PJM Command Line Interface

PJM Interconnection LLC
Version 1.0
07-05-2012

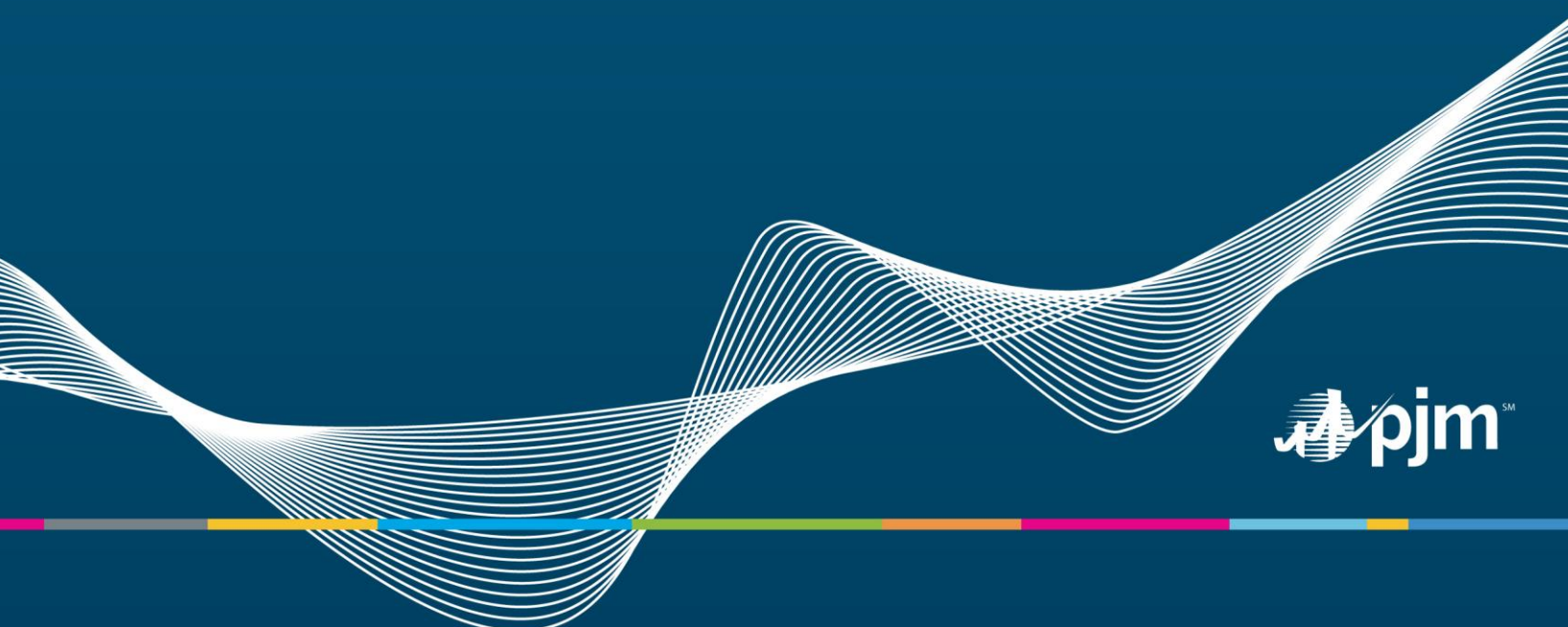

Table of Contents

Purpose

The purpose of this document is to give an overview of the command line interface (CLI), a Java based interface for transferring formatted files to and from PJM participant facing applications. Because the interface is "browser-less", it can be used by an end user or a custom automation program written by the participant.

PJM provides this specification to aid PJM customers in building an external interface to PJM applications. PJM will provide assistance to customers seeking to understand or clarify details in this specification. However, due to the customizable nature of this external interface and the varied environments in which PJM customers will implement it, PJM is unable to provide application support for these customer-built external interfaces.

System Requirements

All required software will either be provided by PJM or available for download from http://www.oracle.com or a vendor of your choice.

