Action tasks

by Ovidiu Predescu, Jeff Turner

1. Action tasks

These tasks are used to make HTTP requests to a Web or SOAP server, or to wait for incoming HTTP requests on a local URL. A test uses these tasks to interact with the server or to receive incoming requests from Web or SOAP clients.

Care should be taken if the client and the server machines are separated by a firewall. Anteater allows firewall traversal using HTTP proxies. To setup firewall traversal, use the JVM `http.proxyHost` and `http.proxyPort` properties on the client side, for the HTTP request to work across the firewall. This can be done by setting up the `ANTEATER_OPTS` environment variable like this:

```
$ ANTEATER_OPTS='-Dhttp.proxyHost=<host> -Dhttp.proxyPort=<port>
$ export ANTEATER_OPTS
```

The current Anteater action tasks are:

- httpRequest
- soapRequest
- fileRequest
- listener

1.1. httpRequest

This task makes an HTTP request to a server, and waits for the result. Upon receiving the message, it applies the match tasks specified within it on the HTTP response obtained. If at least one match task succeeds, the request is consider successful.

Attribute name	Type	Default value	Description
description	String		A text description of what the httpRequest achieves. This will be used in reporting.
host	String	`localhost`	The name of the host to which the request is to be sent.
port	integer	`8080`	The port number of the Web server to which the request is to be sent.
timeout	String	`30s`	The socket timeout. This determines how long Anteater waits for a non-responding HTTP service before aborting. The suffixes *ms*, *s* and *m* indicate milliseconds, seconds and minutes. Fractions of seconds and minutes can be used, so `1.5 s` means `1500 ms`. A zero or negative number will set an infinite timeout.
path	String		The path of the HTTP request, including any additional GET parameters.
user	String		The user name to be used with basic authentication. If no user is specified, no authentication is used.
password	String		The password to be used with basic authentication.

href	String		The URL of the Web server to which to send the request. Use either a combination of the host and port attributes, or the `href` attribute, to specify where the server is running. The `host`, `port` and `path` attributes are most commonly used when sending requests to the local host. When you send a request to the local host, you can also ignore the `host` attribute, as the default value is `localhost`.
method	String	`GET`	The HTTP request type ('GET' or 'POST'). This can also be specified as a nested method element.
content	String		An URL of a resource whose content is to be sent to the Web server. If you don't specify any protocol, the assumed protocol is `file:`. If the file doesn't start with a `/`, it is assumed to be relative to the directory where Anteater was started from. The URL of the resource can be remote, e.g. you can specify the content to be `http://www.acme.com/`. Anteater will fetch the remote resource and pass it to the Web server specified by the httpRequest element.

			The content can also be specified by a nested contentEquals element.
protocol	String	`HTTP/1.0`	The HTTP protocol to be used.
debug	Integer		Debug level for messages. Use a debug level greater than 0 to get meaningful information on what's going on over the wire.
useTidy	boolean	`false`	Whether JTidy should be applied on the response obtained from the server to XML-ize the generated HTML.
followRedirects	boolean	`false`	Whether, if the HTTP request returns a 302 client-side redirect message, Anteater should automatically issue another HTTP request to the redirected-to URL and return that response. This is useful in many scenarios, eg where a login page automatically redirects a logged-in client to another page.

Table 1: Attributes

Element name	Description
header	HTTP header to be passed in the HTTP request sent to the server.
parameter	Specifies an additional GET or POST parameter to be passed in the HTTP request to the server.
method	Specifies whether to do a HTTP GET or POST operation. Overrides the 'method' attribute.

