Browserless REST-like calls

Overview

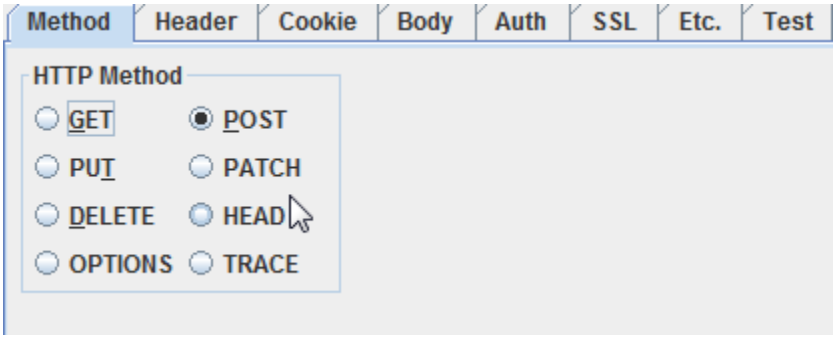
To use the browserless eDART API, currently you need to submit the calls to a helper program provided by PJM called "filetransfer.jar". This program fulfills two main tasks:

- it decides what PJM environment to use- production or training- and configures the urls and connections to that environment
- performs user authentication

After a session is established, this program will forward the actual request to the eDART server, and return back the response to the user. While this program makes some things easier for the user, it also requires the user to indicate the eDART environment to connect to using the 'sandbox' parameter. Also it is an external program which may be a concern to some users.
An alternative is available and in this document, we'll explain how one can also use the eDART browserless API through REST-like http calls directly to the server URLs.
The eDART server URLs are:

URL
https://edart.pjm.com (production)
https://edarttrain.pjm.com (training)

The HTTP method for all calls to the eDART API must be POST, for both downloads and uploads.



Tools for REST

Making HTTP calls to a server can be done in many ways. Any high level modern language (java, .net, python, perl...etc) would have libraries to make HTTP calls and receive the response and act on it. Once you get the HTTP response it's up to you what to do with it - you could save the response in a file, parse it, send an email if there were any errors, and so on.

WizTools.org RESTClient

To get started with REST quickly, there are already a number of user-friendly clients available.
The better known ones are browser extensions, Firefox and Chrome both have a few, but the features are limited and the interfaces are very basic and as such are not recommended.
One widely used option is a standalone open source Java application available at https://github.com/wiztools/rest-client, get the latest version.

The rest of this document uses screenshots from this particular REST client to illustrate how calls are made, but again the user has the choice of making HTTP calls with a different client, or programmatically from the user's own application.
Note that this client allows the user to save requests, which is a very useful feature, and it also has a command line interface.

Sample eDART Java client

The eDART team at PJM developed a simple client in Java to help with testing. It is slightly customized to the needs of eDART, so from a user perspective it is easier to use than a generic tool. Note that this is not an official tool, it's just something a user can study, look at the source and take as a starting point to automate processes in Java. The binary can also be used as a replacement for filetransfer.jar.
The source and binary packages are available here: http://www.pjm.com/pub/etools/edart/xmldocs/xmldoc.html

Since the argument list can get pretty long, it is recommended to save the command calls in shell scripts: .sh for Unix shells, .bat for Windows.
In Unix shell, a command can be split on multiple lines by separating the lines with \ (backslash). In Windows shell files, the character is ^ (caret).
Here's an example of a call in both shells:
For Windows

```
java -jar C:/Personal/edartrest-15.4.0.jar download ^ -username=user1 ^ -
password=**** ^ -url=https://edarttrain.pjm.com ^ -type=transmission ^ -
transtype=revise ^ -id=66747 ^ -output=ticket66747.xml
```

For Unix

```
java -jar C:/Personal/edartrest-15.4.0.jar download \ -username user1 \ -
password **** \ -url https://edarttrain.pjm.com \ -type=transmission \ -
transtype=revise \ -id=66747 \ -output=ticket66747.xml
```

An example of a convenience feature here is that the user only provides the system's host address (edarttrain.pjm.com), not the full servlet path (edarttrain.pjm.com/j2ee/servlet/com.pjm.xml.download.edart.EdartXMLDownload) - the application will construct the full path.

