



Scripting and REST interface Guide

Version 5.9.0, 5.8.0

THE WORD "EXPERIAN" AND THE GRAPHICAL DEVICE ARE TRADEMARKS OF EXPERIAN AND REGISTERED IN THE EU, USA AND OTHER COUNTRIES.

THIS DOCUMENT CONTAINS INFORMATION, PROPRIETARY TO EXPERIAN DATA QUALITY, WHICH IS PROTECTED BY INTERNATIONAL COPYRIGHT LAW. THE INFORMATION CONTAINED HEREIN MAY NOT BE DISCLOSED TO THIRD PARTIES, COPIED OR DUPLICATED, IN WHOLE OR IN PART, WITHOUT THE PRIOR WRITTEN CONSENT OF COPYRIGHT OWNER. PLEASE CONTACT EXPERIAN TO FOR ANY CONSENT ENQUIRIES.

(C) 2007 – 2018 EXPERIAN DATA QUALITY

Preface

This document is the starting point for anyone wishing to automate functionality or integrate both into and out of Experian Pandora. For further help concerning this document or the use of Experian Pandora, please contact your support representative. This document is very technical in nature and assumes programming knowledge by the reader.

Defining and Executing a script from the UI

Experian Pandora supports powerful customisation and bespoke integration functionality via the scripting engines it provides. It allows users to write their own code in an environment that has access to all the functionality and data within the Server (security notwithstanding) in order to do any task imaginable.

The Experian Pandora Server supports scripting using the following Script Engines:

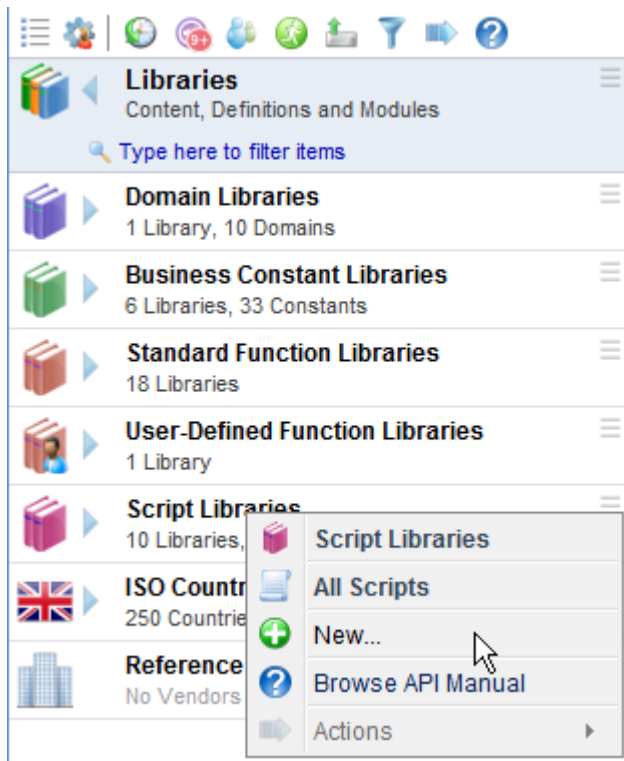
- JavaScript – Oracle’s Nashorn engine (V1.8.0_91)
- Groovy – Groovy V2.4.3
- Scala – www.scala-lang.org V2.11.7
- Ruby- jRuby engine V1.7.22
- R – org.renjin V0.7.1577
- Python – jython engine

The Experian Pandora UI provides dialogs to create, debug and run scripts against particular objects or over the entire repository.

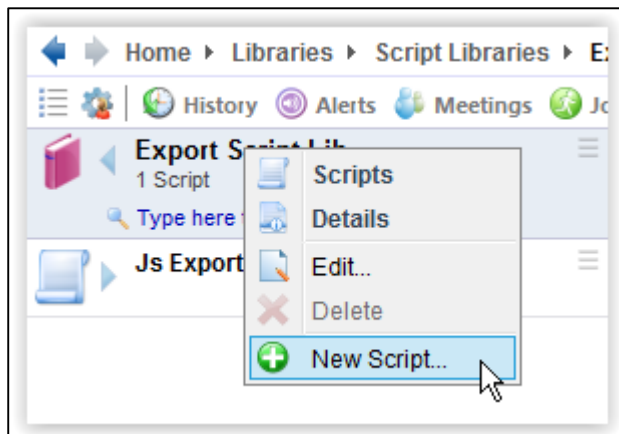
Creating a script in the UI

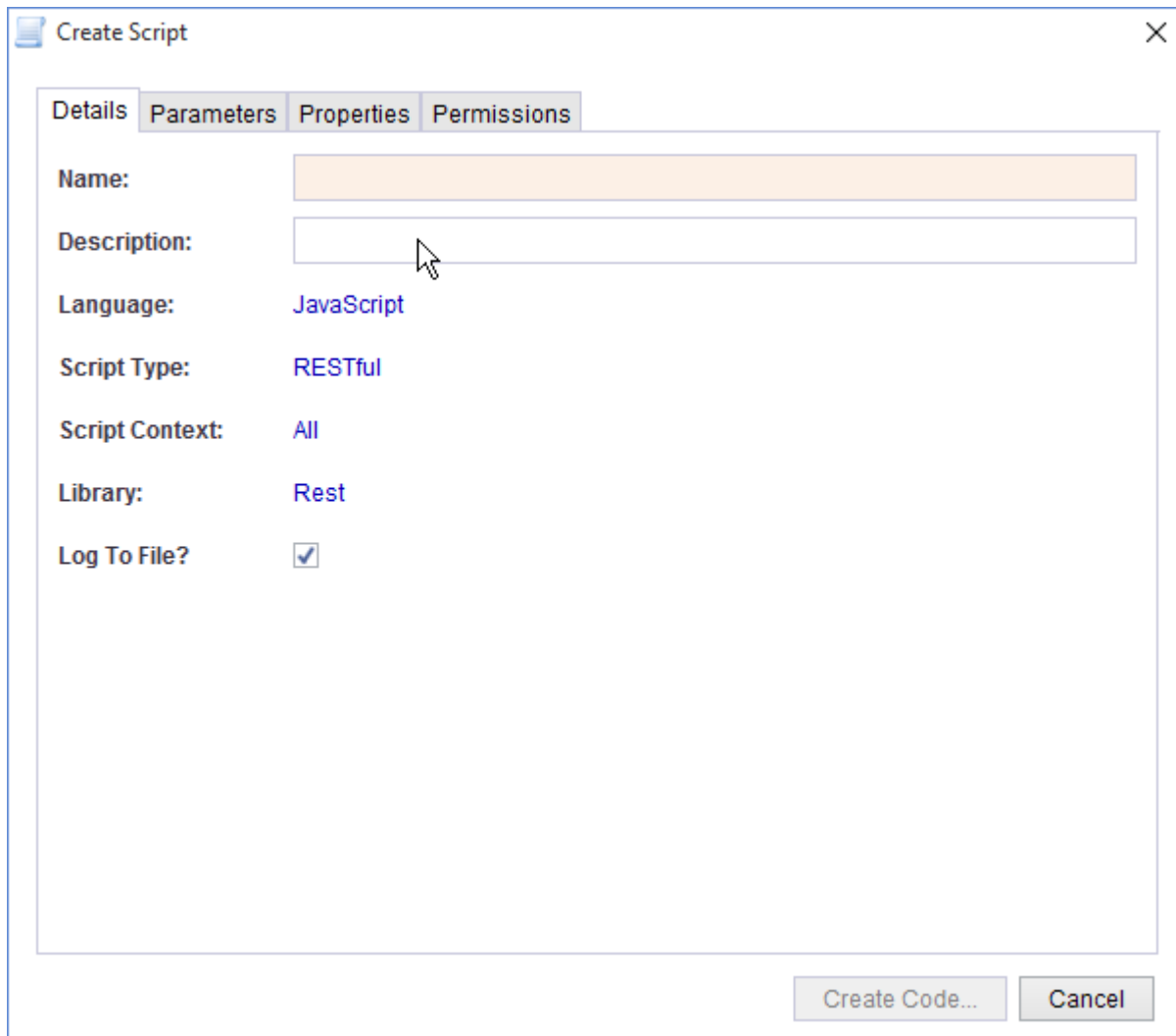
To add a script you first need to add a Script Library. Script libraries provide you with the means of grouping functionally identical scripts together. For example you may wish to create a Library for Export scripts and another Library for Alert scripts.

