The HTTP Plug-in

Table of contents

1 HTTP plug-in class

To use the HTTP plug-in, specify:

```
grinder.plugin=net.grinder.plugin.http.HttpPlugin
```

2 HTTP plug-in properties

This table lists the HTTP plug-in properties that you can set in `grinder.properties` in addition to the core properties (../g3/properties.html) . You can use the TCPSniffer to record (../g2/tcpsniffer.html#HttpPluginSnifferFilter) HTTP plug-in scripts.

`grinder.test0.parameter.url`	The URL to call. The HTTP `GET` method is used unless `grinder.test0.parameter.post` is specified. The contents of the file can be varied using a `string bean`, see this FAQ (../faq.html#post-and-string-beans) .
`grinder.test0.parameter.post`	Specify a file containing `POST` data to send. The value can be varied using a string bean.
`grinder.test0.parameter.header.name`	(Where `name` can be an arbitrary string). Add a `name:` header to the request with the specified value. The value can be varied using a string bean.
`grinder.plugin.parameter.stringBean`	Fully qualified class name of a Java bean that can generate dynamic strings. See What's a String Bean?.
`grinder.test0.parameter.ok`	Fail if the returned page doesn't contain this string. The value can be varied using a string bean.
`grinder.test0.parameter.basicAuthen` `grinder.test0.parameter.basicAuthen` `grinder.test0.parameter.basicAuthen`	Used together, these specify an HTTP BASIC authentication header that will be sent with the request. If you specify one of these values, you must specify all three. Note, the default HTTPClient implementation only sends this if challenged by the server, as a browser would, and the specified basicAuthenticationRealm must match the realm required by the WWW-Authenticate header in the challenge. The values can be varied using a string bean.
`grinder.plugin.parameter.useCookies`	Set to `false` to disable cookie handling (it defaults to `true`).
`grinder.plugin.parameter.disablePers`	Set to `true` to send a `Connection: close` message with every request. See http://www.innovation.ch/java/HTTPClient/advanced_info.html#pers_con. Only works with the default HTTPClient implementation. The default is `false`.
`grinder.plugin.parameter.followRedi`	Set to `true` to automatically follow redirects, so you don't have to have additional URLs in you scripts. The default is `false`. You should always set this to `false` for TCPSniffer generated scripts.

	If you are using WebLogic Server and Web Application form-based authentication you leave this property set to false. This is because a redirect request that follows authentication contains a key cookie - setting the property to `true` prevents the cookie from being picked up by the plug-in.
`grinder.useHTTPClient`	Controls which of the two different HTTP libraries the HTTP plug-in uses. The default value is `true`, meaning that the HTTPClient implementation should be used.

2.1 What's a String Bean?

When using the HTTP plug-in, it is often necessary to parametrise parts of URLs, POST data, and other request strings. The easy way to do this is to use a *string bean* - a Java Bean that returns `String`s from its `get` methods.

For example, suppose you wanted to modify the URL `http://myhost/test.jsp?n=1` such that `n` is set to a random number each time? Easy! Here's how.

1. Write a simple bean:

```
// MyStringBean.java
package mystuff;

import java.util.Random;

public class MyStringBean {
  private Random m_random = new Random();

  public String getRandomInteger() {
    return Integer.toString(m_random.nextInt());
  }
}
```

2. Compile this and put it in your `CLASSPATH`. Then alter your `grinder.properties` to include this line:

```
grinder.plugin.parameter.stringBean=mystuff.MyStringBean
```

3. Alter the test URL parameter as follows:

```
grinder.test0.parameter.url=http://myhost/test.jsp?n=<getRandomInteger>
```

The `<beanMethodName>` syntax can be used in URL strings, `POST` data files, HTTP headers and in OK strings. It must correspond to a `public` method of the string bean that takes no parameters and returns a `String`.

The HTTP plug-in is relaxed about a partial string bean tag matches (for example `<notAMethod>` or `<abc<def>xyz>`); if it can't find a suitable match it simply outputs the literal text. This allows string bean tags to be used within XML `POST` data. If you find that your string bean is not invoked when you expect it to be, use the TCPSniffer (../g2/tcpsniffer.html) to find out what is actually being sent and check your spelling.

Each string bean instance is instantiated per thread, and maintains its state between invocations. If your bean needs additional information regarding the test life cycle (for example, to reset a counter and the beginning of a cycle), it can implement the

`net.grinder.plugin.http.StringBean` interface. See the examples in `net.grinder.plugin.http.example`.

Advanced string beans can implement the `net.grinder.plugin.http.StringBean` and/or `net.grinder.plugin.http.HTTPClientResponseListener` interfaces to receive callbacks about the test life cycle. See the examples in `src/net/grinder/plugin/http/example`.