contentEquals	Specifies the HTTP body contents to send, probably as a HTTP POST operation. The Content-Length header will be automatically computed and added. This element overrides the 'content' attribute.
match	Specifies a set of rules to be used to match the HTTP response against. You can specify multiple match elements as children of an httpRequest or soapRequest element. If all the tests inside the `match` element succeed, the matching is considered successful and the action succeeds, otherwise the action fails.
namespace	Assigns a namespace mapping, from prefix to URI. This mapping will be used in any XML-aware testers defined in this action task.
logger	Used to specify a logger for this HTTP operation. Often this will be a reference, eg `<logger refid="mylogger"/>`. It is usually preferable to set a logger in a group, than directly on action tasks.
session	Used to specify a session for this HTTP operation. This is useful when one specifically doesn't want to use the default session, or wants to run a few HTTP operations with a completely separate session. Often this will be a reference, eg `<session refid="mylogger"/>`. It is usually preferable to set a session in a group, than directly on action tasks.
uses	If this action task requires an Anteater optional feature (eg a new XSD or RNG schema), then this tag can be used to declare the dependency. The task will check if the requirement is satisfied, and print a helpful message if it isn't. Note that 'uses' tags from the action task's groups are also evaluated when the action task is run.

Table 2: Elements allowed inside httpRequest

Examples

Send an HTTP `GET` request to `http://localhost:8080/`:

```
<httpRequest/>
```

Equivalent to an HTTP `GET` request to `http://localhost/`:

```
<httpRequest port="80"/>
```

Equivalent to an HTTP `POST` request to `http://localhost:8080/servlets/example` passing the content of `/etc/passwd` to the Web server:

```
<httpRequest path="/servlets/example" method="POST" content="/etc/passwd"/>
```

1.2. soapRequest

Sends a SOAP request to a SOAP server. This is equivalent to the httpRequest task, but with the following additions:

- an additional `SOAPAction` header set to `""` is passed in the request.
- the method is set to `POST`
- the `Content-type` header is set to `text/xml`.

Attribute name	Type	Default value	Description
description	String		A text description of what the httpRequest achieves. This will be used in reporting.
host	String	`localhost`	The name of the host to which the request is to be sent.
port	integer	`8080`	The port number of the Web server to which the request is to be sent.
timeout	String	`30s`	The socket timeout. This determines how long Anteater waits for a non-responding HTTP service before aborting. The suffixes *ms*, *s* and *m* indicate milliseconds, seconds and minutes. Fractions of seconds and minutes can be used, so `1.5 s` means `1500 ms`. A zero or negative number will set an infinite timeout.

path	String		The path of the HTTP request, including any additional GET parameters.
user	String		The user name to be used with basic authentication. If no user is specified, no authentication is used.
password	String		The password to be used with basic authentication.
href	String		The URL of the Web server to which to send the request. Use either a combination of the host and port attributes, or the `href` attribute, to specify where the server is running. The `host`, `port` and `path` attributes are most commonly used when sending requests to the local host. When you send a request to the local host, you can also ignore the `host` attribute, as the default value is `localhost`.
method	String	`GET`	The HTTP request type ('GET' or 'POST'). This can also be specified as a nested method element.
content	String		An URL of a resource whose content is to be sent to the Web server. If you don't specify any protocol, the assumed protocol is `file:`. If the

			file doesn't start with a `/`, it is assumed to be relative to the directory where Anteater was started from. The URL of the resource can be remote, e.g. you can specify the content to be `http://www.acme.com/`. Anteater will fetch the remote resource and pass it to the Web server specified by the httpRequest element. The content can also be specified by a nested contentEquals element.
protocol	String	`HTTP/1.0`	The HTTP protocol to be used.
debug	Integer		Debug level for messages. Use a debug level greater than 0 to get meaningful information on what's going on over the wire.
followRedirects	boolean	`false`	Whether, if the HTTP request returns a 302 client-side redirect message, Anteater should automatically issue another HTTP request to the redirected-to URL and return that response. This is useful in many scenarios, eg where a login page automatically redirects a logged-in client to another page.
useTidy	boolean	`false`	Whether JTidy should be

			applied on the response obtained from the server. Since we send a SOAP request, we expect to receive back a SOAP message, so there's little reason to XML-ize this content. Set this attribute to true if the server returns broken HTML instead of SOAP.