In the Navigator panel select: **Home > Libraries > Script Libraries** and right-click to select **New**



Next, open the Script Libraries item, right-click on your new library and select 'New Script...', and the following dialog will be displayed:





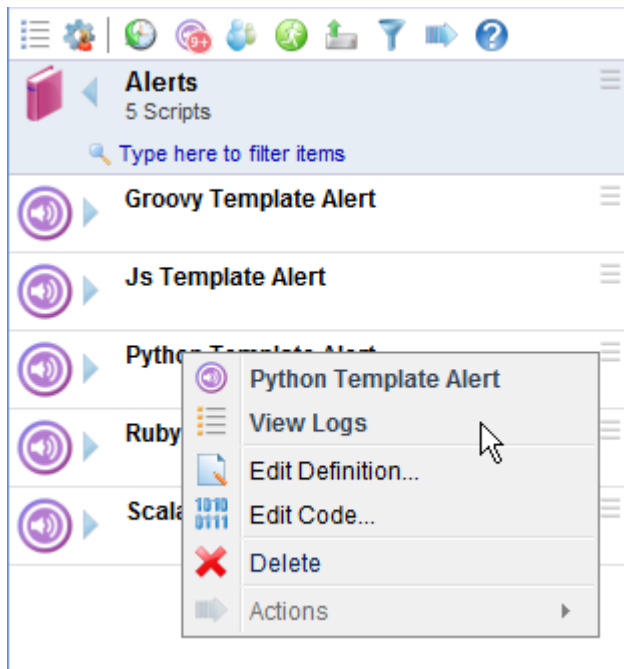
The 'Create Script' dialog box features a title bar with a close button (X) and a tabbed interface with four tabs: 'Details', 'Parameters', 'Properties', and 'Permissions'. The 'Details' tab is active, showing the following fields:

- Name:** A text input field with an orange background.
- Description:** A text input field with a mouse cursor hovering over it.
- Language:** A dropdown menu with 'JavaScript' selected.
- Script Type:** A dropdown menu with 'RESTful' selected.
- Script Context:** A dropdown menu with 'All' selected.
- Library:** A dropdown menu with 'Rest' selected.
- Log To File?** A checkbox that is checked.

At the bottom right of the dialog are two buttons: 'Create Code...' and 'Cancel'.

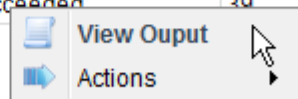
This dialog allows you to select a name and description for your script, the language you wish to use, the type of the script, the context in which it should appear and whether it should produce a log file (a redirection of the script's stdout and stderr).

If you tick "Log to File" all script output will be written to a file in the servers' data/scriptoutput directory. The file may be viewed by navigating to the Script Library that contains the script and right-click on the script name, selecting **View Logs**. In the example dialog this is **Home > Libraries > Script Libraries > Alerts > Python Template Alert**



Right Click on the log file you wish to view and Select **View Output**. Note that viewing script logs is subject to permissions checks.

Logs for Python Template Alert			
Last Updated on 16 October 2015 14:16:21 BST			
	Start Time	Status	Row count
1	12-Oct-2015 13:52:33	Failed	55
2	12-Oct-2015 13:57:31	Failed	55
3	12-Oct-2015 14:01:26	Failed	55
4	12-Oct-2015 14:34:17	Succeeded	29
5	12-Oct-2015 14:40:41	Succeeded	29
6	16-Oct-2015 12:16:48	Failed	62
7	16-Oct-2015 12:27:29	Failed	62
8	16-Oct-2015 12:29:29	Failed	62
9	16-Oct-2015 12:31:09	Succeeded	39
10	16-Oct-2015 12:33:06	Succeeded	39
11	16-Oct-2015 13:01:51	Succeeded	39



The contents of the log file will be displayed as a drilldown, and can be exported using the standard tools if necessary.

Output for Python Template Alert	
Last Updated on 16 October 2015 13:01:51 BST	
	
39 Rows	
1	Start script execution at 16-Oct-2015 13:01:51.821
2	
3	Reserved Script Parameters:
4	alertId = "2152"
5	scriptId = "2551"
6	restUrl = "http://169.254.151.188:7900/"
7	responseRequired = "true"
8	acknowledgeUrl = "http://169.254.151.188:7900/alerts/2152?action=ACKNOWLEDGE&acknowledger_id=2551"
9	osFlavour = "windows"
10	scriptActionType = "ALERT"
11	scriptContextType = "NONE"
12	scriptName = "Python Template Alert"
13	scriptLanguage = "PYTHON"
14	username = "Administrator"
15	
16	User defined Script Parameters:
17	No user defined parameters
18	
19	Global Script Parameters:
20	psession: defined
21	pobjects: defined
22	parameters: defined
23	pobjects[0] = "" (org.json.JSONObject)
24	pobjects[1] = "" (null)
25	
26	Script Output:
27	running the PYTHON ALERT script "Python Template Alert" on behalf of Administrator ...
28	pobjects has 2 elements
29	pobjects[0] class: org.json.JSONObject
30	pobjects[1] class: null
31	alert = {"requires_acknowledgement":"true","is_response_required":"true","object_type":"User","created":"Fri O
32	alertObject = None
33	acknowledgeUrl = http://169.254.151.188:7900/alerts/2152?action=ACKNOWLEDGE&acknowledger_id=2551
34	responseRequired = True
35	title = " User Login Failure "
36	description = " User Login Failure "
37	HTML file is C:\Users\Graham\AppData\Local\Temp\ack_2662337529882314622.html\n
38	
39	End script execution at 16-Oct-2015 13:01:51.951

The log file is split into four distinct sections.

Reserved Script Parameters are those parameters that Pandora provides as Global variables to your script. These will depend on the context of your script.

User Defined Script Parameters are key-value pairs that the user may have defined in either the parameters tab (in the script edit dialog) when you create or edit a script, or Runtime parameters (added when a script is run interactively in the client)

Global Script Parameters are Global variables that are always available to the script (although their values may be null). The PSession object provides access to the internal API. PObjects is an ArrayList that will contain context dependent Objects. Parameters is a Map containing key-value pairs defined by the User or Script.

Script Output will be any output generated by the script using the script languages' print statements. Note that both R and Groovy script engines currently don't support redirection of script output, so any printable output in these language will appear in the server log.

Script Languages

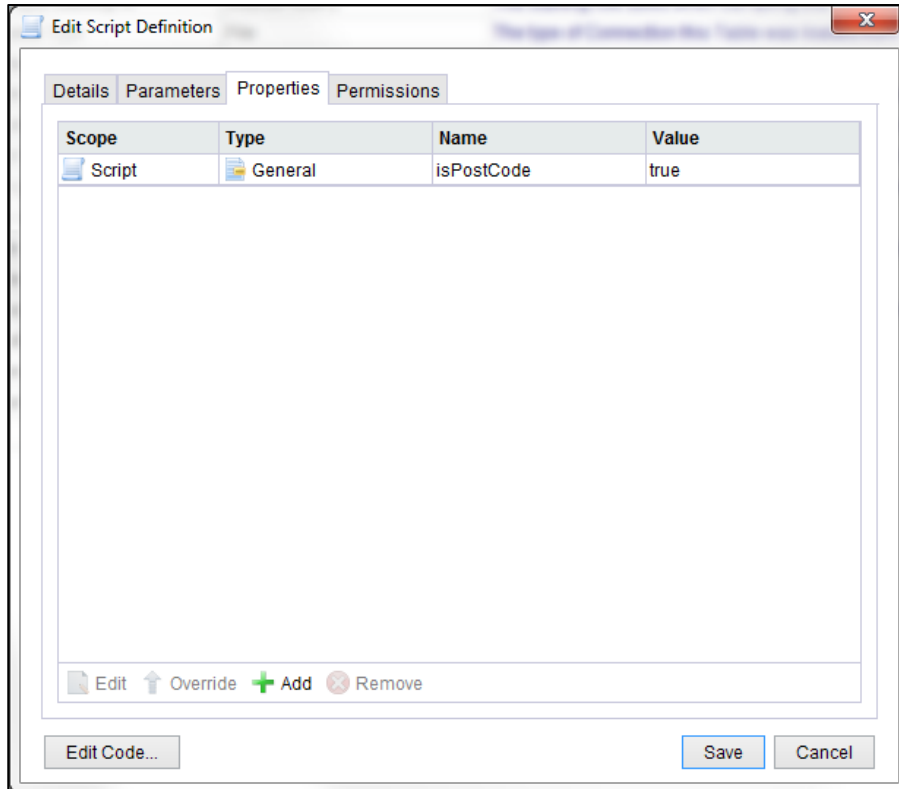
The following Script languages are supported:

- JavaScript
- Groovy
- Scala
- Ruby
- R
- Python

Script Properties

The script properties tab allows you to define named properties for the script.