- A reliable internet connection
- Java Runtime Environment (JRE): Version 6 or higher is required. Java is available for Solaris, Linux, Mac OSX, AIX, and Windows. Java downloads are available at: http://www.oracle.com
- PJM provided Java CLI program: client side class to authenticate with PJM and transfer files. The current version is provided as a Java Archive (JAR) file.

Release History

Release	Date	Description
1.0	July 3rd, 2012	Initial Release including Gas Pipeline and eSchedules.

Usage

The Command Line Interface can be used in one of two ways: either as a standalone program or through the API. The following section describes the two different usage options.

Standalone Application

The CLI can be run as a standalone application from the command line by passing arguments to the application.

Example Standalone Execution

```
C:\Personal\tools\esuite-cli>java -jar esuite-cli.jar

PJM Command Line Interface 1.0
Copyright 2011-2012, PJM Interconnection LLC.  All rights reserved
Java: Java(TM) SE Runtime Environment 1.6.0_31-b05 Sun Microsystems Inc.
OS: Windows Vista 6.0 x86

ERROR: MissingOptionException: Missing required options: u, p, s, a, d

usage: esuite-cli

 -a,--action <String>          Action to execute
 -d,--directory <Directory>    Directory location of destination output
                               (required)
 -f,--file <File>              File location of upload file (required if
                               this is an upload operation)
 -i,--idpUrl <URL>             PJM Identity Provider Single-Sign On URL
                               (optional for public URLs)
 -l,--logging <Level>          Log level of output TRACE, DEBUG, INFO,
                               WARN, ERROR (default INFO)
 -o,--output <File>            Output result file or downloaded file
                               (optional - will use name sent from
                               application)
 -p,--password <String>        Password (either encrypted or clear text)
 -q,--query <param=value>      Query parameters to append to URL (optional)
 -r,--realm <realm>            Realm for which HTTP Basic Auth credentials
                               apply (only valid for BASIC AuthType)
 -s,--serviceUrl <URL>         PJM Application URL
 -t,--authType <type>          Authentication Type either BASIC or FORM
                               (default to FORM)
 -u,--username <String>        Username
 -x,--proxy <param=value>      Proxy parameters for internet proxy in form
```

```
                                of proxyPort=8000 for proxyHost, proxyPort,
                                proxyUser, proxyPassword (optional)
 -z,--timeout <arg>             Socket and connect timeout in milliseconds,
                                defaults to 60000 (optional)
```

As you can see the CLI is self documenting and will alert you if you are missing any required parameters.

Parameter Details

The table below describes the parameters in detail.

Argument	Description	Required/Optional
-u,--username	PJM account username from CAM.	**Required**
-p,--password	PJM account password from CAM. Password may be encrypted to prevent having clear text in any files. If encrypted must be surrounded by ENC(..). See section "Password Encryption".	**Required**
-i,--idpUrl	Identity provider URL also known as Single Sign On (SSO) URL. This URL is what authenticates you as a valid PJM user.	Optional for public service URL and required for secured public URL
-s,--serviceUrl	Service URL is the application URL you want to upload or download from. Example: `https://gaspipe.pjm.com/gaspipe/`	**Required**
-a,--action	The action to execute which will be a partial URL appended onto the service URL. This will be either an upload or download action. Example: `/rest/public/download/csv/notifications`	**Required**
-q,--query	Query parameters to append to the URL. These are documented later and are specific for each application. They are in the form of <param=value>	Optional (depends on the application and which query params it may accept)
-d,--directory	Directory location of destination output files. These may be results from an upload or the contents of a download operation.	**Required**
-f,--file	File location of upload file	Optional (required if performing an upload operation)

