you must specify the VENDOR setting.
- Changing the vendor affects the User-Agent header, as well as the underlying implementation used to emulate the client.

When using a live streaming server, you must specify the VENDOR setting.

- Custom Header

```
1 RTSP://IP_address[:port]/filename.extension <CUSTOMHEADER
overwrite_policy header_name header_value
applicable_methods>
```

The CUSTOMHEADER streaming setting allows you to add (or replace) a header in the RTSP messages generated by Avalanche. The custom header parameters include the following:

  - overwrite_policy-Specifies how to handle the custom header as follows:
    - INSERT-Adds the header that you specify onto all the other headers in the message.
    - REPLACE-Adds the header that you specify onto all the other headers in the message, only if it does not already exist. Otherwise, the new value replaces the old value.

    NOTE: The INSERT option may lead to duplicate headers, whereas the REPLACE option prevents duplication. With REPLACE, if there are already duplicates caused by previous INSERTs, Avalanche will currently replace all of them. However, this functionality may change in the future, for example, replacing only the first header. Therefore, it is recommended that you do not depend on this functionality, and avoid duplicate headers altogether.

- header_name-Specifies the custom header name as follows:
  - string-Any string enclosed in double quotes ("").
  - apply variable-Apply variable name in an Action list, enclosed in double quotes (""). If you use an uninitialized variable, Avalanche generates an error.
- header_value-Specifies the custom header value as follows:
  - string-Any string enclosed in double quotes ("").
  - apply variable-Apply variable name in an Action list, enclosed in double quotes (""). If you use an uninitialized variable, Avalanche generates an error.

IMPORTANT: You must not insert leading or trailing commas, colons, tabs, spaces, etc., into the header names or values. Avalanche automatically inserts those as appropriate.

- applicable_methods-Specifies the methods applicable to the custom header as follows:
  - ALL-Inserts the header into all outgoing messages. This is useful for headers such as User-Agent that need to be included in all messages.
  - NONE-Never sends the header. This is useful for debugging purposes. For example, if you want to prevent a header from being sent, you can simply change the applicable method to NONE, rather than deleting or commenting out the custom header entry in the Actions list.
  - ANNOUNCE, DESCRIBE, GET_PARAMETER, OPTIONS, PAUSE, PLAY, RECORD, REDIRECT, SET_PARAMETER, SETUP, TEARDOWN-Sends the header only for the methods that you specify.

NOTES:

- Avalanche does not currently support generating ANNOUNCE, REDIRECT or RECORD messages.
- Not all vendor implementations support every method.

Examples - Custom Header

```
1 RTSP://10.1.79.2/some_url <CUSTOMHEADER REPLACE "User-
Agent"<APPLY "my_user_agent"> ALL>
```

Sends a User-Agent header that is dynamically generated using a variable (`my_user_agent`). (You can use a forms database to store the header value in the variable.) You use the REPLACE policy to overwrite the default User-Agent header, and APPLY to extract the value from the variable `my_user_agent`. You use the ALL method, since User-Agent headers need to be included in all messages.

NOTE: Changing the user agent string sent does not change the underlying vendor implementation that Avalanche uses for that action. It only allows Avalanche to masquerade as some other type of client. (This also applies to changes via the client profile.)

```
1 RTSP://10.1.79.2/some_url <CUSTOMHEADER REPLACE "Accept"
"some_string_here" SETUP>
```

Replaces the Accept header (with "some_string_here") sent during the SETUP stage only.

```
1 RTSP://10.1.79.2/some_url <CUSTOMHEADER REPLACE
"Transport" "RTP/avp/UDP;port=100" SETUP>
```

Replaces the Transport header (with "RTP/avp/UDP;port=100") sent during the SETUP stage only.

```
1 RTSP://10.1.79.2/some_url <CUSTOMHEADER REPLACE "Scale"
"some_string_here" PLAY >
```

Replaces the Scale header (with "some_string_here") sent during any PLAY phase.

```
1 RTSP://10.1.79.2/some_url <CUSTOMHEADER INSERT <APPLY
"variable_name"> "constant-value" ALL>
```

Adds the header that is dynamically generated using a variable (variable_name) onto all the other headers in the message. (You can use a forms database to store the header name in the variable.) The header value is always "constant-value." The ALL method includes the headers in all messages.

- User Agent (QuickTime only)

  WARNING: This setting has been deprecated as of Avalanche release 2.30. You can still use USERAGENT, however, it will be converted to CUSTOMHEADER, and you will receive a warning message. You should update your Action lists accordingly, since this setting may or may not be supported in the future.

  ```
  1 RTSP://10.1.79.2/cawsample_36k_60s_va.mov <USERAGENT
  "My_User_Agent">
  ```

  The USERAGENT streaming setting allows you to specify the agent on a per-URL basis rather than a per-profile basis. The effect of this setting is the same as that in the Client Profiles Streaming pane. If you specify both a USERAGENT streaming setting and a Streaming User Agent via the client profile, the USERAGENT streaming setting takes precedence. If you do not specify either user agent, Avalanche uses the default based on the vendor implementation (QuickTime, Windows Media, or RealMedia), as determined by the streaming file extension (.mov, .asf, .rm).

NOTE: Changing the user agent string sent does *not* change the underlying vendor implementation that Avalanche uses for that action. It only allows Avalanche to masquerade as some other type of client. (This also applies to changes via the client profile.)

- Accept (QuickTime only)

  WARNING: This setting has been deprecated as of Avalanche release 2.30. You can still use `ACCEPT`, however, it will be converted to `CUSTOMHEADER`, and you will receive a warning message. You should update your Action lists accordingly, since this setting may or may not be supported in the future.

  ```
  1 RTSP://10.1.79.2/cawsample_36k_60s_va.mov <ACCEPT
  "some_string_here">
  ```

  The ACCEPT streaming setting allows you to change the "Accept:" header sent by the client to the server during the SETUP phase of an RTSP transaction.

  NOTE: Overriding the default Accept string does *not* make the client capable of accepting whatever MIME types are specified in the string. It only allows the client to appear to the server as accepting them.

- Transport (QuickTime only)

  WARNING: This setting has been deprecated as of Avalanche release 2.30. You can still use `TRANSPORT`, however, it will be converted to `CUSTOMHEADER`, and you will receive a warning message. You should update your Action lists accordingly, since this setting may or may not be supported in the future.

  ```
  1 RTSP://10.1.79.2/cawsample_36k_60s_va.mov <TRANSPORT
  "some_string_here">
  ```

  The TRANSPORT streaming setting allows you to change the "Transport:" header sent by the client to the server during the SETUP phase of an RTSP transaction.

  NOTE: Overriding the default Transport string does *not* make the client capable of accepting whatever transports are specified in the string. It only allows the client to appear to the server as accepting them.

- Scale (QuickTime only)

  WARNING: This setting has been deprecated as of Avalanche release 2.30. You can still use `SCALE`, however, it will be converted to `CUSTOMHEADER`, and you will receive a warning message. You should update your Action lists accordingly, since this setting may or may not be supported in the future.

  ```
  1 RTSP://10.1.79.2/cawsample_36k_60s_va.mov <SCALE
  "some_string_here">
  ```

The SCALE streaming setting allows you to change the "Scale:" header sent by the client to the server during any PLAY phase of an RTSP transaction. This allows you to play a file at a non-default speed.

IMPORTANT: Using the Scale setting will cause *any* command that overloads the PLAY command (seek, rewind, fast forward, etc.) to also use the same scale, effectively rendering the ones that depend on special scale values useless (specifically, rewind and fast forward). Therefore, you should use this setting only in actions in which you are not using rewind (RW) and fast forward (FF) commands.

- Media Rate (QuickTime only)

  WARNING: This setting has been deprecated as of Avalanche release 2.30. It was previously used to calculate the MDI:DF (media delivery index:delay factor). In the current Avalanche release, the Video Quality Analyzer (VQA) handles this calculation. Use the Client Profiles VQA tab to define VQA parameters.

  ```
  1 RTSP://10.1.79.2/cawsample_36k_60s_va.mov <MEDIARATE integer>
  ```

- Chunks Per Packet (QuickTime only)

  WARNING: This setting has been deprecated as of Avalanche release 2.30. It was previously used to calculate the MDI:MLR (media delivery index:media loss rate). In the current Avalanche release, the Video Quality Analyzer (VQA) handles this calculation. Use the Client Profiles VQA tab to define VQA parameters.

  ```
  1 RTSP://10.1.79.2/cawsample_36k_60s_va.mov <CHUNKSPERPACKET integer>
  ```

You can chain together multiple streaming settings in the same line, any order, as in the following example:

- `1 RTSP://10.1.79.2/cawsample_36k_60s_va.mov PLAY 10 PAUSE 5 SEEK 10 10 <CUSTOMHEADER REPLACE "User-Agent" "My-User-Agent" ALL> <VENDOR "Microsoft">`

NOTES:

- Commands (PLAY, PAUSE, etc.) must be placed together in a sequence, begin with the PLAY command, and precede streaming settings (CUSTOMHEADER, VENDOR, etc.).
- If you specify more than one setting of the same type, the second setting overrides the information set by the first.
- The names of the settings are case-sensitive, and must always be in uppercase.