If no properties have been set, the script will run against any object belonging to the selected context group. However, if a property has been defined, the contextual Repository Object (or a parent of it) must have the same property defined or the script will not be available.



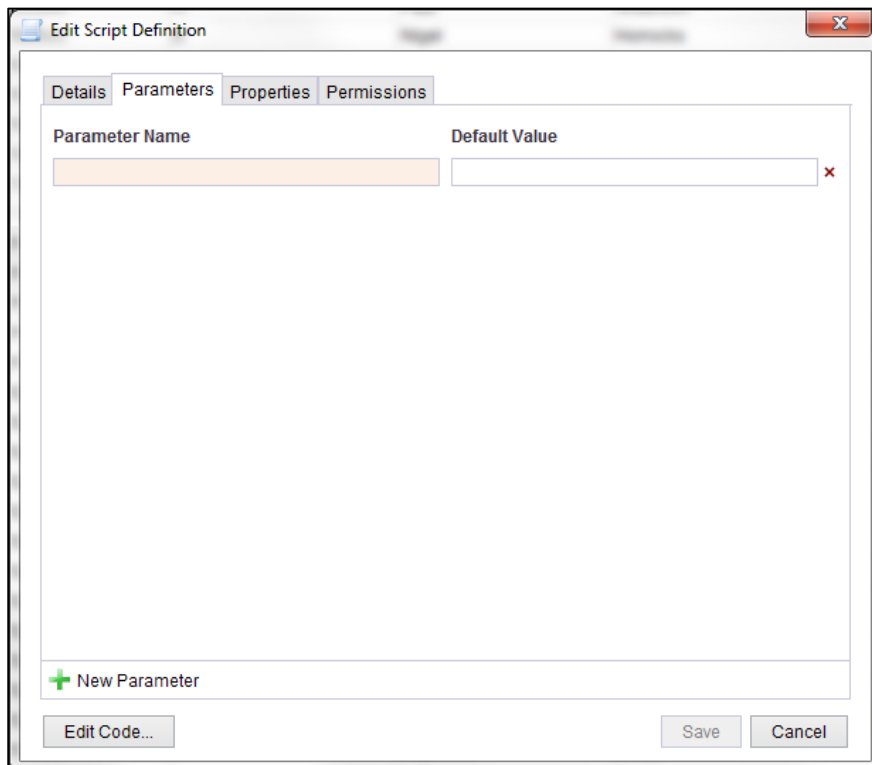
This functionality is to restrict certain scripts so they can only execute against objects that are marked similarly. For example, if your script can only cope with postcode columns, you can get a property of the script 'isPostCode'. When you load data, you can set this property on postcode columns and the script will run on these columns and these only.

Script Parameters

The dialog also allows you to define parameters for your script, along with their default values. In some cases, when you execute a script from the UI it will prompt for the parameters you have defined in this dialog allowing users to override them at runtime.

The parameters defined by this dialog will be available to the script in the **parameters** Global Variable. This is essentially a Map object (TypedPropertySet) of key-value pairs. Values may be obtained using any of the following method calls (e.g. if (parameters.contains("key")) {...})

Return Type	Method
Boolean	contains(key)
TypedProperty	get(key)
Boolean	getBoolean(key)
Double	getDecimal(key)
Long	getNumber(key)
String	getString(key)
JSONObject	getJSONObject(key)
Boolean	isEmpty()
Integer	size()
String	getPropertiesString(key)
Iterator<TypedProperty>	Iterator()



Script Types

The script type defines how the script will behave and where it can be used.

Script Type	Description
Server	General purpose script that runs on the server. The script runs asynchronously on the server and does not produce any UI visible output.
Alert	Executed when an alert has been triggered. Only visible in Alerts dialogs.
Export	Create a server job to export data. All output will be directed to the output file (if selected).
RESTful	Execute RESTful calls in a web UI window. When executed, a dialog will appear which allows you to choose the output type (html, XML, JSON), whether to pivot the data, and to enter any parameters specified before running the script in a web window within the UI.
Client UI	Script that runs in a Java UI window. When executed a Java UI container will appear and the script will be executed within it. The window will allow you to see errors in the script, console output from the script, and allow you to modify and rerun the script. The window also allows the script to add custom UI components to its toolbar.
Batch Script	UI available script that runs as a job. If any parameters have been defined, they will be prompted for before the script runs.
Client Module	Client-side script available to call from other scripts, but not directly.
Server Module	Server-side script available to call from other scripts, but not directly.
Save As	Script that may be run on a Table allowing access to each row/column/cell in the table. The output will be another Table. Rows/columns may be added/deleted/reordered and cell data modified

Script types are explained in more detail later in this document.

Script Context

This allows you to define the context for the Script. This is the type of Repository Objects that this script is applicable to in Experian Pandora. If an Object (e.g. a Table) has any scripts that have been denoted as being associated with that Object and the user has the Execute permission for the scripts, then a Scripts menu item will be visible that will allow you to select the relevant script and run it immediately.

The available contexts are listed in the following table:

Script Context	Description
None	Do not display in UI.
All	Display on all listed UI objects.
Global	Display on global Action button on the toolbar. There will be no context for scripts executed from this button.
Selection	The script will be accessible from the right-click menu on a drilldown list view. The script will be provided a JSON object (pobjects array) that contains the data that has been selected on the view.
Column	Limit the script to Column objects.
Dependency	Limit the script to Dependency objects.
Domain	Limit the script to Domain objects.
Relationship	Limit the script to Relationship objects.
Key	Limit the script to Key objects.
Note	Limit the script to Note objects.
Note Entry	Limit the script to Note Entry objects.
Saved Drilldown	Limit the script to Saved Drilldown objects.
Table	Limit the script to Table objects.
View	Limit the script to View objects.
None	The context is not relevant
All	Applicable to all Repository objects

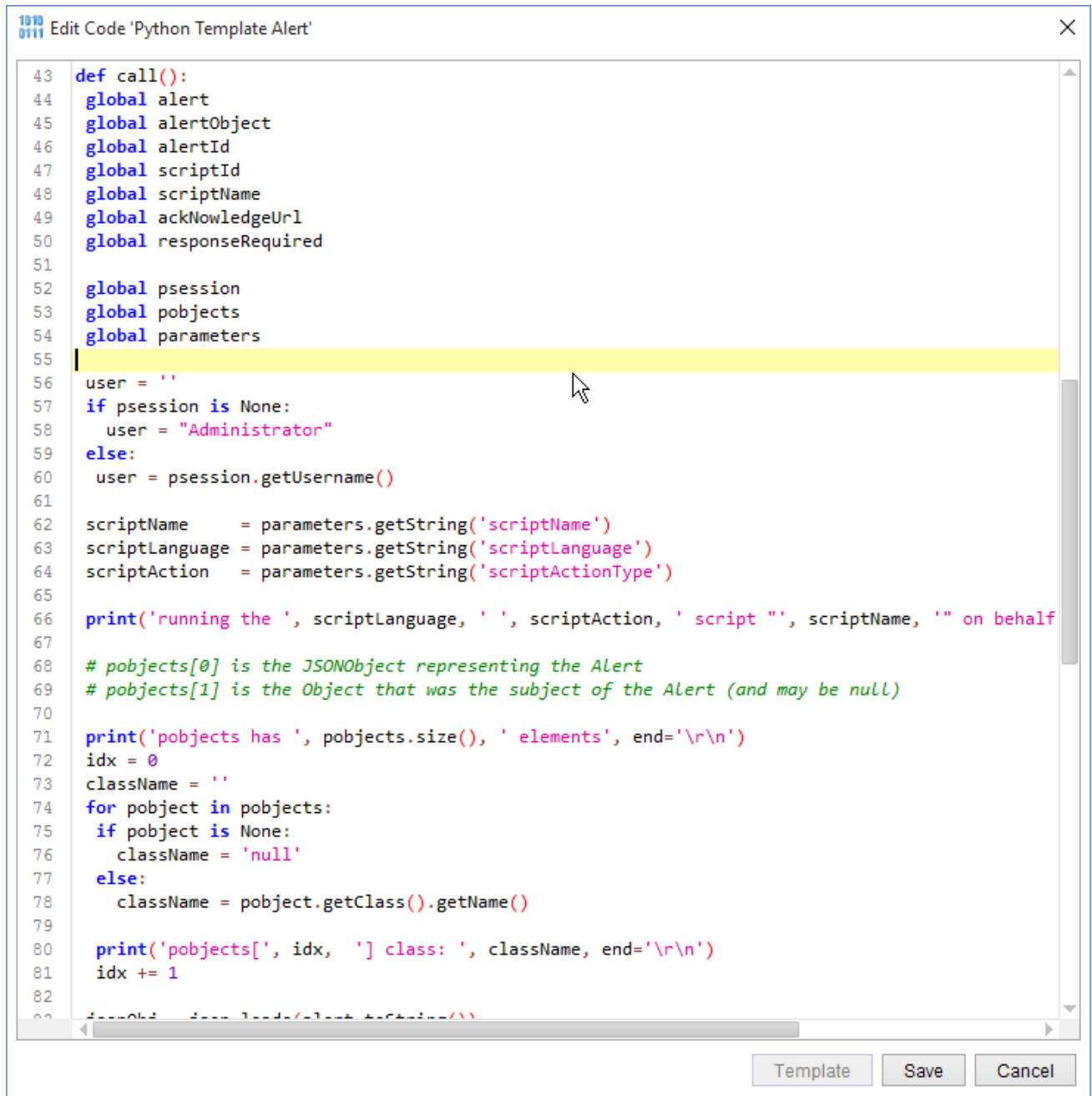
All Script Types (other than Alert scripts) have access to the Pandora API. Scripts may only be executed if the User has the relevant permission (Execute). A more detailed discussion of each script type is given below.