Downloads

For downloads add this path to the server name:

```
j2ee/servlet/com.pjm.xml.download.edart.EdartXMLDownload
```

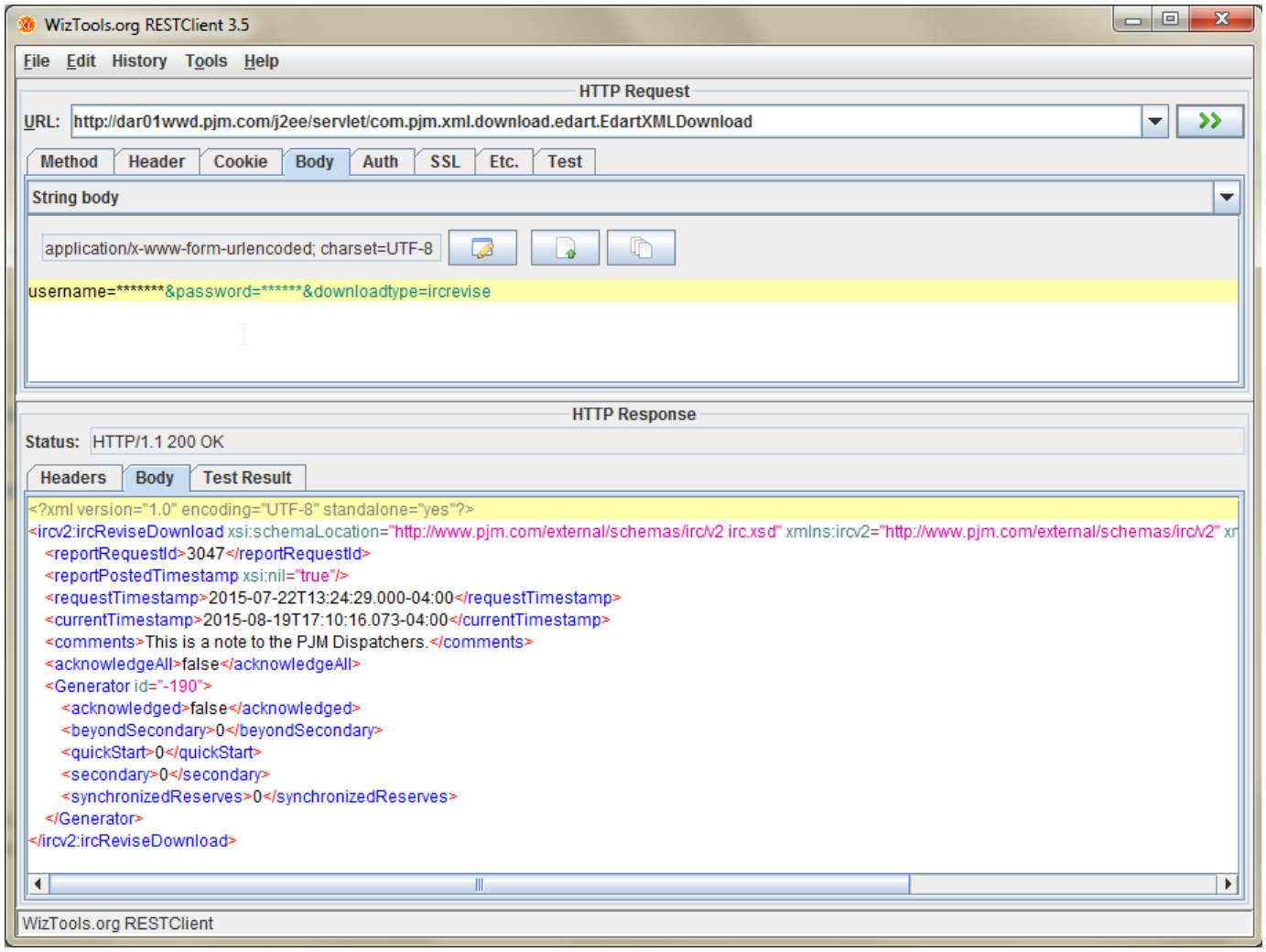
So the download URLs would be:

Download servlet URL
https://edart.pjm.com/j2ee/servlet/com.pjm.xml.download.edart.EdartXMLDownload (production)
https://edarttrain.pjm.com/j2ee/servlet/com.pjm.xml.download.edart.EdartXMLDownload (training/sandbox)

To execute a download, submit (at least) the following parameters:

Parameter
username
password
downloadtype

Certain types of downloads may require additional parameters, for example to download tickets by date the user has to submit start and stop dates.
The http request body would be a urlencoded form entity, here's a screenshot of a sample IRC Revise download:



IRC Uploads

For uploads add this path to the server name:

```
j2ee/servlet/com.pjm.xml.upload.edart.EDartXMLUpload
```

So the upload URLs would be:

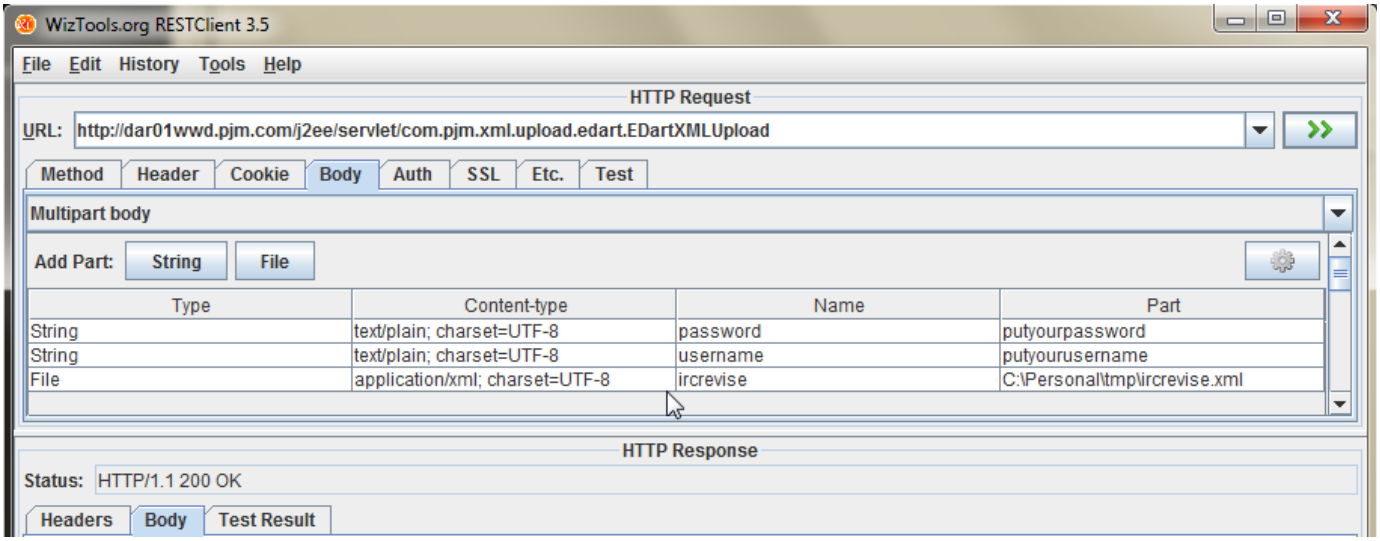
Upload servlet URL
https://edart.pjm.com/j2ee/servlet/com.pjm.xml.upload.edart.EDartXMLUpload (production)
https://edarttrain.pjm.com/j2ee/servlet/com.pjm.xml.upload.edart.EDartXMLUpload (training/sandbox)

To execute an upload, submit a multipart body with the following data parts:

HTTP request body entity
username
password

file to be uploaded

Here's a sample screenshot:



When the file is uploaded, the contents is what matters, not the name. In the screenshot above we see the name as "ircrevise" but it can be anything, the server will process the file based on the contents.