HTTP to RTSP Redirect

In some cases, you may want to search a server response, such as HTTP, and find the actual RTSP URL that you want to use. You could then extract this RTSP URL, and use it as the next URL in the Action list. This scenario is typical, because of mobile streaming and the use of content engines that hide the true location of the streaming content. The following is an example:

```
1 HTTP://192.168.0.1/this_file_contains_an_rtsp_url.html
ASSIGN VARIABLE <myvar mysearch "RTSP://" "" 0 1 body>
1 RTSP://<APPLY myvar>
```

Where myvar is a variable name, and mysearch is a search criteria file name.

Forms Database

You can use a forms database for entering the name of the streaming file, as in the following example:

```
ASSIGN VARIABLE <myvar myformsdb.$1>
1 RTSP://<APPLY myvar>
```

Where myvar is a variable name, and myformsdb is the name of a forms database.

An example forms database is as follows:

```
10.1.79.2/cawsample_36k_60s_va.mov
10.1.79.2/cawsample_80k_378s_vo.mov
10.1.79.2/cawsample_160k_60s_va.mov
```

Defining the Server Profile

Complete this section only if you are defining a Device test. Use the **Server Profiles** tab to configure information for a streaming server.

To define a server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **Streaming** as the server type.
4. Enter the port where the server resides. The default port is 554.
5. Select emulation settings.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing SIP

To complete SIP information and run the test:

1. Learn about SIP testing.
2. Define information for the client.
3. Add an Actions list.
4. Add a content file (if you want to use pre-recorded Wave files).
5. Define information for the server. (Device test only)
6. Run the test and review results.

About SIP Testing

SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls. RTP data streams carry the voice data.

Avalanche performs the role of multiple clients. It initiates a SIP session by using the corresponding SIP message exchange, sends pre-recorded Wave files, receives RTP data, and terminates the session by using the corresponding SIP message exchange. Avalanche supports SIP over UDP and TCP transport modes. Avalanche processes any session update messages it receives, but does not initiate session update messages. Avalanche can generate calls to real SIP devices, such as IP Phones, and with a simulated server, can transmit traffic between gateways, SIP firewalls and SIP proxy servers. For more information about the role of Avalanche, see Configuring SIP in the Help.

You can use a simulated server to perform the role of multiple SIP user agents, terminating the call. The server accepts incoming calls, and can echo the RTP stream, if any, or reply back with its own stream, if you select a content file in the Server Profile. It processes any session update messages it receives, but does not initiate session update messages. The server can accept calls from real SIP devices, such as gateways, SIP firewalls, SIP phones, and SIP proxy servers. For more information about the role of the simulated server and detailed Server Profile field definitions, see Server Profiles Tab SIP in the Help. The Help also describes what Avalanche supports with respect to the client and server and their use of a wave file, with or without codecs.

Defining Client Information

Define the client SIP parameters that you want to use.

To configure the client:

1. Click the **Client** tab, and then the **Profiles** tab.
2. Select the client profile with which you want to work.

3. In the SIP pane of the **DNS/MM4/SIP/Streaming** tab, complete entries to define the local SIP port, the port number on which the first RTP listeners will be created, and the number of RTP channels that will be created. Enter a port number value that is even and in the acceptable port range. The port value is incremented by two for each additional RTP channel. It is recommended that you set the number of channels to be greater than the expected maximum number of simultaneous calls to provide more realistic traffic. For more information, see Client Profiles SIP Fields in the Help.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

To add an Actions list for a SIP test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter information that simulates SIP. Refer to the following syntax and examples.

Default Syntax

```
sip://192.168.44.1 LOCALHOST="caw.com" TRANSPORT=UDP
```

- LOCALHOST—The host name used in the From header.
- TRANSPORT—The transport mode used in the session, in this case UDP.

NOTE: The default syntax does not specify a Wave file. Without a Wave file, Avalanche sends an on-hold Session Description Protocol (SDP); Avalanche does not send RTP data during the session. After session setup, Avalanche starts sending the stream from the Wave file.

In addition, because the default syntax does not specify call length parameters in the URL, Avalanche defaults to a call length randomly chosen between a minimum of 10000 ms (10 seconds) and a maximum of 30000 ms (30 seconds). As soon as the call is accepted, Avalanche calculates the call time, and terminates the call after this time.

Syntax

```
sip://ip:[port]

LOCALHOST = STRING

[REMOTEUSER = STRING]

[REMOTEHOST = STRING]

[CALLLENGTH_MIN = INTEGER]

[CALLLENGTH_MAX = INTEGER]
```

```
[TRANSPORT = (UDP | TCP)]

[LOCALUSER = STRING]

[WAVE = STRING]

[MAXFORWARDS = INTEGER]

[REQUESTURI = STRING]

[SIPHEADER = STRING]

[REGISTRAR = STRING] (for UDP only)

[REGISTRARPORT = INTEGER] (for UDP only)

[REGEXPIRATION = INTEGER] (for UDP only)

[RTP_INTERVAL = INTEGER]
```

- LOCALHOST-The host name used in the From header.
- REMOTEUSER-The username used in the To header. If no request URI is configured, this name will be used in the Request URI.
- REMOTEHOST-The host name used as host in the To header. If no REQUESTURI is configured, this name will be used in the Request URI. By default, the destination host of the Action is used.
- CALLLENGTH_MIN, CALLLENGTH_MAX-The number of milliseconds used as the interval (between MIN and MAX) for the random selection of the call time. The default MIN is 10000 (10 seconds) and the default MAX is 30000 (30 seconds).
- TRANSPORT-The transport mode used in the session: UDP (default) or TCP.
- LOCALUSER-The username used in the From and local Contact header.
- WAVE-The name of the Wave file. If present, the client will send the RTP data for the session from this file.
- MAXFORWARDS-The value of the Max-Forwards header. The default is 70.
- REQUESTURI-If present, used as the Request URI in the initial request. Otherwise, the default sequence as described for REMOTEHOST is used.
- SIPHEADER-"Header name: value" adds an additional header to the first INVITE request. One Action can have several SIPHEADER parameters.
- REGISTRAR (for UDP only)-The IP address of the Registrar server to which to register before the client makes a call. (This action is for registration purposes only. It does not initiate a call.)
- REGISTRARPORT (for UDP only)-The port number of the Registrar server. You can choose to change the default port number (5060), but make sure that the Registrar is listening on that port.

- REGEXPIRATION (for UDP only)-The time, in seconds, after which registration expires, and the client should register again.
- RTP_INTERVAL-The number of milliseconds to use as the interval for sending RTP packets. This value overrides the default interval of the RTP codec. The RTP_INTERVAL controls only the interval of one-way RTP traffic, so you must not use the Server Profiles SIP RTP configuration (Use Content File and Codecs Configuration fields). The RTP_INTERVAL must be greater than 20 milliseconds.

NOTE: The host and port (default 5060) in the URI string are used as the destination for the first request in the call for UDP, and all requests in the call for TCP. The destination of requests inside the call for UDP depends on the received Contact header and Route set.

To add a predefined route set for the first request in the call for UDP, and all requests for TCP, the REQUESTURI must be configured to have the correct value according to RFC 3261, and the set of Route headers must be added by using the SIPHEADER parameter of the Action.

Examples

- A minimal SIP URL:

  ```
  sip://192.168.42.11 LOCALHOST="somewebsite.com"
  ```

  The LOCALHOST parameter is used as the host part of the From header in SIP messages. This is the only mandatory parameter.

  If the remote port differs from 5060, it should be specified. In the following example, 5067 is used:

  ```
  sip://192.168.42.11:5067 LOCALHOST="somewebsite.com"
  ```

  NOTE: If the IP address or port is incorrect, the session in considered unsuccessful after 32 seconds for UDP. This is a protocol-specific feature.

- WAVE parameter:

  ```
  sip://192.168.42.11 LOCALHOST="somewebsite.com" WAVE
  ="welcome.wav"
  ```

  The WAVE parameter specifies a Wave file name to be used. If this parameter is not specified, the session will be a signaling-only session without RTP.

  IMPORTANT: The Wave file that you specify must be an uncompressed, linear (PCM), 8000 Hz, 16-bit, mono u-Law Wave file. (You can use a standard Windows sound recorder to create the file.)

  NOTE: You must upload the corresponding Wave file using the Content Files tab.

Avalanche encodes your Wave file into the codec(s) that you specify. The following codecs are supported:

- G.711u
- G.711a
- G.723.1
- G.726-16
- G.726-24
- G.726-32
- G.726-40
- G.728
- G.729AB

To use specific codecs, you add a codec list to your URL using the following syntax:

AUDIO={codec1,codec2,...,codecN}

Each codec in the list must be one of the supported codecs.

