Browserless REST-like calls

- Overview
- Tools for REST
 - WizTools.org RESTClient
- Downloads
- IRC Uploads
- Examples
 - Transmission ticket download
 - Generation ticket download
 - Generation ticket upload
 - Hydro upload

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 userl \ -
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:

🕲 WizTools.org RESTClient 3.5		
<u>File Edit History Tools Help</u>		
HTTP Request		
URL: http://dar01wwd.pjm.com/j2ee/servlet/com.pjm.xml.download.edart.EdartXMLDownload		
Method Header Cookie Body Auth SSL Etc. Test		
String body		
application/x-www-form-urlencoded; charset=UTF-8		
username=******&password=******&downloadtype=ircrevise		
HTTP Response		
Status: HTTP/1.1 200 OK		
Headers Body Test Result		
<pre></pre> // intersion="1.0" encoding="UTF-8" standalone="yes"/>		
<pre><ircv2:ircrevisedownload pre="" xmlns:ircv2="http://www.pjm.com/external/schemas/irc/v2" xr<="" xsi:schemalocation="http://www.pjm.com/external/schemas/irc/v2 irc.xsd"></ircv2:ircrevisedownload></pre>		
<reportrequestid>3047</reportrequestid>		
<reportpostedtimestamp xsi:nil="true"></reportpostedtimestamp> <requesttimestamp>2015-07-22T13:24:29.000-04:00</requesttimestamp>		
<currenttimestamp>2015-08-19T17:10:16.073-04:00</currenttimestamp>		
<comments>This is a note to the PJM Dispatchers.</comments>		
<acknowledgeall>false</acknowledgeall>		
<generator id="-190"></generator>		
<acknowledged>false</acknowledged>		
<secondary>0</secondary>		
<synchronizedreserves>0</synchronizedreserves>		
WizTools.org RESTClient		

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

 $\label{eq:https://edart.EDartXMLUpload.edart.EDartXMLUpload (production) } the the transformation of transformation of the transformation of transformat$

 $\label{eq:https://edarttrain.pjm.com/j2ee/servlet/com.pjm.xml.upload.edart.EDartXMLUpload (training/sandbox)$

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:

WizTools.org RESTClient 3.5				
<u>File Edit History Tools H</u> elp				
	HTTP	Request		
URL: http://dar01wwd.pjm.com/j2ee/servlet/com.pjm.xml.upload.edart.EDartXMLUpload				
Method Header Cookie Body Auth SSL Etc. Test				
Multipart body				
Add Part: String File				
Туре	Content-type	Name	Part	
String	text/plain; charset=UTF-8	password	putyourpassword	
String	text/plain; charset=UTF-8	username	putyourusername	
File	application/xml; charset=UTF-8	ircrevise	C:\Personal\tmp\ircrevise.xml	
HTTP Response Status: HTTP/1.1 200 OK				
Headers Body Test Result				

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:

3 WizTools.org RESTClient 3.5	- 0 X
<u>F</u> ile <u>E</u> dit History T <u>o</u> ols <u>H</u> elp	
HTTP Request	
URL: http://dar01wwd.pjm.com/j2ee/servlet/com.pjm.xml.download.edart.EdartXMLDownload	▼ >>
Method Header Cookie Body Auth SSL Etc. Test	
String body	-
application/x-www-form-urlencoded; charset=UTF-8	
username=alexaa_super&password=****&downloadtype=transmission&transtype=revise&id=66747	
HTTP Response	
Status: HTTP/1.1 200 OK	
Headers Body Test Result	
xml version="1.0" encoding="UTF-8"?	•
<pre><edart xmins:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nonamespaceschemalocation="transticrevise.xsd"> </edart></pre>	
<pre>classicevise</pre> <pre>classicevise</pre> <pre>classicevise</pre>	
<pre></pre> description>testing projects/description>	
<project_ids>2,21,221</project_ids>	

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

🕲 WizTools.org RESTClient 3.5			
<u>Eile Edit History Tools H</u> elp			
HTTP Request			
URL: http://dar01wwd.pjm.com/j2ee/servlet/com.pjm.xml.download.edart.EdartXMLDownload	- >>		
Method Header Cookie Body Auth SSL Etc. Test			
String body	-		
application/x-www-form-urlencoded; charset=UTF-8			
username=alexaa_super&password=****&downloadtype=generation&id=305660			
HTTP Response			
Status: HTTP/1.1 200 OK			
Headers Body Test Result			
<pre><?xml version="1.0" encoding="UTF-8"?></pre>			
edart xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="genticreview.xsd">	=		
<pre><genticreview> <pym_ticket_id>305660</pym_ticket_id></genticreview></pre>			

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

3 WizTools.org RESTClient 3.5		and the second second	
<u>File Edit History Tools H</u> elp			
	HTTP Request		
URL: http://dar01wwd.pjm.com/j2ee/servlet/com.pjm.xml.upload.ee	dart.EDartXMLUpload		▼ >>
Method Header Cookie Body Auth SSL Etc.	Test		
	Test		
Multipart body			
Add Part: String File			
Type Content		Name	Part
String text/plain; charset=UTF-			lexaa 📃 🚍
String text/plain; charset=UTF-			****
File application/xml; charset	=UTF-8 file	C	:\Personal\work\etools\edart\testing\dev\
	HTTP Response		
Status: HTTP/1.1 200 OK			
Headers Body Test Result			
xml version='1.0' encoding='utf-8'?			j
<edartreply></edartreply>			
<pre><icket_info></icket_info></pre>			
<company_ticket_id>Adrian ticket 27</company_ticket_id>			
<pro><pro>Ficket di salready in our database or is duplicated in this XML submission.</pro></pro>			
<pro>licket rejected</pro>	×0		

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

WizTools.org RESTClient 3.5	And Index		
<u>File E</u> dit History T <u>o</u> ols <u>H</u> elp			
	HTT	P Request	
URL: http://dar01wwd.pjm.com/j2ee/servle	et/com.pjm.xml.upload.edart.EDartHydroUp	load	▼ >>
Method Header Cookie Body	Auth SSL Etc. Test		
Multipart body			•
Add Part: String File			
Туре	Content-type	Name	Part
String	text/plain; charset=UTF-8	password	*****
String	text/plain; charset=UTF-8	username	alexaa
File	application/vnd.ms-excel; charset=UTF-8	file	C:\Personal\work\etools\edart\testing\dev\
String	text/plain; charset=UTF-8	type	hydro 📃
3			
	HTTP	Response	
Status: HTTP/1.1 200 OK			
Headers Body Test Result			
<html></html>			^
<head></head>			
<ink href="/edatoss/edat_style.css" rel="stylesheet" type="text/css"></ink>			
<link href="/edartcss/button.css" rel="stylesheet" type="text/css"/>			
<pre>rmte>eDAR1 -injdio_upioad</pre> <pre>content="text/html; charset=iso-8859-1"></pre>			
META HTP-EQUV="PRAGMA" CONTENT="NO-CACHE">			
	the second se		
<body></body>			
<prf></prf>			

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