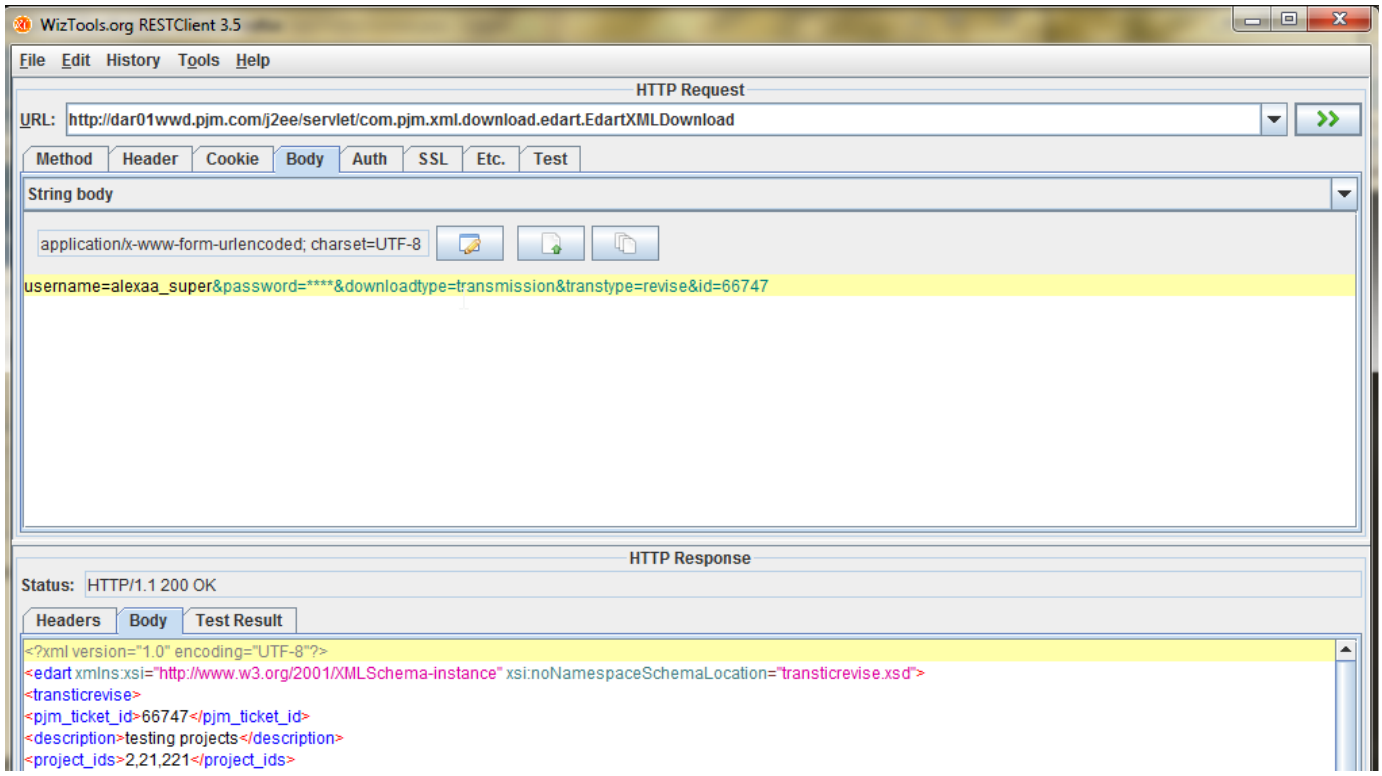
Examples

Transmission ticket download

With eDART client

```
java -jar C:/Personal/edartrest-15.4.0.jar download ^ -username=user1 ^ -
password=**** ^ -url=https://edarttrain.pjm.com ^ -type=transmission ^ -
transtype=revise ^ -id=66747 ^ -output=ticket66747.xml
```