Examples are available in the distribution's examples directory for all supported languages and types. These are provided purely for illustration purposes. Bear in mind that the OS facilities available to the script will depend on the architecture the Experian Pandora server is running on.

Code Editor

Once you have defined your script, click the “Save” or “Edit Code...” buttons to display the script editor. The dialog is also available directly from the script context menu in the Explorer view, and the Java UI container window.

If you have already defined a script it will be displayed here and can be edited in-place. If the Template button is enabled you may use it to populate the editor with a Template example appropriate to the script action/context/language.



```

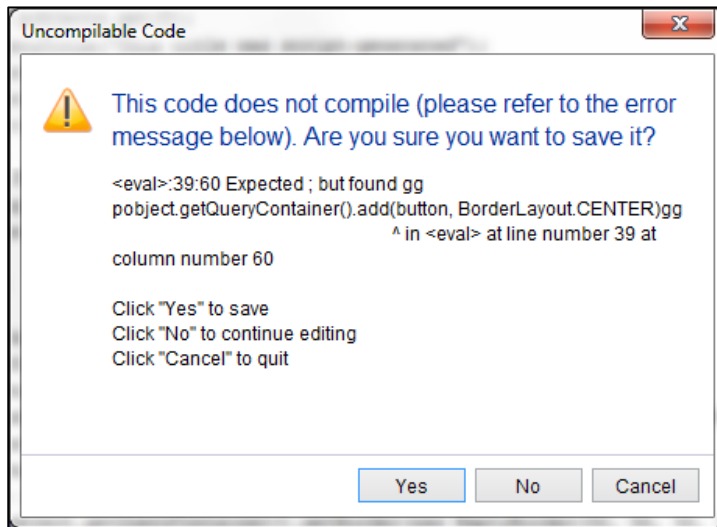
43 def call():
44     global alert
45     global alertObject
46     global alertId
47     global scriptId
48     global scriptName
49     global acknowledgeUrl
50     global responseRequired
51
52     global psession
53     global pobjects
54     global parameters
55
56     user = ''
57     if psession is None:
58         user = "Administrator"
59     else:
60         user = psession.getUsername()
61
62     scriptName = parameters.getString('scriptName')
63     scriptLanguage = parameters.getString('scriptLanguage')
64     scriptAction = parameters.getString('scriptActionType')
65
66     print('running the ', scriptLanguage, ' ', scriptAction, ' script ', scriptName, ' on behalf
67
68     # pobjects[0] is the JSONObject representing the Alert
69     # pobjects[1] is the Object that was the subject of the Alert (and may be null)
70
71     print('pobjects has ', pobjects.size(), ' elements', end='\r\n')
72     idx = 0
73     className = ''
74     for pobject in pobjects:
75         if pobject is None:
76             className = 'null'
77         else:
78             className = pobject.getClass().getName()
79
80     print('pobjects[', idx, ' ] class: ', className, end='\r\n')
81     idx += 1
82
83     from java.lang import Object

```

Template Save Cancel

Enter your code into this window and click the Save button to close the dialog and save your script.

When the Save button is pressed, your script will be checked for errors. If errors are found, a popup window will be displayed containing the details, and you will be given the opportunity to return to your code to fix the error, save your code anyway, or quit without saving.



Executing a script from the UI

Scripts can be executed by right-clicking on a valid object and selecting the relevant script under the Actions menu item.

The exact results will vary depending upon the script type, and the script itself.

Executing a script from the command line

Experian Pandora provides a batch client for executing scripts to automate functionality externally from the client application. Much of what is possible with the client is possible with the script interface. The script execution program is network aware so scripts can run anywhere and do not have to execute on the server host. The script interface is accessible using the **PExecute** command in the client software distribution. Arguments are case-insensitive. The following command line arguments are available:-

Argument	Description
Hostname	This is an optional parameter denoting the server hostname to connect to. This defaults to localhost .
Username	The Experian Pandora username to login as. This is mandatory.
Password	The password to use which can either be in the clear or encrypted. The default is encrypted and this parameter is mandatory.
Port	The port number to connect to on the server. This is mandatory. Port numbers are in the range 1 to 65535.
Encrypted	This is a flag to denote whether the password supplied is encrypted or not. Default is true .
Script	This is the filename of the script to execute.
Language	This is the language the script file is written in. If this flag is not provided, Experian Pandora will determine the language from the file extension.
Headings	When executing an SQL script, this flag denotes whether headings should be outputted. The default is true . This parameter is optional.
Style	This is the output style to use when executing an SQL script. Options are CSV , JSON and XML for the appropriate output formats.

The **PExecute** program returns one of a number of result codes depending on the result of the script. To view these result codes, you can call **PExecute -help result_codes**

If you run **PExecute** with any arguments that are not listed above, these arguments and their values will be passed through into the script and can be accessed using the **getArgument(String name, Object defaultValue)** method.

Script Types

Experian Pandora supports a number of different types of script.

Alert Scripts

Alert Scripts are associated solely with the Alerts mechanism. Only scripts with an Alert Script Type may be used for Alerts. Users may forward Alerts to recipients that may be either a User or a Script. The Script will be supplied with the Inputs:

Alert script Global Variables

Key	Value
psession	A PSession object providing access to the API (or null)
pobjects	An ArrayList of contextual Objects
parameters	A TypedPropertySet (Map) of properties (key-value pairs)
alert	A JSONObject - The JSON representation of the Alert
alertObject	A JSONObject - The JSON representation of the Subject (if any) of the Alert (e.g. a Table)
alertId	The alert's id (Long)
scriptId	The Script's Id (Long)
scriptName	The name of the Script
restUrl	The URL of the server's RESTful interface
responseRequired	True if an acknowledgement is expected
acknowledgeUrl	The URL that may be used to acknowledge this alert

An Alert Script **must** provide the following functions:

- Void call()
- Boolean shouldWait()
- String getResultText()

If the Script doesn't provide all 3 of these functions an error will be reported and the script will not be executed. The Script should not include any executable code outside of these functions. When the Script is executed the server will call the "call()" function. If an acknowledgement is required the server will call the "shouldWait()" function. If this function returns True then the server will wait for an acknowledgement, otherwise it will continue processing other recipients. The wait is subject to the acknowledgement Timeout associated with either the containing Alert Group or the Script recipient. If the Script experiences an error whilst processing the Alert then the server will call the function "getResultText()" to obtain the reason for the error.

Alert Scripts are executed in the context of the Pandora Server so due attention should be paid to security considerations and OS support provided by the scripting language.

Because alert scripts are not actioned explicitly by a user, their output is written to the Administrator's log file.