Example:

sip://192.168.1.11 LOCALHOST="caw.com" TRANSPORT=UDP WAVE="123.wav" AUDIO={G.711u}

NOTES:

- If you specify a Wave file without a codec list, Avalanche uses G.711u, and logs a warning message.
- Avalanche does not enforce codec order. For example, if Avalanche sends {G.711u, G.729AB} in the INVITE message, and receives {G.729AB, G.711u} in the OK message, G.729AB will be used.
- If the file length is greater than the call length, the file is truncated. If the file length is less than the call length, the file is replayed.

- TRANSPORT parameter:

  The TRANSPORT parameter can be either UDP or TCP. If this parameter is omitted, UDP will be used as the default transport mode.

  NOTE: This is the only method to specify the transport mode on the client side.

UDP examples:

```
sip://192.168.42.11:5067 LOCALHOST="somewebsite.com"
```

```
sip://192.168.42.11:5067 LOCALHOST="somewebsite.com"
TRANSPORT=UDP
```

TCP example:

```
sip://192.168.42.11:5067 LOCALHOST="somewebsite.com"
TRANSPORT=TCP
```

- CALLLENGTH_MIN and CALLLENGTH_MAX parameters:

```
sip://192.168.42.11:5067 LOCALHOST="somewebsite.com"
CALLLENGTH_MIN = 20000 CALLLENGTH_MAX = 30000
```

The call length is randomly selected from the interval between CALLLENGTH_MIN and CALLLENGTH_MAX. The values are in milliseconds. The call length in this example will be between 20 and 30 seconds. The CALLLENGTH_MAX must be less than the call timeout value on the server side.

- RTP interval of 1 second:

```
sip://192.168.42.11:5067 LOCALHOST="somewebsite.com"
CALLLENGTH_MIN = 20000 CALLLENGTH_MAX = 30000
RTP_INTERVAL=1000
```

- RTP interval of 500 milliseconds:

```
sip://192.168.42.11:5067 LOCALHOST="somewebsite.com"
CALLLENGTH_MIN = 20000 CALLLENGTH_MAX = 30000 WAVE="123.wav"
AUDIO={G.711u} RTP_INTERVAL=500
```

- REMOTEUSER parameter with a phone number:

```
sip://192.168.42.11:5067 LOCALHOST="somewebsite.com"
REMOTEUSER="18007747368"
```

The REMOTEUSER parameter specifies the username or dial number of the remote site. It is not mandatory when the simulated server accepts the call, but may be required by a real device.

- SIP over UDP, 60-second call with only signalization:

```
sip://10.10.10.10 LOCALHOST="somewebsite.com"
LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
REMOTEUSER="18007747368" TRANSPORT=UDP CALLLENGTH_MIN=60000
CALLLENGTH_MAX=60000 AUDIO={G.711a}
```

- SIP over UDP, 60-second call with signalization and RTP using G.711a codec:

```
sip://10.10.10.10 LOCALHOST="somewebsite.com"
LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
REMOTEUSER="18007747368" TRANSPORT=UDP CALLLENGTH_MIN=60000
CALLLENGTH_MAX=60000 WAVE="voice_10_sec.wav" AUDIO={G.711a}
```

- SIP over UDP, 60-second call with signalization and RTP using G.711u codec:

```
sip://10.10.10.10 LOCALHOST="somewebsite.com"
LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
REMOTEUSER="18007747368" TRANSPORT=UDP CALLLENGTH_MIN=60000
CALLLENGTH_MAX=60000 WAVE="voice_10_sec.wav" AUDIO={G.711u}
```

- SIP over UDP, 60-second call with signalization and RTP using G.726 codec at 32 Kbps:

```
sip://10.10.10.10 LOCALHOST="somewebsite.com"
LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
REMOTEUSER="18007747368" TRANSPORT=UDP CALLLENGTH_MIN=60000
CALLLENGTH_MAX=60000 WAVE="voice_10_sec.wav" AUDIO={G.726-32}
```

- SIP over UDP, 60-second call with signalization and RTP using G.728 codec:

```
sip://10.10.10.10 LOCALHOST="somewebsite.com"
LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
REMOTEUSER="18007747368" TRANSPORT=UDP CALLLENGTH_MIN=60000
CALLLENGTH_MAX=60000 WAVE="voice_10_sec.wav" AUDIO={G.728}
```

- SIP over UDP, 60-second call with signalization and RTP using G.729AB codec:

```
sip://10.10.10.10 LOCALHOST="somewebsite.com"
LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
REMOTEUSER="18007747368" TRANSPORT=UDP CALLLENGTH_MIN=60000
CALLLENGTH_MAX=60000 WAVE="voice_10_sec.wav" AUDIO={G.729AB}
```

- SIP over TCP, 60-second call with only signalization:

```
sip://10.10.10.10 LOCALHOST="somewebsite.com"
LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
REMOTEUSER="18007747368" TRANSPORT=TCP CALLLENGTH_MIN=60000
CALLLENGTH_MAX=60000 AUDIO={G.711a}
```

- SIP over TCP, 60-second call with signalization and RTP using G.711a codec:

```
sip://10.10.10.10 LOCALHOST="somewebsite.com"
LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
REMOTEUSER="18007747368" TRANSPORT=TCP CALLLENGTH_MIN=60000
CALLLENGTH_MAX=60000 WAVE="voice_10_sec.wav" AUDIO={G.711a}
```

- SIP over TCP, 60-second call with signalization and RTP using G.711u codec:

```
sip://10.10.10.10 LOCALHOST="somewebsite.com"
LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
REMOTEUSER="18007747368" TRANSPORT=TCP CALLLENGTH_MIN=60000
CALLLENGTH_MAX=60000 WAVE="voice_10_sec.wav" AUDIO={G.711u}
```

- SIP over TCP, 60-second call with signalization and RTP using G.726 codec at 32 Kbps:

```
sip://10.10.10.10 LOCALHOST="somewebsite.com"
LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
REMOTEUSER="18007747368" TRANSPORT=TCP CALLLENGTH_MIN=60000
CALLLENGTH_MAX=60000 WAVE="voice_10_sec.wav" AUDIO={G.726-32}
```

- SIP over TCP, 60-second call with signalization and RTP using G.728 codec:

  ```
  sip://10.10.10.10 LOCALHOST="somewebsite.com"
  LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
  REMOTEUSER="18007747368" TRANSPORT=TCP CALLLENGTH_MIN=60000
  CALLLENGTH_MAX=60000 WAVE="voice_10_sec.wav" AUDIO={G.728}
  ```

- SIP over TCP, 60-second call with signalization and RTP using G.729AB codec:

  ```
  sip://10.10.10.10 LOCALHOST="somewebsite.com"
  LOCALUSER="lchabenet" REMOTEHOST="somewebsite.com"
  REMOTEUSER="18007747368" TRANSPORT=TCP CALLLENGTH_MIN=60000
  CALLLENGTH_MAX=60000 WAVE="voice_10_sec.wav" AUDIO={G.729AB}
  ```

- REGISTRAR parameters (for UDP only):

  ```
  sip://192.168.41.10 LOCALHOST="192.168.41.21" TRANSPORT=UDP
  REMOTEUSER="1000" LOCALUSER="5001" REGISTRAR="192.168.41.10"
  REGISTRARPORT=5061 REGEXPIRATION=180
  ```

  SIP clients can register themselves to a Registrar Server before making a call. The REGISTRAR parameter specifies the IP address of the Registrar server to which to send the registration request. You can choose to change the default port number (5060) using the REGISTRARPORT parameter. Use the REGEXPIRATION parameter to specify the time, in seconds, after which registration expires, to register again.

Adding a Content File

If you configured the Actions list to use pre-recorded Wave files, upload them by using the Content Files tab before you run the SIP test.

To add content files:

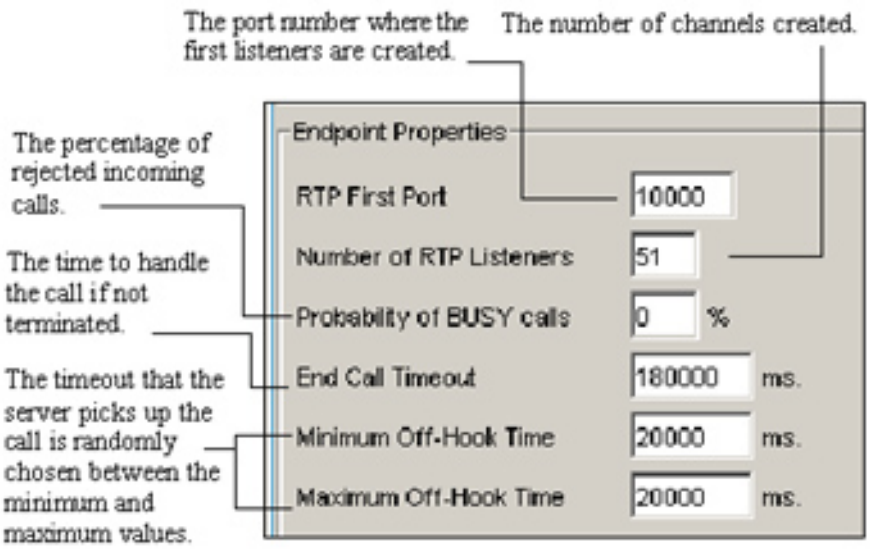
1. Click the **Content Files** tab.
2. Click the **Add** button. A directory dialog box appears. By default, it lists files that appear in the default Content file directory. When you first install the Avalanche software, this directory contains sample streaming files. You can also store your own content files in this directory.
3. If you stored your files in the default directory, skip to the next step. Otherwise, navigate to the location that contains the Wave file that you want to use.
4. Select the file, and then click the **Add** button. The file appears in the **Content Files** tab.