With generic http rest client:

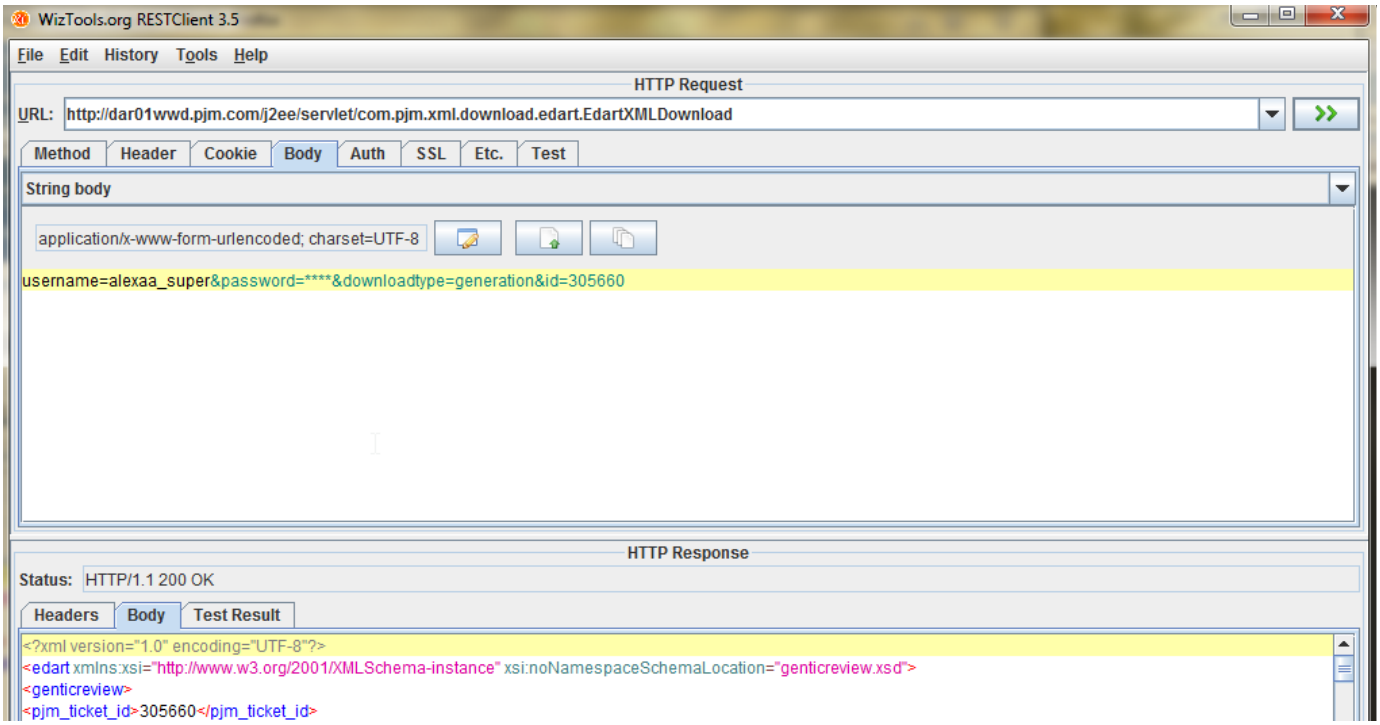


Generation ticket download

With eDART client

```
java -jar C:/Personal/edartrest-15.4.0.jar download ^ -username=user1 ^ -
password=**** ^ -url=https://edarttrain.pjm.com ^ -type=generation ^ -
id=305660
```