Export Scripts

Export Scripts provide the means for custom exports. They may be selected in the “**Save As > Target > Script**” or “**Save As > File > Script**” dialog for a Table or Drilldown. Select a script Library and then a Script from the menu items. Only Scripts that have an Export Script Type will appear in these menus/ dialog.

“**Save As -> File**” will return the file (if any) to the Client’s machine. “**Save As > Target**” saves the output file (if any) to the server’s data/export directory.

Export script Global Variables

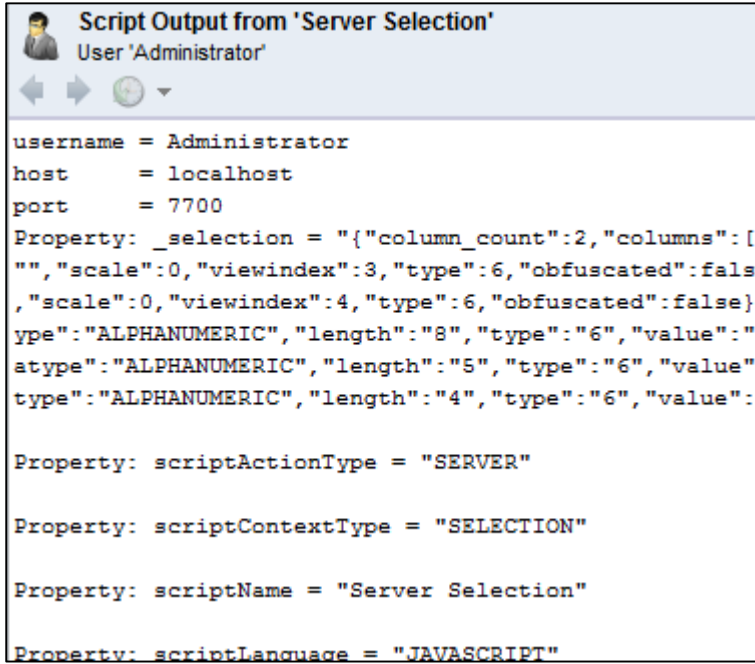
Key	Value
psession	An API Object that provides public function calls to the Pandora server (see below for a complete list of API function calls)
pobjects	- An ArrayList of contextual Objects - pobjects[0] is a PScriptQuery object object that provides the following function calls - long getRowCount() - int getColumnCount() - PDataCell getCell(long row, int column) - String getColumnName(int columnIndex) - void setProgress(int percentProgress)
parameters	A TypedPropertySet (Map) of properties (key-value pairs)
pout	A FileOutputStream Object or null If the user provided a File Name in the “ Save As ” dialog then this OutputStream may be used for writing column values to an output file.

Export scripts execute in the context of the client Username and are subject to Permissions and Capability checks.

Server Scripts

These scripts are general purpose server-side scripts that can run in the context of a Repository Object or run globally with no context.

They run asynchronously on the Server and any output generated by the script will be displayed asynchronously in a console window:



```
Script Output from 'Server Selection'
User 'Administrator'

username = Administrator
host      = localhost
port      = 7700
Property: _selection = "{"column_count":2,"columns":[
  "", "scale":0, "viewindex":3, "type":6, "obfuscated":false
  , "scale":0, "viewindex":4, "type":6, "obfuscated":false}
  type":"ALPHANUMERIC", "length":"8", "type":"6", "value":"
  atype":"ALPHANUMERIC", "length":"5", "type":"6", "value"
  type":"ALPHANUMERIC", "length":"4", "type":"6", "value":

Property: scriptActionType = "SERVER"

Property: scriptContextType = "SELECTION"

Property: scriptName = "Server Selection"

Property: scriptLanguage = "JAVASCRIPT"
```

If the window is closed, the script will carry on running until it finishes.

Batch Scripts

Batch scripts are run as Jobs by the Scheduler. They are subject to permissions and capability checks on the user that submits the job.

The following global variables are defined for a Web script:

- `psession` – An API Object that provides public function calls to the Pandora server (see below for a complete list of API function calls)
- `pobjects[]` – The `pobjects` array contains the context objects that were selected (if any) when the script was executed. Each object is one of the following types (depending on the menu item context):
 - Table
 - Column
 - Join
 - Dependency
 - Domain
 - Key
 - Note
 - Note Detail
 - Saved Query
 - View Table
- `scriptProperties` – A Map Object that provides the following property key-value pairs
 - Script parameters – parameters associated with the script

RESTful Scripts

Users may execute scripts via the RESTful interface. The general format of the URL is:

```
http://servername:restfulport/object/object_type/object_name/resource_type/resource_name.output_format?username=admin&password=admin
```

For running the script named TEST on a Table named CUSTOMER:

```
http://localhost:7900/object/TABLE/CUSTOMER/SCRIPT/TEST.xml?username=admin&password=admin
```

The URL is broken down as:-

- localhost The DNS name or IP address of the Pandora server
- 7900 The RESTful port
 - see **Home > Settings > Server Settings > Communication > REST Web Server TCP/IP Port**
- /object Request an Object from the Repository
- /TABLE The target repository object type (a PObject)
- /CUSTOMER The target repository object name (In this example the Table name)
- /SCRIPT The Resource Type (a Script)
- /TEST.xml The Name of the Script (TEST) and its output format (.xml)
- ?username The user's login name
- &password The user's password

Note that the Script Name that you specify must have a Script Context Type of "Web"

The following global variables are defined for a Web script:

- psession – An API Object that provides public function calls to the Pandora server (see below for a complete list of API function calls)
- pobjects[] – The pobjects array contains the context objects that were selected (if any) when the script was executed. Each object is one of the following types (depending on the menu item context):
 - Table
 - Column
 - Join
 - Dependency
 - Domain
 - Key
 - Note
 - Note Detail
 - Saved Query
 - View Table
- scriptProperties – A Map Object that provides the following property key-value pairs
 - Script parameters – parameters associated with the script

Client UI Scripts

Client Scripts allow you to run a script within a Java UI window on the client.

The following global variables are defined for a Client UI script:

- `psession` – An API Object that provides public function calls to the Pandora server (see below for a complete list of API function calls)
- `pobjects[0]` – This object is the container window. It is used to add UI components to the window.
- `pobjects[1]`, `pobjects[N]` – The remaining objects in the `pobjects` array contain the context objects that were selected when the script was executed. Each object is one of the following types (depending on the menu item context):
 - Table
 - Column
 - Join
 - Dependency
 - Domain
 - Key
 - Note
 - Note Detail
 - Saved Query
 - View Table
- `scriptProperties` – A Map Object that provides the following property key-value pairs
 - Script parameters – parameters associated with the script

Client Module

The Client Module is used to define scripts that run synchronously on the client and can be invoked by other scripts.

Client Modules return information by implementing the `getResult` function. For example to return the variable 'text' to the calling script, implement the following in your Client Module:

```
var text = "Server Script Invoked";
function getResult() {
    return text;
}
```

To invoke a Client Module script and obtain its returned value, you first need to standardise the script name, find the script object within the repository, and once found invoke it and get its return value. For example, to execute a script called "Client Script" you need to implement the following:

```
importClass(com.experian.utility.string.StringUtil);
importClass(com.experian.api.PObjectType);

// Call Client Module called "Client Script"
var scriptName = "Client Script";
scriptName = StringUtil.standardize(scriptName);
scriptObj = psession.findObject(PObjectType.SCRIPT, scriptName);
if (scriptObj != null) {
    var retval = psession.invokeClientModule(scriptObj, "args");
    if (retval != null) {
        print("Client Module returned: " + retval);
    }
}
```

The script object can be obtained by calling the `psession.findObject` method. The name of the script needs to be standardised in order to successfully find it.

The Client Module can then be invoked with any argument by passing the args as (ideally a JSON structure) the second parameter to the `psession.invokeScript` method.

Server Module

The Server Module type can be used to define scripts that run as a job on the server that can be invoked by a different script. The Server Method script will execute on the server synchronously, so is capable of returning information to the calling script.

Like Client Modules, Server Modules return data to the invoking script by implementing the `getResult` function. See the Client Module section above for an example.

Once you have defined a Server Module script, you can invoke it from another script using the following (JavaScript) code:

```
scriptName = "Server Script";
scriptName = StringUtil.standardize(scriptName);
scriptObj = psession.findObject(PObjectType.SCRIPT, scriptName);
if (scriptObj != null) {
    var retval = psession.invokeServerModule(scriptObj, "My Args");
    if (retval != null) {
        print("Server Module returned: " + retval);
    }
}
```

The script object can be obtained by calling the `psession.findObject` method. The name of the script needs to be standardised in order to successfully find it.