Defining the Server Profile

Complete this section only if you are defining a Device test. Use the **Server Profiles** tab to configure information for a SIP server.

To define a server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **SIPTCP** or **SIPUDP** as the server type.
4. Enter information in the Endpoint Properties fields.



NOTE: Before the Off-Hook time expires, the ringing signal is sent to the caller.

5. Select the **Use Content File** checkbox if you want to use a streaming file, and then select the name of the streaming file that you want to load as a content file from the drop-down menu.
6. For Codecs Configuration, click the **Edit** button to display a window in which you can select codec(s) to use to encode your Wave file.
7. For UDP only, you can select the **Register Enabled** checkbox if you want to send a registration request from the server to a Registrar server. Enter registrar information in the associated fields.

For more information about SIP fields, see Server Profiles Tab SIP in the Help.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

NOTE: For information about SIP statistical data see the client SIP Statistics and server SIP Statistics in the Help.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing SIPNG

IMPORTANT:

- Avalanche provides a legacy SIP implementation, denoted by "SIP" in the Client Profile, and "SIPTCP," "SIPUDP," and "SIPProxyUDP" in the Server Profile, which may not be supported in the future. It is recommended that you transition to using the latest SIP functionality provided by SIPNG (Session Initiation Protocol *Next Generation*), which provides more complete coverage of the SIP protocol.
- A SIPNG configuration can run against a legacy SIP configuration.

NOTE: SIPNG provides the following enhancements over the legacy SIP implementation:

- Improved SIP over IPv6
- More powerful SIP proxy
- More complete SIP statistics
- Proxy over TCP
- Registering with authentication
- Real SDP negotiation
- Sending different RTP streams by both client and server
- Initiating BYE transactions by both client and server
- Encoding form support for long and short header field names
- Improved stability

To complete SIP information and run the test:

1. Learn about SIP testing.
2. Define information for the client.
3. Define phonebook information for an Actions list.
4. Define information for the server. (Device test only)
5. Run the test and review results.

About SIP Testing

SIP (Session Initiation Protocol) is a signaling protocol, widely used for setting up and tearing down multimedia communication sessions, such as voice and video calls over the Internet. RTP data streams are used to carry the voice data.

Avalanche performs the role of multiple clients. It initiates a SIP session by using the corresponding SIP message exchange, sends pre-recorded content files, receives RTP data, and terminates the session by using the corresponding SIP message exchange. Avalanche supports SIP over UDP and TCP transport modes. It can generate calls to real SIP devices, such as IP Phones, and with a simulated server, can transmit traffic between gateways, SIP firewalls and SIP proxy servers.

You can use a simulated SIPNG Endpoint server to perform the role of multiple SIP user agents, terminating the call. It accepts incoming calls, and echoes the RTP stream, if any. The simulated server can accept calls from real SIP devices, such as gateways, SIP firewalls, SIP phones, and SIP proxy servers. For more information about the role of the simulated SIPNG Endpoint server and detailed Server Profile field definitions, see Server Profiles Tab SIPNG_Endpoint in the Help.

You can also use a simulated SIPNG Proxy server to route SIP requests and reply to the next hop, which can be another proxy or an endpoint. For more information about the role of the simulated SIPNG Proxy server and detailed Server Profile field definitions, see Server Profiles Tab SIPNG_Proxy in the Help.

SIP Request Messages

SIPNG supports the following SIP request messages:

- INVITE - Initiates the session. Includes the client session description in the message body. Uses re-INVITEs to change the session state.
- ACK - Sends the final confirmation from the client that initiated the session (INVITE originator).
- BYE - Terminates the session. Can be initiated by the calling or called party.
- REGISTER - Clients use to register location information with SIP servers.

SIP Response Messages

SIPNG supports the following SIP response messages:

- 1xx: Informational - Provisional (ringing, trying, forwarded, etc.)
- 2xx: Success - Final (OK)
- 3xx: Redirection - Final (multiple choices, moved, use proxy)
- 4xx: Client error - Final (bad request, not found, forbidden)
- 5xx: Server error - Final (server error, bad gateway, service unavailable)
- 6xx: Global failure - Final (decline, does not exist)

Content File Support

You can specify a content file to be used as audio content in the Media Properties pane of the Client Profiles SIPNG tab and the SIPNG_Endpoint Server Profiles tab. Avalanche encodes your content file into the codec(s) that you specify under Media Properties. The following codecs are supported:

- G.711u
- G.711a
- G.723.1
- G.726-16
- G.726-24
- G.726-32
- G.726-40
- G.728
- G.729AB

IMPORTANT: The content file that you specify must be an uncompressed, linear (PCM), 8000 Hz, 16-bit, mono u-Law file. (You can use a standard Windows sound recorder to create the file.)

NOTES:

- If you specify a content file without a codec list, Avalanche generates an error message.
- Avalanche does not enforce codec order. For example, if Avalanche sends {G.711u, G.729AB} in the INVITE message, and receives {G.729AB, G.711u} in the OK message, G.729AB will be used.
- If the file length is greater than the call length, the file is truncated. If the file length is less than the call length, the file is replayed.

Related RFCs

Consult the following documents for industry-standard information related to SIPNG:

RFC 3261 - SIP: Session Initiation Protocol

RFC 2617 - HTTP Authentication: Basic and Digest Access Authentication

RFC 2327 - SDP: Session Description Protocol

RFC 3550 - RTP: A Transport Protocol for Real-Time Applications (also RTCP)

Defining Client Information

Define the client SIPNG parameters that you want to use.

To configure the client:

1. Click the **Client** tab, and then the **Profiles** tab.
2. Select the client profile with which you want to work.
3. Click the **SIPNG** tab, and complete entries to define basic SIP properties, registration configuration, and media properties. For more information, see Client Profiles SIPNG Fields in the Help.

Defining Phonebook Information for Actions List

Define the SIPNG phonebook information for use in an Actions list. The Actions list identifies the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

To define phonebook information:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new Actions list, or select an existing list from the drop-down menu.
3. Click the **Phonebook** tab.
4. Click the **New** button  to create a new phonebook.
5. Enter phonebook information, such as transport type and call length. For more information, see Phonebook Fields in the Help.
6. Select and copy the text in the Example Action field, and then paste it into your Actions list.

Defining the Server Profile

NOTE: Complete this section only if you are defining a Device test.

Use the **Server Profiles** tab to configure information for an emulated SIPNG server, which can respond to SIPNG requests generated by Avalanche. You use a SIPNG_Endpoint or SIPNG_Proxy server, depending on your testing scenario.

To define a SIPNG server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **SIPNG_Endpoint** or **SIPNG_Proxy** as the server type.
4. Enter the port where the server resides. The default port is 5060.

5. For a SIPNG_Endpoint server, complete entries to define endpoint properties, registration configuration, and media properties. For more information about SIPNG_Endpoint fields, see Server Profiles Tab SIPNG_Endpoint in the Help.
6. For a SIPNG_Proxy server, complete entries to define SIP proxy properties, registration configuration, and the routing table. For more information about SIPNG_Proxy fields, see Server Profiles Tab SIPNG_Proxy in the Help.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The **Monitor** tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing SMTP

To complete SMTP-specific information and run the test:

1. Learn about Avalanche SMTP testing.
2. Add an Actions list.
3. Add content files. (if you are attaching files)
4. Define the server profile. (Device test only)
5. Run the test and review results.

About SMTP Testing

Avalanche simulates a client sending various mail messages from an SMTP server. The Action list describes the list of transactions and controls their sequence of execution. By default, SMTP uses port 25 for transport, and levels are not relevant to the SMTP protocol.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

NOTE: You can alternatively use ESMTP actions instead of SMTP actions in your Action list. For more information, see ESMTP Actions.

With an SMTP Action list you define:

- Envelope information: From, To, and Subject (optional).
- Message information: The content of the message. You can use several ways to provide the message.

To create an Actions list for an SMTP test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter actions that simulate SMTP. Refer to the following syntax and examples.