With generic http rest client

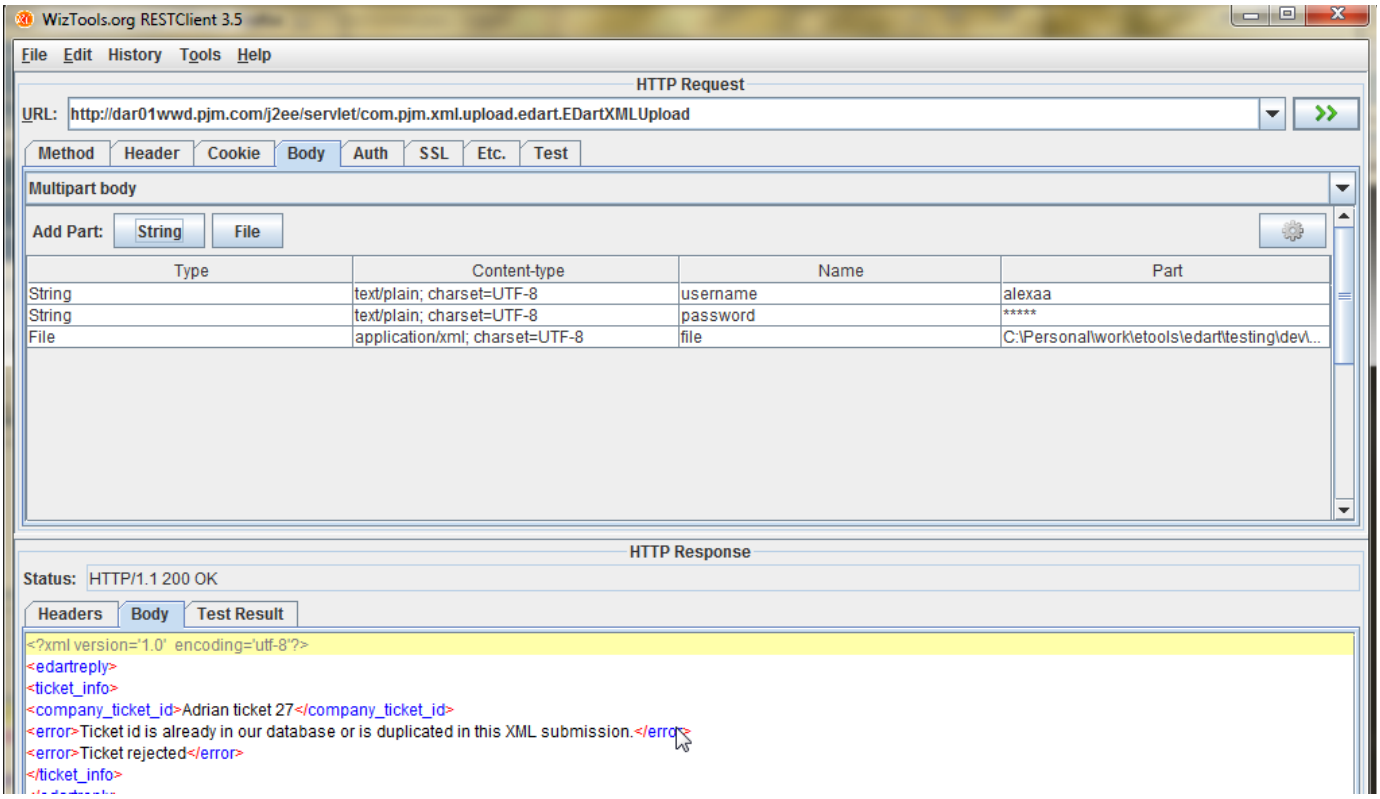


Generation ticket upload

With eDART client

```
java -jar C:/Personal/edartrest-15.4.0.jar upload ^ -username=alexaa ^ -
password=**** ^ -url=https://edarttrain.pjm.com ^ -source="C:
\\Personal\\Gentic\\newgentic1.xml"
```