Table 1: Attributes

Element name	Description
header	HTTP header to be passed in the HTTP request sent to the server.
parameter	Specifies an additional GET or POST parameter to be passed in the HTTP request to the server.
method	Specifies whether to do a HTTP GET or POST operation. Overrides the 'method' attribute.
contentEquals	Specifies the HTTP body contents to send, probably as a HTTP POST operation. The Content-Length header will be automatically computed and added. This element overrides the 'content' attribute.
match	Specifies a set of rules to be used to match the HTTP response against. You can specify multiple match elements as children of an httpRequest or soapRequest element. If all the tests inside the `match` element succeed, the matching is considered successful and the action succeeds, otherwise the action fails.
namespace	Assigns a namespace mapping, from prefix to URI. This mapping will be used in any XML-aware testers defined in this action task.
logger	Used to specify a logger for this HTTP operation. Often this will be a reference, eg `<logger refid="mylogger"/>`. It is usually preferable to set a logger in a group, than directly on action tasks.

session	Used to specify a session for this HTTP operation. This is useful when one specifically doesn't want to use the default session, or wants to run a few HTTP operations with a completely separate session. Often this will be a reference, eg `<session refid="mylogger"/>`. It is usually preferable to set a session in a group, than directly on action tasks.
uses	If this action task requires an Anteater optional feature (eg a new XSD or RNG schema), then this tag can be used to declare the dependency. The task will check if the requirement is satisfied, and print a helpful message if it isn't. Note that 'uses' tags from the action task's groups are also evaluated when the action task is run.

Table 2: Elements allowed inside soapRequest

Examples

This example demonstrates how a listener's match tasks and testers can be arranged to implement if/then/else logic.

The incoming request can be a SOAP message containing either an receipt acknowledge message or a SOAP fault message, indicating an error.

```
<listener path="/receipt">
    <match method="POST">
      <xpath select="/soap:Envelope/soap:Body/receipt-ack"/>
      <xpath select="/soap:Envelope/soap:Body/response-to" assign="replyHref"/>
      <sendResponse href="${replyHref}/>
    </match>
    <match assign="failed">
      <matchBody select="/soap:Envelope/soap:Body/rfq"/>
      <matchMethod code="POST"/>
    </match>
</listener>
```

1.3. fileRequest

This task is the same as httpRequest, but tests against a local file instead of doing a HTTP request.

Attribute name	Type	Default value	Description
description	String		A text description of what the httpRequest achieves. This will be

			used in reporting.
path	String		The path of the local file to test, relative to the Anteater script's `basedir`
debug	Integer		Debug level for messages. Use a debug level greater than 0 to get meaningful information on what's going on over the wire.

Table 1: Attributes

Element name	Description
match	Specifies a set of rules to be used to match the HTTP response against. You can specify multiple match elements as children of an httpRequest, soapRequest or fileRequest element. If all the tests inside the `match` element succeed, the matching is considered successful and the action succeeds, otherwise the action fails.
namespace	Assigns a namespace mapping, from prefix to URI. This mapping will be used in any XML-aware testers defined in this action task.
logger	Used to specify a logger for this HTTP operation. Often this will be a reference, eg `<logger refid="mylogger"/>`. It is usually preferable to set a logger in a group, than directly on action tasks.
uses	If this action task requires an Anteater optional feature (eg a new XSD or RNG schema), then this tag can be used to declare the dependency. The task will check if the requirement is satisfied, and print a helpful message if it isn't. Note that 'uses' tags from the action task's groups are also evaluated when the action task is run.

Table 2: Elements allowed inside fileRequest

Examples

Reads a local file, `resources/responses/text.txt`, and checks that its contents (ignoring whitespace) is a certain value.

```
              <fileRequest path="resources/responses/text.txt"
                           description="tests a text file">
                <match>
                  <contentEquals ignoreSpaces="true">
                    Here is some freeform text saved with DOS linefeeds.
                  </contentEquals>
                </match>
            </fileRequest>
```

1.4. listener

In addition to sending out HTTP requests to Web and SOAP servers, Anteater has the ability to receive incoming HTTP requests. This ability is very useful when you want to implement high level SOAP and XML protocols, like ebXML or BizTalk, which make use of asynchronous SOAP messages to exchange information between parties.

Applications implementing such protocols will accept an HTTP request as a high level asynchronous request. The response to such a request is not usually meaningful. Instead, the server will later generate a reply as another HTTP request to an URL specified by the client in the original message.

In such applications, the client and server role changes depending on the phase of the conversation. To be able to test such applications, a party in such a conversation should be able to act both as a client and as a server.