-o,--output	Output result file or downloaded file. The default will be the file name sent from the application such as "companies_2012-06-29-152315.csv". This parameter overrides that name with whatever you pass in like "companies.txt" would always force the name to companies.txt.	Optional
-l,--logging	Log level of output TRACE, DEBUG, INFO,WARN, ERROR (default INFO)	Optional (defaults to INFO)
-t,--authType	Authentication Type either BASIC or FORM (default to FORM)	Optional (default to FORM)
-r,--realm	Realm for which HTTP Basic Auth credentials apply (only valid for BASIC AuthType)	Optional (only applies to BASIC authentication)
-x,--proxy	Proxy parameters for internet proxy. If your company internet access requires a proxy enter it here. Some proxies only require a URL and port number while some are secured as well and require a username and password. Example: `-x proxyHost=myHost -x proxyPort=9000 -x proxyUser=myProxyUser -x proxyPassword=myProxyPass`	Optional (use only if your company uses a proxy)
-z,--timeout	Sets the socket and connection timeouts. Defaults to 60 seconds however for some large files you may need to increase this timeout.	Optional (defaults to 60000 milliseconds)

Password Encryption

Many participants choose to store their credentials in a file when calling the CLI. PJM provides example setEnv.cmd file for storing all of the common parameters of the CLI so a participant can then create custom CMD files that extend it. However, as a good security practice it is never recommended to store a password in plain text in a file. The CLI comes with a password encryption tool and associated CMD file for encrypting your password. You may then use this encrypted password in the password parameter of the CLI.

Example Password Encryption

Using the password.cmd command file included with the CLI distribution.

```
C:\Personal\tools\esuite-cli>password pjm123

PJM CLI Password Encryptor 1.0
Copyright 2011-2012, PJM Interconnection LLC.  All rights reserved
```

```
Java: Java(TM) SE Runtime Environment 1.6.0_30-b12 Sun Microsystems Inc.
OS: Windows Vista 6.0 x86

Encrypting password 'pjm123'...
Encrypted password: ENC(zoCSGqDTGqueZjXlI3a4Rg==)
```

In this example the password "pjm123" was encrypted to "ENC(zoCSGqDTGqueZjXlI3a4Rg==)". The entire string including the ENC() must be passed to the password parameter for the CLI to realize the string is encrypted and needs to be decrypted.

Command Files (.cmd)

PJM includes DOS command files with the CLI release. These CMD files are executable in a Windows Command Prompt and allow for easier configuration of the file operations. All re-usable parameters are stored in setEnv.cmd and then that CMD file is reference in all of the custom CMD files per application. The command files are documented in the text of each file and all of the possible application calls will be included as examples.

API – Application Programming Interface

An application programming interface (API) is a specification intended to be used as an interface by software components to communicate with each other. If a participant would like to embed code directly in their Java application rather than calling the external CLI program, they can use the API in their application. This requires the developer to implement their own Java program and use the provided behaviors. The API is designed to uses Streams for input and output. Downloads come in the form of a java.io.OutputStream allowing you to control the stream yourself using a FileOutputStream, ByteArrayOutputStream, or any OutputStream implementation. Uploads are sent in the form of java.io.InputStream so you can use any stream you want and not be constrained to using a file on disk if you already have the data in memory.

The example can be found in the CLI distribution in the location /docs/api/ExampleApi.java.

The API parameters mirror the CLI parameters listed in the section "Parameter Details".

Download Example API Usage

```
   /**
    * Example call to download a file from PJM into a ByteArrayOutputStream in
    * memory if you did not want to write the results to disk.
    */
public static void downloadToMemory() {
   try {
      final PjmRemoteCommand command = new PjmRemoteCommand();

      // first set up SSO and credentials
      command.setIdpUrl("http://sso.pjm.com/sso/");
```

```
      command.setServiceUrl("http://inschedule.pjm.com/inschedule/");
      command.setUsername("pjmuser");
      command.setPassword("Pjm4567!");

      // the URL action you want to execute
      command.setAction("/rest/secure/download/csv/contracts");

      // any query parameters required by the download
      command.getQueryParams().put("start", "01-01-2012");
      command.getQueryParams().put("stop", "01-31-2012");

      // create an output stream to capture the output result
      final ByteArrayOutputStream baos = new ByteArrayOutputStream();
      command.setOutputStream(baos);

      // execute the command and inspect the results
      final PjmResult result = PjmClient.execute(command);

      if (result.getResultCode() == PjmResult.CLI_SUCCESS) {
         System.out.println("Download succeeded for file: " + result.getFileName());
      } else {
         System.out.println("Download failed for file: " + result.getFileName());
      }

      // now you can inspect or manipulate the Outputstream results
      final String output = new String(baos.toByteArray());
      System.out.println(output);
   } catch (PjmClientException ex) {
      ex.printStackTrace();
   }
}
```