The Server Module can then be invoked with any argument by passing the args as (ideally a JSON structure) the second parameter to the `psession.invokeScript` method.

An example Server Module script that returns some text with the argument value appended to them:

```
var text = "Server Script Invoked with params: ";
function getResult() {
    return text + parameters.getString("args");
}
```

Save As

The Save As type can be used to define scripts that may be used to iterate over a Table's rows/columns. You may:

- add rows
- delete rows
- reorder rows
- add columns
- delete columns
- reorder columns
- rename columns
- modify cell data

The resultant rows/columns may be output to a new table or a new version of the same table.

Global Variables

Key	Value
psession	A PSession object providing access to the API (or null)
pobjects	An ArrayList(1) - A PObject representing the Table
parameters	A TypedPropertySet (Map) of properties (key-value pairs) – see below

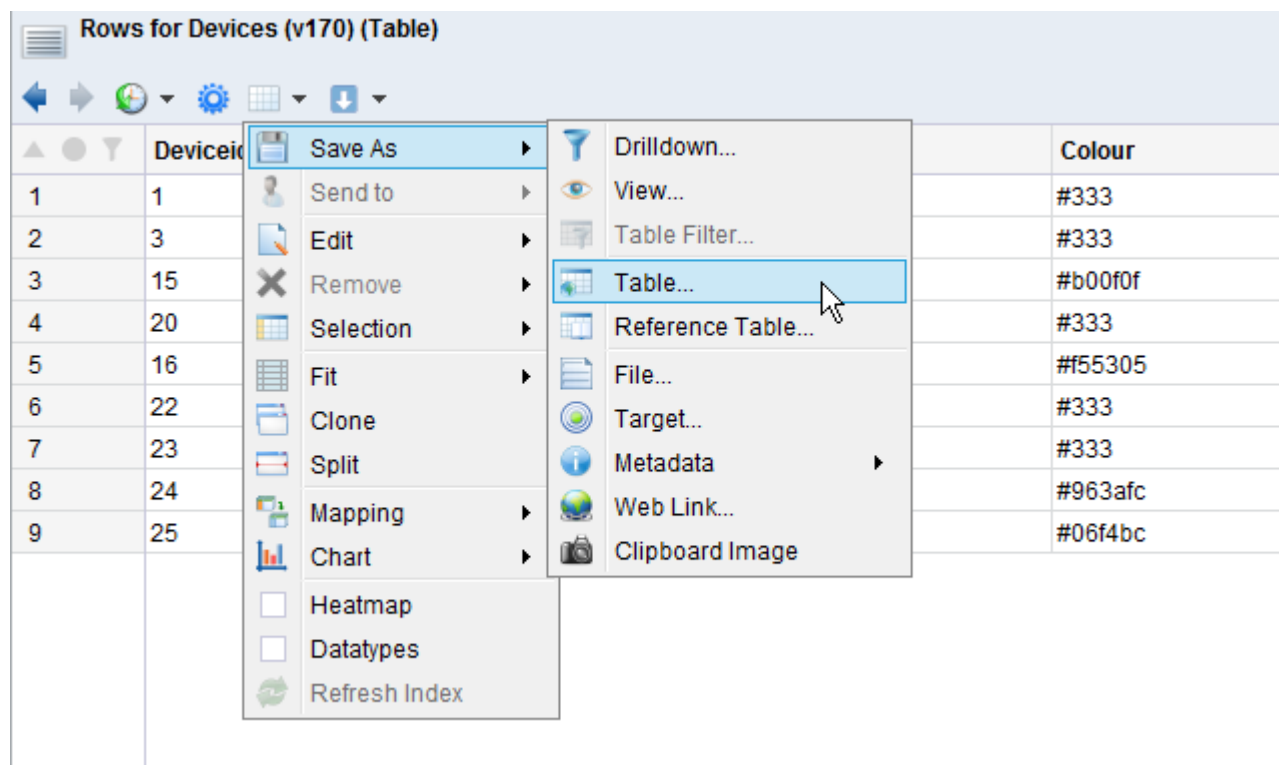
Properties (in the parameters Global Variable)

Key	Value
jobData	A PJob object representing the job that was scheduled to execute the table load
+ Script properties	Any script parameters inherited from the Script definition and any parameters defined in the Table Load definition

Since the output is another table, users should provide a method called `getMappingDocument()`. This should return a `UserDocument` that describes what transformations the script made to the data. When a user requests a Mapping Report for a table that was created by a script, the Mapping code will call this script function and embed the `UserDocument` in the Mapping Report in the Script section. The template (example) code for scripts with the Save As action provide an example of this function.

To execute a scripted save-as function navigate to **Home > Data > Tables > MyTable > Rows** and then select **Save As > Table**

Rows for Devices (v170) (Table)



	DeviceID	Colour
1	1	#333
2	3	#333
3	15	#b00f0f
4	20	#333
5	16	#f55305
6	22	#333
7	23	#333
8	24	#963afc
9	25	#06f4bc

You may then select the appropriate script that will be used to load the table.

Save As Table

Details

Permissions

Load as:

Devices

Description:

Default Schema:

Default

Default Project:

<None>

Ignore empty Tables?

☐

Storage Type:

Compressed and Indexed

Scheduled Load Time:

Now

Exclude from Relationships?

☐

Script:

<None>

<None>

Test Additional Params

Test Reload With Params

Test Renamed Column

Scratch

Test Reordered Column

Test Deleted Column

Test Additional Rows

 This will load as a new version of **Devices v170**

Columns...

OK

Cancel

Script Languages

Experian Pandora currently supports the JavaScript, Groovy, Ruby, Scala, R and Python languages. Although this documentation will concentrate on the JavaScript language, the objects and methods described are transferable to the other languages. Also, example scripts will be provided for all languages as examples of how to integrate them with Experian Pandora.

JavaScript

JavaScript scripts use Oracle's Nashorn engine. They can utilise any java class that is delivered with the product to reference Repository objects and methods, or Java Swing classes to display, configure and use UI components.

Pandora will prefix your scripts with `load("nashorn:mozilla_compat.js");`
All the Nashorn JavaScript extensions are available for use.

For example, to use JavaScript to find a repository object (in this case a script called "My Script") you need to run the following code:

```
importClass(com.experian.utility.string.StringUtil);
importClass(com.experian.api.PObjectType);
scriptName = "Server Script";
scriptName = StringUtil.standardize(scriptName);
scriptObj = psession.findObject(PObjectType.SCRIPT, scriptName);
```

In order to add UI Components to a Client UI Window, you need to import the relevant swing classes. For example to add a button to a Client UI window, and handle actions when pressed you need to run the following code:

```
importClass(javax.swing.JButton);
importClass(java.awt.BorderLayout);
importClass(java.awt.event.ActionListener);

pobject = pobjects[0];

myBtn = new JButton("My Button");

var buttonPressed = new ActionListener({
    actionPerformed: function(e) {
        print("Button Pressed");
    }
});

myBtn.addActionListener(buttonPressed);
pobject.getQueryContainer().add(myBtn, BorderLayout.SOUTH);
pobject.getQueryContainer().revalidate();
pobject.getQueryContainer().repaint();
```

If you need to load external script libraries then use the "load" statement.

Script libraries are loaded by default from the (server's) current working directory if the name is not qualified. E.g.

```
load("XMLWriter-1.0.0.js"); // load from the working directory
load("file:///c:/pandora/XMLWriter-1.0.0.js"); // load from an explicit directory
load("http://localhost:7900/web/js/XMLWriter-1.0.0.js"); // get the library from root/web/
```

Python

The server supports Python 3, so pre-v3 'print' statements are not supported. If you need to print to console (for example) the new print() function should be used instead.

All scripts will be prefixed with `"from __future__ import print_function";`

The following Python example script obtains the current user, queries the selected context object (pobject[0]) and tries to obtain any user-defined parameters associated with the script:

```
global pobjects
global psession
global parameters

print ('Python script executed by ', psession.getUsername())

# Get the PScriptQuery object and list it's properties
pobject = pobjects.get(0)
ostring = pobject.toString()
print ('object = ', ostring)

# Get the table name parameter
tableName = parameters.getString('TableName')
print ('TableName = ', tableName)
```

The output of this script can be seen in the script log file which can be obtained by running it, then right-clicking on the script icon and selecting the "View Output" menu option.