Syntax

The basic syntax is:

```
smtp://IP_Address FROM=<user@host> TO=<user@host>
SUBJECT=<"my_subject"> DATA=<content of message>
```

See the following syntax for details about each component, followed by an explanation of each component and examples of use.

Specifying the sender (FROM)

```
FROM=<user@host>

or

FROM=<database^#>
```

NOTE: The above syntax specifies a forms database, where *database* is the filename and # is the column number.

Specifying the recipient (TO)

```
TO=<user@host>

or

TO=<database^#>
```

You can specify the recipient as a list of email addresses in comma-separated value (CSV) format. For example,

```
TO=<user@host,user@host>
```

or a combination of email addresses and addresses in a forms database

```
TO=<user@host,database^#>
```

Specifying the subject (SUBJECT) (optional)

```
SUBJECT=<databasefile^#>

or

SUBJECT=<"this_is_my_subject">
```

Sending data

```
DATA=<content of message>
```

Sending a body file

```
SMTP_BODY_FILE=<"filename" "content-type">
```

Sending a raw message

```
SMTP_RAW_MESSAGE=<"filename">
```

Using a forms database with fixed content

```
DATA=<FIXED,300>
```

- IP address-Identifies the IP address of the test server.
- FROM-Specifies the sender information, using its email address and domain name.
- TO-Specifies the recipient.

- SUBJECT (optional)-Specifies the subject information. You can include a subject or reference a forms database file. If you don't specify SUBJECT, Commander uses a random subject during the test. You specify the subject within double quotes. Keywords are not case-sensitive.

- DATA (optional if using a body or attachment)-Specifies the content of the email message sent to the SMTP server. With DATA, you can use the following parameters:

    - FIXED-Followed by an integer, such as FIXED 300, specifies the length of the generated string used as the message body. This length does not include the message header.

    - RANDOM=UNIFORM- Followed by two integers, such as RANDOM=UNIFORM 300 600, specifies a randomly generated string of random length to be used as the message body. The length is uniformly distributed between a minimum and a maximum set of values.

- SMTP_BODY_FILE (optional)-As an alternative to DATA, you can specify the filename and content-type of the body that should be included as the Multipurpose Internet Mail Extensions (MIME) that represents the body of the message. Format the contents of the file that you specify for the SMTP_BODY_FILE as ASCII text, since it is used as is.

- SMTP_ATTACH_FILE (optional)-Specifies the filename and content-type of an attachment to include as an attachment in the MIME Message. The contents of the file that you specify for an SMTP_ATTACH_FILE is base64-encoded. You can specify any text for content-type; it is appended to the Content-Type header.

- SMTP_RAW_MESSAGE (optional)-Specifies the raw data's filename of the entire MIME message included after the standard headers. The contents of the file that you specify for an SMTP_RAW_MESSAGE can be in any format, and is used as is. You specify the filename within double quotes. If you use SMTP_RAW_MESSAGE, you cannot use DATA, SMTP_BODY_FILE, or SMTP_ATTACH_FILE.

  IMPORTANT: You must add the files that you specify as content files in the Content Files Tab. See Adding Content Files later in this topic.

- Actions (optional)-You can specify any number of the following to be sent: RSET, EHLO (using ESMTP), SOML, SEND, SAML, VRFY, EXPN, HELP, TURN, ETRN, and XXXX.

  NOTE: If you use an emulated server, there is no decode logic to act on these Actions, simply an acknowledgement that the Action was sent.

- REPEAT (optional)-Specifies the number of times to send the message within the same SMTP session. The default value is 1.

- DATABASE-Identifies the filename and column number (#) if you use a forms database.

Examples

- IP Address Examples:

  An IPv4 address

  ```
  smtp://192.168.42.11 FROM=<a@somewebsitea.com>
  TO=<b@somewebsiteb.com> DATA=<FIXED,300>
  ```

  An IPv6 address

  ```
  smtp://[2107::0200:FF:FE00:8201] FROM=<a@somewebsitea.com>
  TO=<b@somewebsiteb.com> DATA=<FIXED,300>
  ```

- Simulated content:

  Fixed 300 bytes

  ```
  smtp://192.168.42.14 FROM=<user1@somewebsite.com>
  TO=<user2@somewebsite.com> SUBJECT=<"TEST"> DATA=<FIXED, 300>
  ```

  Random 300-600 bytes

  ```
  smtp://192.168.42.14 FROM=<user1@somewebsite.com>
  TO=<user2@somewebsite.com> SUBJECT=<"TEST">
  DATA=<RANDOM=UNIFORM 300 600>
  ```

- Content in body file:

  ```
  smtp://192.168.42.14 FROM=<user1@somewebsite.com>
  TO=<user2@somewebsite.com> SUBJECT=<"TEST">
  SMTP_BODY_FILE=<"mymsg" "plain/text">
  ```

- Multiple recipients, fixed data content, SOML to be sent, and the message repeated three times:

  ```
  smtp://192.168.42.11 FROM=<a@somewebsitea.com>
  to=<b@somewebsiteb.com,c@somewebsitec.com> DATA=<FIXED,300>
  SOML REPEAT=3
  ```

- Fixed content with raw virus file:

  ```
  smtp://192.168.42.14 FROM=<user1@somewebsite.com>
  TO=<user2@somewebsite.com> SUBJECT=<"I love you">
  SMTP_RAW_MESSAGE=<"iloveyou.exe">
  ```

- Raw data file as the entire MIME message when the file does not need to be encoded:

  ```
  smtp://192.168.42.11 FROM=<a@somewebsitea.com>
  to=<b@somewebsiteb.com> subject=<"my_subject">
  SMTP_RAW_MESSAGE=<"themsg">
  ```

- File as the message body and other files attached:

  ```
  smtp://192.168.42.11 FROM=<a@somewebsitea.com>
  to=<b@somewebsiteb.com>

  SMTP_BODY_FILE=<"mymsg" "plain/text">

  SMTP_ATTACH_FILE=<"wordDoc" "application/msword">

  SMTP_ATTACH_FILE=<"PDFDoc" "application/pdf">

  SMTP_ATTACH_FILE=<"ExcelDoc" "application/vnd.ms-excel">
  ```

NOTE: If you use SMTP attachments, consider that large attachment files create extensive network overhead. Some error conditions may occur, such as throttling due to low memory, if the total size of attachments is greater than the network bandwidth.

- Character set option (entire specified string appears in the Content-Type header)

```
SMTP_BODY_FILE=<"mymsg" "plain/text; charset=US-ASCII">
```

- Fixed content with forms database:

```
smtp://192.168.42.11 from=<SMTPdb^1> to=<SMTPdb^2>
subject=<SMTPdb^3> DATA=<FIXED,300>
```

The following forms database could support the previous forms database action:

```
asmith@somewebsite.com, eyee@somewebsite.com, Hi

bbrown@somewebsite.com, jsant@somewebsite.com, Hi

cjones@somewebsite.com, rmoe@somewebsite.com, Hi

dkline@somewebsite.com, ejay@somewebsite.com, Hi
```

Adding Content Files

If you want to use SMTP_BODY_FILE, SMTP_ATTACH_FILE, or SMTP_RAW_MESSAGE, you must add the files that you specify as content files in the Content Files Tab before running the test.

To add content files:

1. Click the **Content Files** tab.
2. Click the **Add** button. A directory dialog box appears. By default, it lists files that appear in the default content file directory. When you first install the Avalanche software, this directory contains sample streaming files for use with a streaming test. You can also store your own content files in this directory.
3. Select the file, and then click the **Add** button. The file appears in the **Content Files** tab.

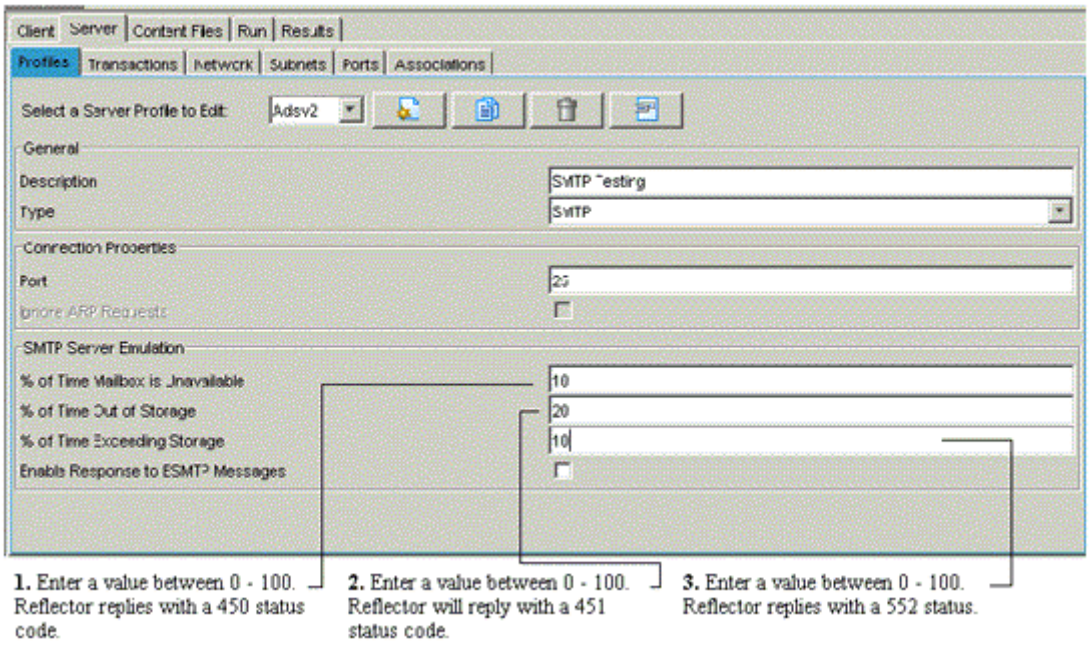
Defining the Server Profile

Complete this section only if you are defining a Device test.