Upload Example API Usage

```
/**
  * Uploads a file from disk to the PJM Servers.
 */
public static void upload() {
   try {
      final PjmRemoteCommand command = new PjmRemoteCommand();
      command.setIdpUrl("http://sso.pjm.com/sso/");
      command.setServiceUrl("http://inschedule.pjm.com/inschedule/");
      command.setUsername("pjmuser");
      command.setPassword("Pjm4567!");

      // the URL action you want to execute
      command.setAction("/rest/secure/upload/file/");

      // a file upload requires an InputStream. It can be from a File like
      // this example, an in memory stream, or any other way of producing an
      // InputStream. The UploadFileName is purely a name for the server to
      // receive for logging purposes
      final File file = new File("my-upload.txt");
      command.setUploadFileName(file.getName());
```

```
      command.setInputStream(new FileInputStream(file));

      // create an output stream to capture the output result
      final ByteArrayOutputStream baos = new ByteArrayOutputStream();
      command.setOutputStream(baos);

      // execute the command and inspect the results
      final PjmResult result = PjmClient.execute(command);

      if (result.getResultCode() == PjmResult.CLI_SUCCESS) {
         System.out.println("Download succeeded for file: " + result.getFileName());
      } else {
         System.out.println("Download failed for file: " + result.getFileName());
      }

      // now you can inspect or manipulate the Outputstream results
      final String output = new String(baos.toByteArray());
      System.out.println(output);

   } catch (PjmClientException ex) {
      ex.printStackTrace();
   }
}
```

Supported Applications

As PJM refreshes each application in its portfolio they will be designed to use this Command Line Interface. The goal of this CLI is to have "convention over configuration". This means that each application will handle file uploads and downloads using the same convention allowing this CLI to support current and future applications without needing to be released every time a new application is released.

Each application will be listed below as well as what Actions that application exposes to the CLI.

Gas Pipeline

Type	Action	Parameters
Download	`/rest/public/download/csv/notifications` Downloads a CSV file of all current critical notifications.	None

InSchedule

Type	Action	Parameters

Upload	/rest/secure/upload/file/ Uploads a flat file type of either internal schedules or reconciliations.	File to upload
Upload	/rest/secure/upload/xml/ Uploads an XML file type of either internal schedules or reconciliations.	File to upload
Download	/rest/secure/download/csv/contracts Downloads a CSV file of contracts. Can choose a date range or pending only if you only want pending contracts	pending=true/false start=01-01-2012 stop=01-31-2012
Download	/rest/secure/download/csv/schedules Downloads a CSV file of internal schedules. Can choose a date range or pending only if you only want pending schedules	pending=true/false start=01-01-2012 stop=01-31-2012
Download	/rest/secure/download/csv/companies Downloads a CSV file of Company static data. This is a complete list of active companies in the system. (includeHeader will put the header at top of file making it invalid CSV)	includeHeader=true/false
Download	/rest/secure/download/csv/reconciliations Downloads a CSV file of Reconciliation data for a date range. (includeHeader will put the header at top of file making it invalid CSV)	includeHeader=true/false start=01-01-2012 stop=01-31-2012
Download	/rest/public/download/csv/edclossfactor Downloads a CSV file of EDC Loss Factor data for a date range. (includeHeader will put the header at top of file making it invalid CSV)	includeHeader=true/false start=01-01-2012 stop=01-31-2012