Groovy

Scripts can also be written in Groovy. Groovy scripts have access to the same global objects as the other scripting languages. Note that logging to file or console doesn't work due to a bug in Groovy.

As an example of a Groovy script, this obtains the current user, queries the selected context object (pobject[0]) and tries to obtain any user-defined parameters associated with the script:

```
String username = psession.getUsername()
println "Groovy script executed by $username"

// Get the table name parameter
String tableName = parameters.getString("TableName")
```

```
println "Table Name: $tableName"

// Get context object properties
String props = pobjects.get(0).toString()
println "Object Properties: $props"
```

The output of this script can be seen in the script log file which can be obtained by running it, then right-clicking on the script icon and selecting the “View Output” menu option.

Ruby

Ruby scripts can also be used, and they contain the same global objects as the other scripting languages.

Pandora will prefix your scripts with `"include Java";`

For example, this script obtains the current user, queries the selected context object (pobject[0]) and tries to obtain any user-defined parameters associated with the script:

```
puts ('Ruby script executed by ' + $psession.getUsername())

# Get the table name parameter
puts ('Table Name: ' + $parameters.getString('TableName'))

# Get context object properties
puts ('Object Properties: ' + $pobjects.get(0).toString())
```

The output of this script can be seen in the script log file which can be obtained by running it, then right-clicking on the script icon and selecting the “View Output” menu option.

R

R scripts can also be used, and they contain the same global objects as the other scripting languages. The R script engine (Renjin) does not support redirection of the stdout/stderr. All Script output will be written to the server log file. If you wish to include other R plugins, the jar files should be placed in the server's data directory **root/external/jar** (root is the representation of the server **property Home > Settings > Server Settings > Storage > Data**)

You may obtain these plugins from <http://packages.renjin.org/>

Scala

Scala scripts can also be used, and they contain the same global objects as the other scripting languages.

Currently there are no example scripts (Templates) for Scala

RESTful

It is also possible to execute a script using the REST API. In order to do this, the script must be placed in the /scripts folder on the server. It can be accessed by calling /script/*scriptName*. The script will be passed the following arguments:

Argument	Description
pobjects[]	An array of PObjects on which the script was run. If the script was executed against one item it can be obtained using pobjects[0].
resourceType	The type of resource that was requested. This will be one of pdf , html , txt , text , json , or xml .

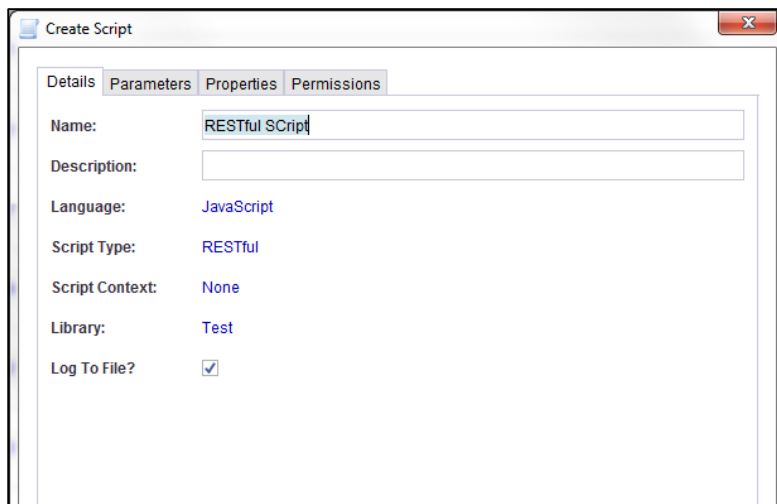
Using this method, scripts can easily be used to display custom documents over the REST API. The script must call the **setReturnValue(Object returnValue)** method which can take one of the following:

- An array of bytes (byte[]) which is data already formatted for return by the REST server
- A **JSONObject** which is a JSON object of key/value pairs (can be hierarchical) which will be automatically formatted according to the **resourceType**.
- A **PDocument** which will be formatted into the appropriate return type (if possible), e.g. **pdf**

Adding Scripts as New REST Resources

Scripts can be used as new resources that can be accessed using the REST API.

To do this, create a script library in Experian Pandora, then add a new script to it. In the Create Script dialog the script type should be set to be RESTful, and the Script Context should be set to limit the objects that the script can be executed on in the UI. Note that the Script Context only applies to the UI, and does not limit the use of the script from the restful engine.



The screenshot shows the 'Create Script' dialog box with the following details:

- Name:** RESTful Script
- Description:** (empty field)
- Language:** JavaScript
- Script Type:** RESTful
- Script Context:** None
- Library:** Test
- Log To File?** ☒

In the Create Code edit window, enter the script that is required. To define the data that is returned you must implement a setReturnValue function and return the value in there. For example, this script returns the object it was invoked against and the resource type requested.

```
setReturnValue(  
    "The object you called this with was " +  
    getArgument("object", null).getProperty(PPProperty.DISPLAY_NAME) +  
    " and the type of request was " +  
    getArgument("resourceType", ResourceType.XML)  
);
```

The url that will call this script is defined as:

```
http://localhost:7900/object/TABLE/CUSTOMER/SCRIPT/TEST.xml?username=my_username&password=my_password
```

The black parts define installation specific items that should be replaced with your servername, http port, username and password. The bold parts highlight the table being accessed in the script.

The script will receive two arguments:

1. **pobject[0]** which is the repository object specified in the URL
2. **resourceType** which is the output type requested – XML in this case.

RESTful Object Types

Object types defined by /object are the same as those defined by the PObjectType enumerations:

- TABLE
- BUSINESS_TERM
- DATASOURCE
- DOMAIN_LIBRARY
- BUSINESS_CONSTANT_LIBRARY
- FUNCTION_LIBRARY
- SCRIPT_LIBRARY
- NOTE_CATEGORY
- NOTE_IMPACT
- USER_GROUP
- USER_ROLE
- PROJECT
- SCHEMA
- BUSINESS_CATEGORY
- DOMAIN
- BUSINESS_CONSTANT
- USER
- NOTE
- NOTE_DETAIL
- VIEW
- QUERY
- DEPENDENCY
- KEY
- TABLE_VERSION_GROUP
- COLUMN
- RELATIONSHIP
- FUNCTION
- SCRIPT

RESTful Return Types

After the script has finished processing, it should return the resultset back to the REST engine. The result can be one of the following JAVA object types:

1. String

As per the example above, is a simple string. What happens here is that the REST engine simple formats the text into UTF-8 bytes and returns it. In the example, a resource type of XML is requested, but it returns a type that is self-formatted. It could have built up an XML document and returned that as a string which would be received correctly by the client.

2. byte[]

All webservers return content from requests as arrays of bytes (byte[]) usually in UTF-8 encoding. Here, the script could do the job of the web server and return a byte array, in which case it is simply passed back. This is useful if your script was, for example, accessing fixed resources (like pdf files) and simply streaming back their byte content.

3. JSONObject

The framework of the REST server is based on JSON objects (JavaScript Object Notation). These are simple objects made up of keys and values, but the values can themselves be objects (or lists of objects). If you return your result as a JSON Object, then the REST server will attempt to format sensible output automatically. Ask for JSON, you get what you created. Ask for XML, you will get the XML representation of that. Ask for PDF, HTML or TXT and you will get a tabular, flattened hierarchy of the JSON object you created – this is how most requests are handled by Experian Pandora.

4. DocumentElement

The DocumentElement framework is what is used to create Mapping documents, Quality documents, etc. and is built up of elements such as Sections, Tables, etc. You can create these directly and return a DocumentElement which will use the proper Document renderers for PDF, HTML and TXT.

5. PDocument

This is an API wrapper to quickly make documents for the DocumentElement framework. You can quickly convert data output in queries to document components that can be added to a PDocument to make a custom document easily. The output is a DocumentElement as per (4) above.

6. Webbable

This is an interface used in Experian Pandora for objects that support ability to output themselves as JSON objects. If it is Webbable, it has a toJSON() method. This is useful if you have some 3rd party classes that are created using high-level API calls, don't want to concern yourself with JSON creation in the script.

Common Scenarios

Context Objects

If the script is executed from the context of one or more objects in the UI it will be passed these objects in the pobjects array.