To define an SMTP server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **SMTP** as the server type.
4. Enter the port where the server resides. By default, SMTP uses port 25 for transport.

5. Set SMTP server emulation settings. By default they are 0.



Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

NOTE: The statistical data for SMTP tests are described in SMTP Statistics in the Help.

Testing Telnet

To complete Telnet-specific information and run the test:

1. Learn about Avalanche Telnet testing.
2. Define Telnet commands on the Client Profile tab.
3. Define Telnet commands on the Server Profile tab. (Device test only)
4. Create a forms database. (If using a forms database to supply client data.)
5. Add an Actions list that defines which profile (and forms database if applicable) to use.
6. Run the test and review results.

About Telnet Testing

7. Avalanche can emulate a Telnet client and send commands to a Telnet server or emulated server. You specify the commands to be sent in the Telnet tab on the Client Actions tab.
8. For an emulated server, use the Server Profiles tab to expect and respond to the commands from Avalanche. For a real Telnet server, set up the Avalanche Telnet profile with the exact response the Telnet server provides to the command.

After defining your profiles, create an Actions list to tell the test which profile to use to obtain the Telnet commands. (You can also use an Actions list to refer to a forms database that defines Telnet commands.)

NOTE: Telnet is a send and expect protocol. You must configure the flow of the session in both the client and the server consistently. That which is sent by the client is expected by the server, and vice-versa. The server expects an exact match, character for character. However, the client expects a match only with the first character.

Defining Client Telnet Commands

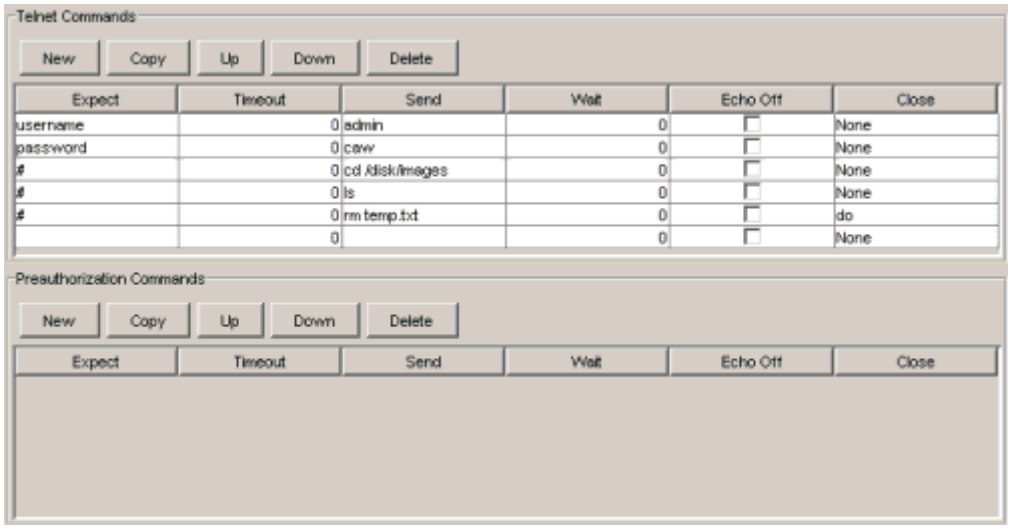
Use the Telnet tab on the Client Actions tab to define client Telnet commands.

To define a Telnet profile:

1. Click the **Client Actions** tab, and then the **Telnet** tab.
2. In the left pane of the **Telnet** tab on the **Client Actions** tab, select the Telnet profile that you want to use, or click the **New** button  and enter a name to create a new profile. You will reference the name of the Telnet profile when you create an Action list to test Telnet.
3. Select if you want to enable option negotiation or send end of line sequences for a send.
4. Click the **New** or **Copy** Telnet Command buttons to add or copy a command.

5. Within the Telnet table, define the commands and values that you want to use. These should be the same sequence of commands that you would use if performing a live Telnet session. For example, you might expect a username prompt, and then send a username; expect a password prompt, and then send a password; expect a command prompt, and then send a command such as ls, and so on. You can enter specific values or reference a column of data in a forms database for values. (For information about forms databases, see "Creating a Forms Database for a Telnet Test.")
6. Use the Preauthorization Command buttons to add or copy a command. Define Preauthorization commands only for a network topology where the Telnet session will go through a double-authentication procedure. Within the Preauthorization table, define the commands that you want to use.

The following shows an example where Telnet commands are defined on the Client Actions tab.



Defining Server Telnet Commands

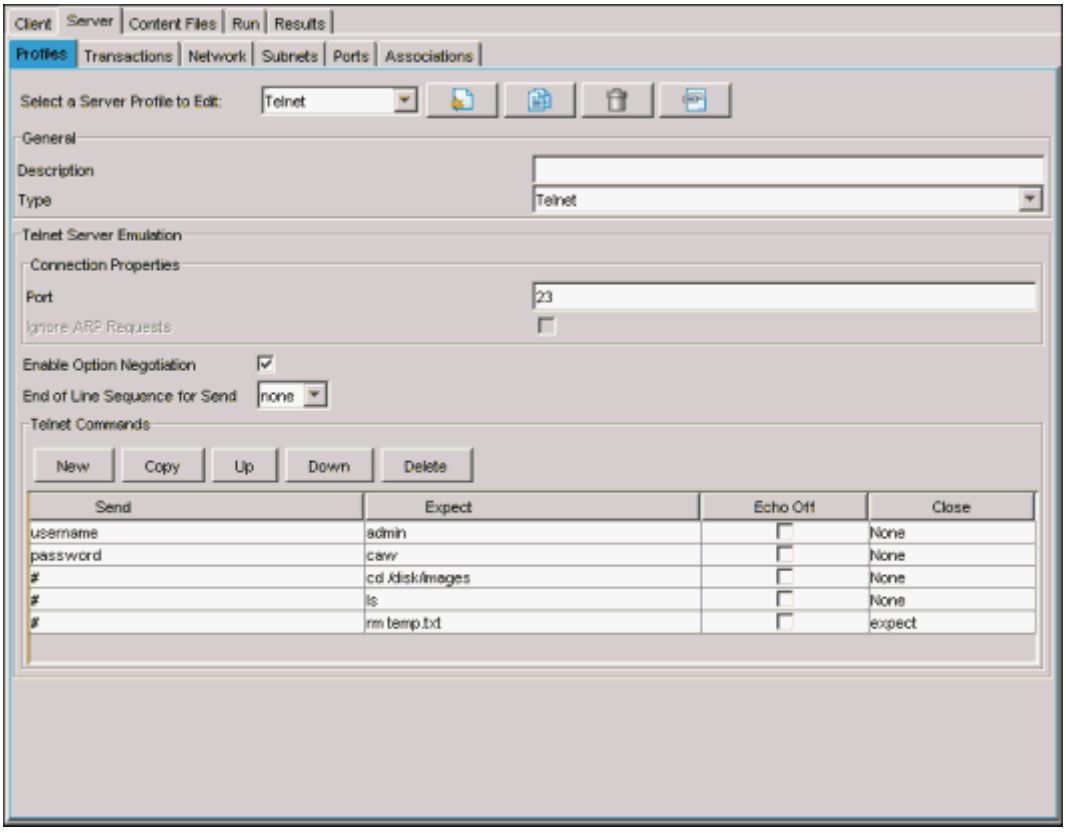
Complete this section only if you are defining a Device test. Use the **Server Profiles** tab to define Telnet commands that the server should expect to receive from the client and send to the client.

To define a server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **Telnet** as the server type.
4. Enter the port where the server resides, and select Telnet server emulation options.
5. In the Telnet Commands pane, click the **New** or **Copy** buttons to add or copy a command.

6. Within the Telnet table, define the commands and values that you want to use. The commands that you enter should correspond exactly to the commands that you entered for the client. For example, if the client expects username, the first command for the server to send is username.

   The following shows an example.