The listener element tells Anteater to stop the processing of the test script, until a request at a specified URL is received. Anteater will act exactly like an HTTP or SOAP server, by listening on the local host on a specified port, waiting for a request on a given URI path you can specify.

To use the `listener` task, the servlet container within Anteater should first be started. This is done by using the servletContainer task, which allows specifying which are the ports Anteater will listen on when acting as an HTTP server.

In the next example, a request is sent to a SOAP server, and then a response is awaited on the local host at the `/receiptAck` path URI. If the name of the machine which runs this Anteater snippet is `soap.acme.com`, the remove SOAP server would then need to send an HTTP request back to `http://soap.acme.com:8080/receiptAck` for the listener task to be unblocked and the Anteater script's execution to continue:

```
<target name="test">
```

```
  <soapRequest href="http://some.remote.server/"
               content="some/file"/>

  <listener port="8080" path="/receiptAck" timeout="7200">
    <namespace prefix="soap" uri="http://schemas.xmlsoap.org/soap/envelope/"/>
    <match>
      <method value="POST"/>
      <xpath select="/soap:Envelope/soap:Body/receipt-ack"/>
      <sendResponse href="responses/response.xml"
                    contentType="text/xml"/>
    </match>
  </listener>
</target>
```

In the above example, if no request is received within 7200 seconds from the start of the listening, the listener task will fail. If a response is received within this time, the incoming request should be an HTTP `POST` request, and should contain in the body a SOAP message with a `receipt-ack` element, for the listener task to succeed. If such a request is received, a response is sent back with the content a local file, using the sendResponse task.

If there's no match task that matches the incoming request, a response may not be generated using sendResponse. In such a case, Anteater will automatically generate a `200 OK` response, with no content in the body, and the enclosing listener element fails. Sending the response ensures the client application obtains a response back, and doesn't block it indefinitely. Future versions of Anteater will allow for the customization of such responses.

For any incoming requests, for which there's no listener task waiting, the response sent back by Anteater is a `404 Not Found`.

Attribute name	Type	Default value	Description
path	String		Defines the URI path where an HTTP request should be received on. Requests received on other URI paths will have a `404 Not Found` response returned, unless other listeners are setup for them.
port	integer		The first port defined by the `port` attribute of servletContainer. See a description of this element for more information.

timeout	integer	`0`	How long in seconds the listener task should wait for an incoming request. If no request is received before the time expires, the listener task fails. A value of `0` specifies an indefinite timeout, so the listener task will wait forever for a request to come.
useTidy	boolean	`false`	Whether JTidy should be applied on the incoming request body, before running the matcher tests on it.

Table 1: Attributes

Element name	Description
match	Match on the incoming request.
namespace	Assigns a namespace mapping, from prefix to URI. This mapping will be used in namespace-aware tasks like xpath. **Note:** This is really a hack; if Ant made namespaces available to tasks, one could instead use normal XML xmlns attributes to declare namespace mappings.
logger	Used to specify a logger for this HTTP operation. Often this will be a reference, eg `<logger refid="mylogger"/>`. It is usually preferable to set a logger in a group, than directly on action tasks.
session	Used to specify a session for this HTTP operation. This is useful when one specifically doesn't want to use the default session, or wants to run a few HTTP operations with a completely separate session. Often this will be a reference, eg `<session refid="mylogger"/>`. It is usually preferable to set a session in a group, than directly on action tasks.

Table 2: Elements allowed inside listener

Examples

This example demonstrates how a listener's match tasks and testers can be arranged to implement if/then/else logic.

The incoming request can be a SOAP message containing either an receipt acknowledge message or a SOAP fault message, indicating an error.

```
<listener path="/receipt">
    <match method="POST">
      <xpath select="/soap:Envelope/soap:Body/receipt-ack"/>
      <xpath select="/soap:Envelope/soap:Body/response-to" assign="replyHref"/>
      <sendResponse href="${replyHref}/>
    </match>
    <match assign="failed">
      <matchBody select="/soap:Envelope/soap:Body/rfq"/>
      <matchMethod code="POST"/>
    </match>
</listener>
```