With generic http rest client

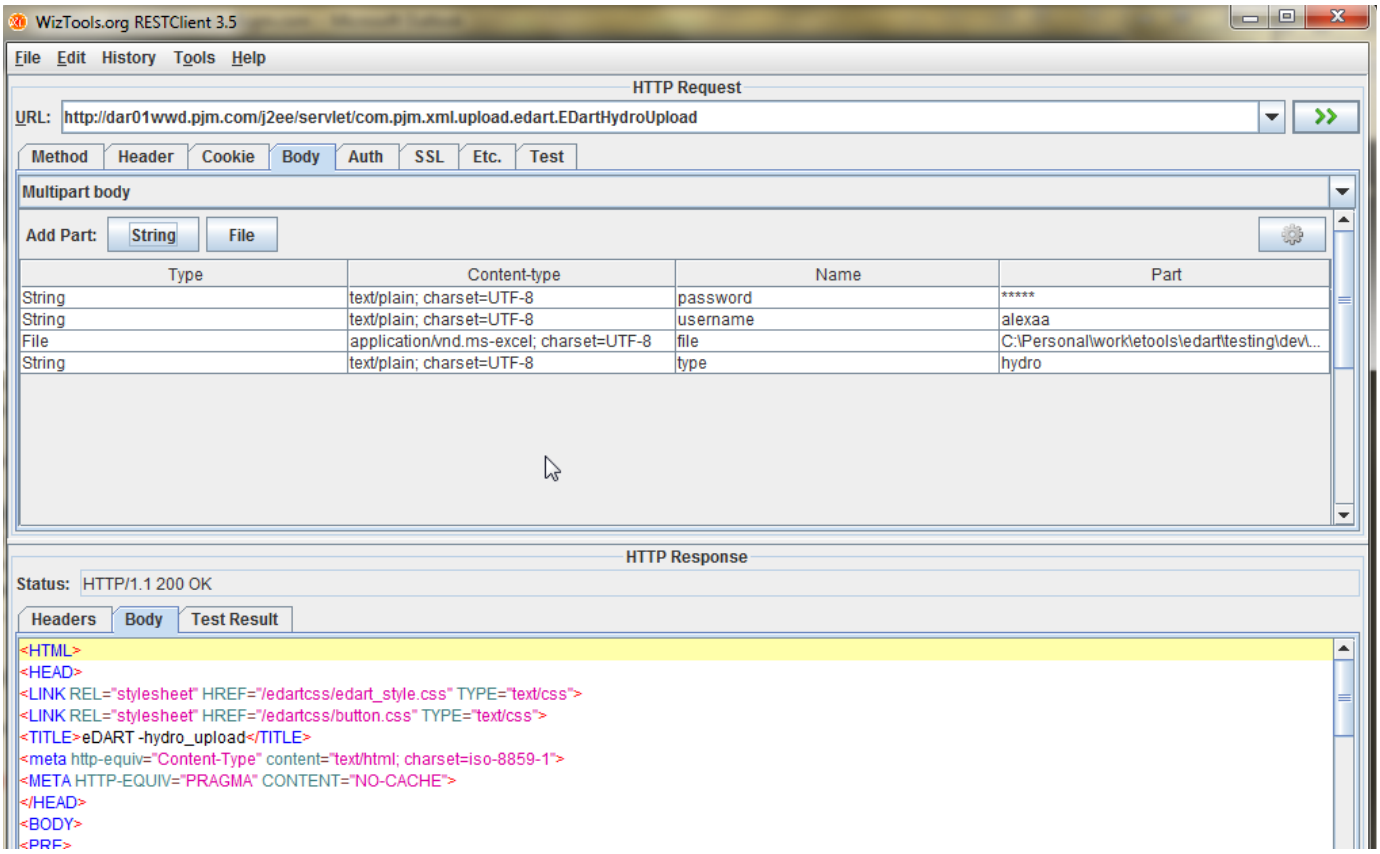


Hydro upload

With eDART client

```
java -jar C:/Personal/edartrest-15.4.0.jar upload ^ -username alexaa ^ -
password **** ^ -url https://edarttrain.pjm.com ^ -source "C:
\\Personal\\Hydro\\HydroForecast_151014125538850.xls" ^ -type hydro
```

With generic http rest client



Note that the Hydro upload servlet URL is currently different than the normal upload URL (EDartHydroUpload, rather than EDartXMLUpload)