   NOTE: The close command that you specify at the end of the session should be **do** on the client side, and **expect** on the server side, or vice-versa.

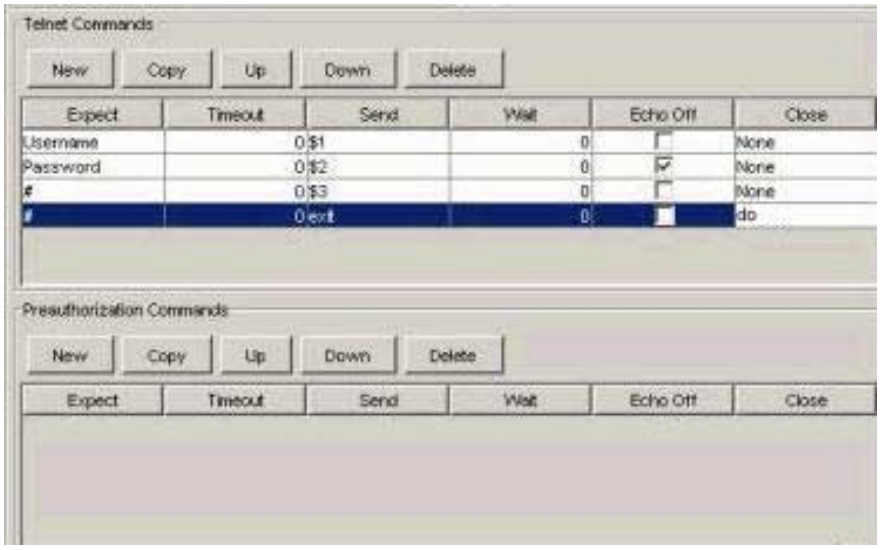
Creating a Forms Database for a Telnet Test

NOTE: Complete this section only if you want to use a forms database to supply Telnet information.

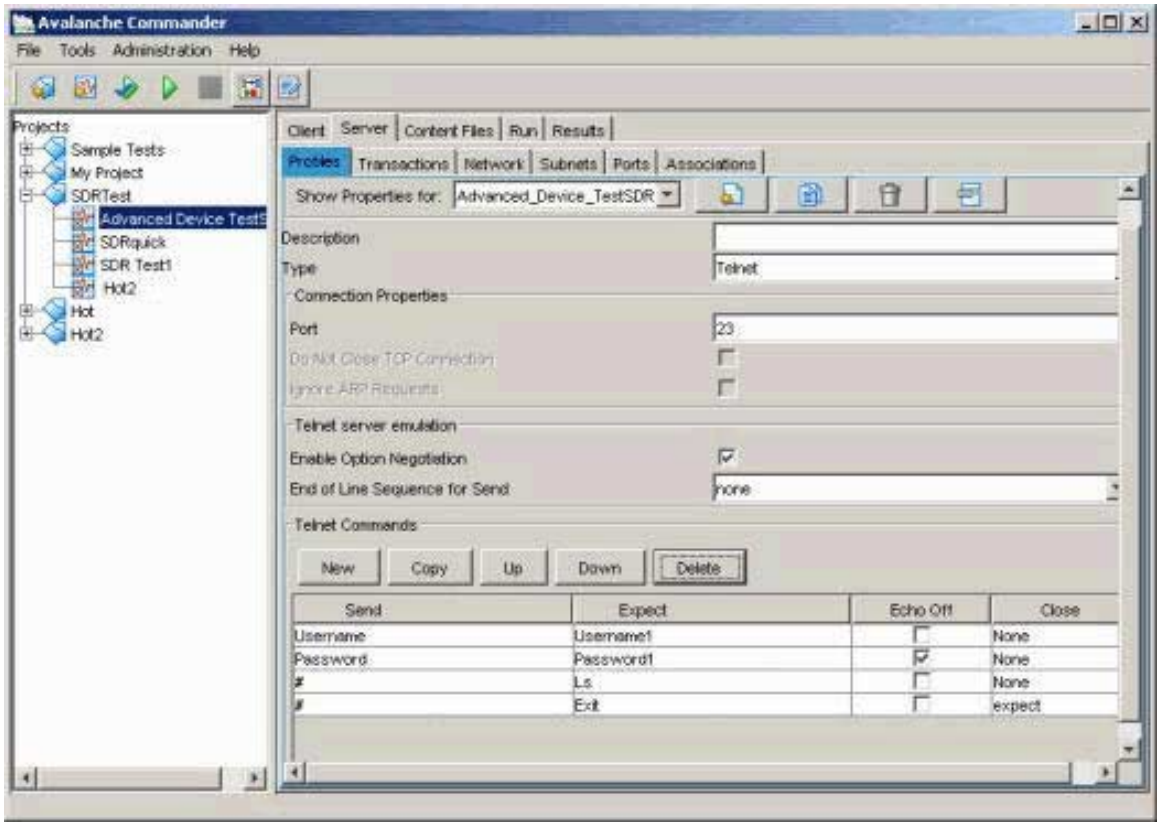
You can also use a forms database to supply information for the client for use with a Telnet protocol. (You cannot use a forms database to supply information for the server.) In the Telnet Commands pane, you identify the form database column of data that you want to send by entering $ and the form database column number such as $1 to refer to the first column, $2 to the second, and so on. For example, in the following image, the $1 in the first Send field refers to the first column of data which will supply a username. You identify the forms database file name in an Actions list associated with the test.

NOTE: For more information about Action Lists, see Using Forms Databases with Action Lists in the Help.

The following image shows how the Client Telnet commands are set up to reference columns 1, 2, and 3 ($1, $2, $3) of a forms database.



The following image shows how the Server Profiles tab is set up to respond to the client.



NOTE: Servers do not support a forms database. So that the server is ready to respond to the information sent from the client, create server profiles that include Send and Expect values corresponding to each column of data in the forms database and for each SimUser accessing the database. For example, in the previous image of the Server Profiles tab, Send and Expect data indicates username and password values that the server should expect.

To create a forms database:

1. In the **Forms** tab on the **Client Actions** tab, click the **New** button  . The **Create New Forms Database** window appears.
2. Complete entries in the Create New Form Data window, and then click **OK.**

The following entries in a forms database could support the examples shown previously in this section:

```
joe,joepass,ls

maria,mariapass,ls

fred,fredpass,ls

harry,harrypass,ls

jose,josepass,ls

kate,katepass,ls

thuy,thuypass,ls

lee,leepass,ls

fran,franpass,ls

rajiv,rajivpass,ls
```

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom. You can create an Action list for Application tests or for Device tests.

To create an Actions list:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter a description for the Actions list.
4. Define the Telnet action as follows.

Syntax

```
telnet://IP address[:port] PROFILE=profile name [FORMS=forms database name]
```

- IP address:port number—IP address for the test. (Port is optional; if it is not included, then 23 is assumed.)
- PROFILE=filename—Specifies the Telnet profile to use with this Action list. The profile name that you specify is the profile that you select in the Telnet Client Emulation pane from the Choose Profile drop-down menu on the Client Profiles tab.
- FORMS=filename—If you use forms database values in the Telnet profile (example username = $1), assign the name of the database here. Different Actions can use different forms databases, but only one forms database can be used on each separate Action.

Examples

- An IPv4 address:

  ```
  telnet://192.168.42.11 PROFILE=telnet_profile
  ```

- An IPv6 address:

  ```
  telnet://[2106::0200:FF:FE00:8102] PROFILE=telnet_profile
  ```

- An IPv4 address with reference to a forms database:

  ```
  telnet://192.168.42.11 PROFILE=Telnet FORMS=telnet_db
  ```

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing VoD Multicast

To complete Video On Demand (VoD) Multicast-specific information and run the test:

1. Learn about Avalanche VoD Multicast testing.
2. Define information for the client.
3. Add an Actions list.
4. Define the server profile. (Device test only)
5. Add content files. (Spirent-RTP or MPEG-2 TS transport method only)
6. Run the test and review results.

About VoD Multicast Testing

VoD is deployed extensively internationally and in the United States. It is important because of the need to test Triple Play (Voice, Video, and Data). Avalanche VoD multicast support provides extensive capabilities for testing the video part of triple play deployments. With Avalanche VoD multicasting emulation, the client emulates TV viewers who are surfing or changing channels. This test helps you assess the behavior of VoD infrastructures. To exercise and measure Digital Subscriber Line Access Multiplexer (DSLAM) behavior, the client emulates TV viewers, while the server emulates video streaming servers.

NOTE: Directing multicast traffic to the correct VLAN is the responsibility of the router.

Defining Client Information

Define the client Video Quality Analyzer (VQA) parameters that you want to use.

To configure the client VQA parameters:

1. Click the **Client** tab, and then the **Profiles** tab.
2. Select the client profile with which you want to work.
3. Click the **VQA** tab, and select the **Enable Video Quality Analysis** checkbox to enable the VQA parameters and generate video quality statistics. For more information on the VQA parameters, see Client Profiles VQA Fields in the Help.