3 HTTPClient

The HTTP plugin has two implementations. The default implementation is based on Ronald Tschalär's excellent **HTTPClient** library. An alternative implementation which uses the JDK's `HttpURLConnection` can also be used, but is deprecated and will be removed from The Grinder in a future release. If you really want to you can specify that the HTTP plug-in should use `HttpURLConnection` instead of HTTPClient:

```
grinder.plugin.parameter.useHTTPClient=false
```

I highly recommend the HTTPClient implementation, see below for some reasons why. However the HttpURLConnection implementation has two features that the HTTPClient implementation doesn't. The first feature is an additional parameter:

`grinder.plugin.parameter.useCookies\`	Set to `false` to remove the `$Version` string from cookies (it defaults to true). This is to work around broken (?) JRun 2.3.3. behaviour.

The HTTPClient cookie support is damn good, so this probably isn't an issue. It will be fixed if it turns out to be a problem.

The second additional feature is the reporting of the *mean time to first byte* statistic in addition to the normal total transaction time statistic. This will be supported by the HTTPClient implementation in a future release.

3.1 HTTPClient versus HttpURLConnection

HTTPClient has many more features than HttpURLConnection, see http://www.innovation.ch/java/HTTPClient/urlcon_vs_httpclient.html (http://www.innovation.ch/java/HTTPClient/urlcon_vs_httpclient.html) for a comparison. I hope to lever features such as proxy support, connection timeouts and persistent cookies into future versions of The Grinder.

You can access many HTTPClient features by setting system properties. See http://www.innovation.ch/java/HTTPClient/advanced_info.html for a list of properties. For example, you can force HTTPClient to use HTTP 1.0 instead of HTTP 1.1 with the following parameter:

```
grinder.jvm.arguments=-DHTTPClient.forceHTTP_1.0=true
```

One of the key advantages for The Grinder is that HTTPClient allows explicit control of connection management, whereas HttpURLConnection uses connection pooling "under the covers". Because HTTPClient uses extra connections, it may appear slower - particularly if the client and server are co-hosted (.././faq.html#timing) . However, its a better model of reality (one cycle equals one browser session).

In my experience, HTTPClient is *much* more RFC compliant, and less buggy than HttpURLConnection.

4 How do I use HTTPS?

There are patches available to HTTPClient to work with several SSL implementations. See http://www.innovation.ch/java/HTTPClient/https.html for details. The instructions that follow assume you are using JSSE 1.0.3 or later.

1. If you are using an old JVM (earlier than 1.4.1), you'll need to install JSSE1.0.3 (http://www.oracle.com/technetwork/java/jsse-136410.html) . I recommend installing the JSSE as an installed extension for simplicity.
2. Download the HTTPClient JSSE patch from http://www.innovation.ch/java/HTTPClient/JSSE.zip

   Extract the class files contained within the zip into a directory called HTTPClient, then create a jar containing that directory:

   ```
   mkdir HTTPClient; cd HTTPClient
   jar xf /download/JSSE.zip
   cd ..
   jar cf HTTPClient-JSSE.jar HTTPClient
   ```

   Add this jar to the *start* of your `CLASSPATH` before running The Grinder. Its worth reading the file README in JSSE.zip.
3. You can now use URLs that start with `https:` in your `grinder.properties`.

You may well need to create a trust store containing CA certificates that sign the server certificate. See the JSSE documentation for full details, here's a quick hint:

```
keytool -import -v -keystore ./mycastore -file d:/wls5/myserver/ca.pem
```

You should then add `-Djavax.net.ssl.trustStore=mycastore` to `grinder.jvm.arguments` in your `grinder.properties`. Refer to the JSSE documentation for other useful properties. In particular `-Djavax.net.debug=ssl` might come in useful.

HTTPClient checks that the host name in each request URL matches the subject DN field in the certificate. If this isn't the case, you might need to add an entry to `/etc/hosts`, `c:/WINNT40/system32/drivers/etc/`, DNS or whatever, and then use that hostname in the request URLs.

The JSSE SSL implementation isn't quick. This should be taken into account when comparing round trip times, as a compiled browser version is likely to be *a lot* faster.

5 How do I use the HTTPS plug-in? (HttpURLConnection implementation only)

> **Note:**
> This information applies to the deprecated HttpURLConnection implementation. For information on using HTTPS with the default HTTPClient implementation, see above.

1. Install JSSE1.0.3 as described above.
2. In your `grinder.properties`, use the HttpsPlugin rather than HttpPlugin:

```
grinder.plugin=net.grinder.plugin.http.HttpsPlugin
```

3. You can now use URLs that start with `https:` in your `grinder.properties`.
4. If you want two-way authentication, add the lines like:

```
grinder.plugin.parameter.clientCert=./philclient.p12
grinder.plugin.parameter.clientCertPassword=acrobat
```

   You can export a P12 certificate from Netscape.