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom.

Each line in the Action list represents a channel (multicast group address) to which to tune. You can also intermix any of the supported Action schemes. For example, interspersing http:// emulates a user accessing the Internet, between tuning to streaming channels.

To add an Actions list for a VoD Multicast test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter information that emulates multicasting. Refer to the following syntax and examples.

Syntax - Basic

```
time_in_ms MCAST://mcast_group_addr

time_in_ms MCRTP://mcast_group_addr
```

- Time_in_ms—Specifies the time, in milliseconds, to remain attached to the channel.
- MCAST://—Specifies raw UDP. This must match the transport method that you choose in the Server Profile tab.
- MCRTP://—Specifies RTP. This must match the transport method that you choose in the Server Profile tab.
- Mcast_group_addr—Specifies the multicast group address. This must match the group address that you choose in the Server Profile tab.

NOTE: If you use RTP streams (mcrtp://. channels listed in the Action list), statistics related to RTP streaming are counted. The mcrtp:// scheme is valid only if the server is producing RTP data on the channel. If the server is broadcasting some other data format, then the RTP statistics will be meaningless.

Example - Basic

In the following example, the viewer tunes to each of .1, .2, and .3 channels for five seconds, then settles on .4 for an hour:

```
5000 MCAST://225.1.1.1

5000 MCAST://225.1.1.2

5000 MCAST://225.1.1.3

3600000 MCAST://225.1.1.4
```

The previous example uses an IPv4 address. The following example uses an IPv6 address:

```
5000 MCAST://[FF0E::2222:2222:1]
```

NOTE: The Internet Group Management Protocol (IGMP) controls multicast channels. In the examples above, an IGMP leave is sent after each action has completed (that is, after its timer expires).

Syntax - With Keywords

```
MCAST://mcast_group_addr DURATION=time_in_ms | DATA=num_of_packets
| LINGER=time_in_ms

MCRTP://mcast_group_addr DURATION=time_in_ms | DATA=num_of_packets
| LINGER=time_in_ms
```

- MCAST://—Specifies raw UDP. This must match the transport method that you choose in the Server Profile tab.
- MCRTP://—Specifies RTP. This must match the transport method that you choose in the Server Profile tab.
- Mcast_group_addr—Specifies the multicast group address. This must match the group address that you choose in the Server Profile tab.
- DURATION=Time_in_ms—Specifies the time, in milliseconds, to listen to the channel.
- DATA=num_of_packets—Specifies the number of packets for which to listen.
- LINGER=time_in_ms—Specifies the time, in milliseconds, to wait before joining the next channel after leaving a session.

Examples - With Keywords

- `MCAST://225.1.1.1 DURATION=5000`

  Listens to the channel for 5 seconds.

- `MCAST://225.1.1.1 DATA=100`

  Listens for 100 packets.

- `MCAST://225.1.1.1 LINGER=1000`

  Waits 1 second before joining the next channel after leaving a session.

NOTE: After the action duration expires, stream data will cease, along with any VQA analysis. Therefore, the duration that you specify in the action must be at least the same as the duration of the load profile.

Forms Database

You can use a forms database to provide many concurrent channels, as in the following example:

```
ASSIGN VARIABLE <myvar myformsdb.$1>

100000 MCAST://<APPLY myvar>
```

Where `myvar` is a variable name, and `myformsdb` is the name of a forms database.

An example forms database is as follows:

```
225.1.1.1

225.1.1.2

.

.

225.1.1.150
```

This example allows 150 channels, as defined in the forms database, to be received concurrently and continuously.

NOTE: The Internet Group Management Protocol (IGMP) controls multicast channels. In the example above, no IGMP leaves are sent until the entire group of 150 channels have completed their action (after 100 seconds), at which time, IGMP leaves are sent for all channels.

Monitoring VQA Statistics

To monitor VQA statistics in the Client VQAStreams tab, you must specify the REAL_TIME_METRICS keyword in your Actions list, as in the following example:

```
100000 mcast://225.1.1.1 REAL_TIME_METRICS
```

Defining the Server Profile

Complete this section only if you are defining a Device test.

To define a VoD Multicast server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **Mcast** as the server type.
4. Enter the port where the server resides. The default port is 2001.
5. Select settings in the Multicast Server Emulation pane. For more information about multicast fields, see Server Profiles Tab VoD Multicasting in the Help.

   NOTES:

   - Avalanche supports uploading MPEG-2 TS or MPEG-4 files in MPEG-2 TS format.
   - Selecting MPEG-2 TS as the content type with RTP as the transport method generates RTP-encapsulated MPEG-2 TS streams according to RFC 2250 recommendations.
   - If you are doing VQA testing, select MPEG-2 TS as the content type, and use the appropriate MPEG-2 TS or RTP transport stream.

Adding Content Files

If you select Spirent-RTP or MPEG-2 TS as the transport method, add the content file that you want to use in the server profile.

NOTE: Spirent-RTP is a Spirent-proprietary file format that consists of pre-formed RTP packets. Spirent supplies several files of this format.

To add content files:

1. Click the **New** button next to the Content drop-down menu. A directory dialog box appears.
2. Locate and select the file that you want, and then click the **Open** button. The file appears in the drop-down menu.

Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.

Testing Windows Media (MMS)

To complete Windows Media-specific information and run the test:

1. Learn about Avalanche Windows Media testing.
2. Add an Actions list.
3. Add content files. (Device test only)
4. Define the server profile. (Device test only)
5. Run the test and review results.

About Windows Media Testing

With version 9, Windows Media has gained widespread acceptance in the streaming community and is poised to achieve even more penetration in the future. Avalanche emulates a number of Microsoft Windows Media Players that retrieve Windows media files. Avalanche supports MMS (Microsoft's proprietary streaming protocol) and RTSP/RTP over the following transport protocols: UDP, TCP, and HTTP. (See Testing RTSP/RTP Streaming for more information about RTSP/RTP.)

Adding an Actions List

Action lists identify the actions that Avalanche takes for each simulated user during a test. Each list line represents a requested object from your test server. As Avalanche simulates new users, each one uses the list from top to bottom. An Action list uses two elements: Level and URI. Enter a 1 preceding the URI to signify a top-level retrieval. Within your Actions list, you identify media files.

For each Action list entry, Avalanche establishes a connection, requests playback, and then terminates the connection when it has completed.

NOTE: The Ramp Down phase time in the load profile must have enough time to allow all the streams to finish playing; otherwise, incomplete sessions are counted as failures. See the Loads Tab in the Help for more information.

To add an Actions list for a Windows-Media test:

1. Click the **Client** tab, and then the **Actions** tab.
2. Click the **New** button  to create a new list.
3. Enter information that simulates retrieving media. Refer to the following syntax and example.

Syntax

```
1 mms://server_IP_address/directory/filename.asf

1 mms://server_IP_address/filename.asf
```

- Server IP address—The address of the server, such as 10.1.79.1.

- Directory-The directory where the file is stored. Define this information if you are testing against an actual server. It is not required if you are using an emulated server.
- Filename-The name of the media file.

NOTE: .asf is the only MMS file format supported.

Example

```
1 mms://192.168.42.11/mymmsfiles/welcome1.asf

1 mms://192.168.42.11/welcome1.asf
```

Adding Content Files

Complete this section only if you are defining a Device test. Add the specified media file as a content file before you start the test.

To add content files:

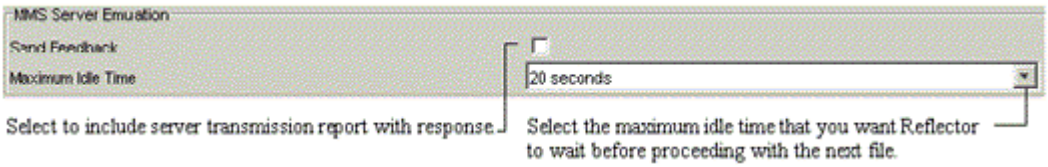
1. Click the **Content Files** tab.
2. Click the **Add** button. A directory dialog box appears. By default, it lists files that appear in the default content file directory. (When you first install the Avalanche software, this directory contains sample streaming files.) Store your content files in this directory.
3. Select the file, and then click the **Add** button. The file appears in the **Content Files** tab.

Defining the Server Profile

Complete this section only if you are defining a Device test. Use a server profile to define the port and server emulation.

To define a server profile:

1. Click the **Server** tab, and then the **Profiles** tab.
2. Click the **New** button  to create a new profile.
3. Enter a description for the profile, and then select **MMS** as the server type.
4. Enter the port where the server resides. The default port is 1755.
5. Select server emulation settings.



Running the Test and Reviewing Results

After completing all set-up steps, run the test.

To run the test:

Click the **Run Test**  icon. The Monitor tab appears. From this tab you can monitor test statistics as Avalanche reports them. For example, click the **Client Stats** or **Server Stats** buttons to view real-time stats and graphical representations.

To view results:

Click the **Results** tab.