The script can query the pobjects array to find how many objects are being passed, and for each of them get their properties. The following script prints the properties for each context object passed:

```
for (var i=0; i<pobjects.size(); i++) {  
    pobject = pobjects[i];  
    print(pobject);  
}
```

Note that if the script is running in a Client UI window, the first pobject will be the window context that is used when adding UI components, the following pobjects are the context objects.

Getting a Context Object's data

Having been passed a context object (or obtained one from the repository through the API) you'll probably need to get its data. This simple example show how to get a context Table's columns and data.

```
// imports  
importClass(com.experian.api.PObjectType);  
importClass(com.experian.api.PObject);  
importClass(com.experian.api.PQueryType);  
importClass(com.experian.api.PProperty);  
importClass(com.experian.api.PDataRow);  
importClass(com.experian.api.PDataCell);  
importClass(com.experian.api.ColumnDefinition);  
  
// get context object (will start from 1 if Client UI)  
var pobject = pobjects.get(0);  
  
// get object type  
var otype = pobject.getProperty(PProperty.OBJECT_TYPE);  
  
// if type is Table  
if (otype.toString() === "Table") {  
    // create drilldown to rows  
    var drilldown = psession.createDrilldown(pobject, PQueryType.ROW)  
    // execute it  
    drilldown.run();  
  
    // get Column data  
    var columns = drilldown.getColumns();  
    columns.forEach(function(column) {  
        // print a list of column names  
        print ("COLUMN: " + column.getDisplayName());  
    });  
}
```

```
// get row count
var rowcount = drilldown.getRowCount();

// get row data for for 0
var orow = drilldown.getRow(0);
// can either get entire row as CSV
var csvRow = orow.toCsv();
// print each cell
while (orow.hasNext()) {
    var cell = orow.next();
    print("CELL: " + cell);
}
}
```

User-defined Parameters

The following javascript sample allows the script to obtain any parameters that have been defined in the script properties dialog, and possibly overridden during the script invocation:

```
var requiredParameter = "TableName";
value = parameters.getString(requiredParameter);
print(requiredParameter + " value: " + tableName);
```

Selection Data

If the script context is 'Selection' it is only available to run over user-selected cells of a drilldown view.

In this case the script will need to obtain the selected data in order to process it. The following script gets the data and prints it out:

```
if (parameters.contains("_selection")) {
    print("Selected Data: " + parameters.getString("_selection"));
}
```

Loading a Table

The flow of this operation is as follows:

1. Get the correct datasource (PDataSource object)
2. Get the table definition for the thing you want to load (PDataSourceTable object)
3. Configure the PDataSourceTable object as necessary (e.g. change metadata of columns, names etc.)
4. OPTIONAL: Save your modified PDataSourceTable object back to the server-based XML configuration
5. Create a PLoadTableDefinition object and set the PDataSourceTable object on that
6. Schedule the load using the PDataSource and PLoadDefinition

Getting the Datasource

There are a few methods for this:

```
PDataSource getInternalDataSource(PDataSourceType type, PImportArea area)
```

This allows you to get one of the internal, file-based datasources, so 'type' must be DELIMITED_FILES or EXCEL_FILES. 'area' is PUBLIC, GROUP or PRIVATE.

```
List<PDataSource> getAllDataSources()
```

Returns a list of all available datasources.

```
List<PDataSource> getDataSources(PDataSourceType type)
```

Returns a list of all available datasources. If 'type' is not null then it will filter the datasource list based on the supplied type.

```
PDataSource getDataSource(PObject table)
```

Returns the datasource that was originally used to load the specified table. This is required when, for example, you want to reload that table again (it's the same process that the GUI uses after you click "Load New Version...")

Getting the table definition

```
List<String> getDataSourceTableNames(PDataSource dataSource)
```

Returns a list of table names that are available to be loaded from the specified datasource. Names are case-sensitive and it is exactly these that should be used when getting the definition for a specific table (see below)

```
List<PDataSourceTable> getDataSourceTables(PDataSource dataSource)
```

As above but gets a list of the actual table definitions, rather than just their names. The name of a datasource table definition can be obtained by calling PDataSourceTable.getName().

```
PDataSourceTable getDataSourceTable(PDataSource dataSource, String tableName)
```

Get a table definition by name from the specified datasource (only useful in conjunction with getDataSourceTableNames)

```
PDataSourceTable getDataSourceTable(PObject table)
```

Attempts to get the current table definition that can be used to reload the specified table. Again, all part of the reload table process.

Modifying the table definition

The most important methods in `PDataSourceTable` are:

`getName()` / `setName(String)` - fairly obvious what these do

`List<PDataSourceColumn> getColumns()`

Returns a list of datasource columns. It is these that allow you to change properties on the columns. `PDataSourceColumn` also has `getName()/setName()`. Alongside that it also has `getColumnMetadata()`, which returns a `ColumnMetaData` object which can then be used to modify properties of the columns.

`deleteLastColumn()` / `addNewColumn()`

These are direct analogies of what happens in the GUI when you add/delete columns on the right-hand side in the source preview part of the loader wizard.

At this point, you can save this definition back to the corresponding XML in the import area by using

```
boolean saveTableConfiguration(PDataSourceType type, PDataSourceTable table)
```

Note: If you are going to load the table anyway, then it is not necessary to save the configuration first.

Loading

Before loading anything, you'll need to create a `PLoadTableDefinition` and then set the `PDataSourceTable` on it, for example:

Assuming you have a `PDataSource` in a variable called 'dataSource' and a `PDataSourceTable` in a variable called 'loadTable':

```
loadDef = new PLoadTableDefinition();  
loadDef.setTable(loadTable);
```

And then load it, for example:

```
scheduleLoad(dataSource, loadDef);
```

The `scheduleLoad` method returns an int which is 0 if something went wrong or it's the job id of the loader job that has been successfully submitted. To reload a table, you can either use `scheduleLoad` again, or you can use `scheduleReload`. For example:

```
scheduleReload(datasource, table, loadDef);
```

table is the table that should have been used to obtain the datasource and datasource table before calling `scheduleReload`.

Exporting a query

The flow of this is as follows:

1. Get the object (PObject)
2. Get the query to run (PQueryType)
3. Create a drilldown (PDrilldown)
4. Execute the drilldown
5. Create a PExportDefinition
6. Schedule the export using the executed drilldown and export definition
7. (OPTIONAL) Fetch the file back to the client

Get the object

There are a few methods which will help you do this:

```
PObject findObject(PObjectType type, String name);  
PObject findObject(PObjectType type, String name, int version);  
PObject findObject(PObjectType type, String name, PObject parent);
```

This allows you to find an object by its name. It also allows you to find a specific version of an object. The final signature allows you to find an object by parent. For example, you could have two tables with different names that both happen to have an email column, this method would allow you to get the correct column out.

```
List<PObject> getObjects(PObjectType type);  
List<PObject> getObjects(PObjectType type, SortOrder order);  
List<PObject> getObjects(PObjectType type, String pattern);  
List<PObject> getObjects(PObjectType type, String pattern, SortOrder order);
```

This allows you to obtain a list of objects of a specified type. You can also have this list sorted by the specified sort order, which can be **UNSORTED**, **ASCENDING**, or **DESCENDING**. You can also filter the list using a regular expression.

The following method can be handy for obtaining a PObjectType:

```
PObjectType getObjectTypeForString(String type);
```

Create and execute the Drilldown

The following methods can be used to create a drilldown:

```
PDrilldown createDrilldown(String query);
```

```
PDrilldown createDrilldown(PObject object, PQueryType type);
```

You can obtain a query string by calling the following method:

```
String getQueryForObject(PObject object, PQueryType type);
```

And you can obtain a list of supported query types by calling:

```
object.getAllowedQueryTypes();
```

Assuming you created a drilldown with one of the above methods called 'drilldown' you can execute the drilldown using either of the following methods:

```
drilldown.run();
```

This will execute the drilldown in the current thread.

```
drilldown.execute();
```

This will execute the drilldown in a new thread. You can call

```
drilldown.isBusy();
```

in order to check whether the drilldown has finished executing.

Note: before you do anything with a drilldown you need to execute it.

Create the PExportDefinition

There are two constructors for the PExportDefinition class:

```
PExportDefinition();
```

```
PExportDefinition(String serverFileName);
```

```
PExportDefinition(ExportType exportType, String serverFilename, String  
charsetName, boolean forceQuotes, char quoteCharacter,  
ExportColumnNameStyle columnNameStyle, PivotOption pivotOption);
```

The empty constructor creates an export definition for a **CSV** export, a server file name of **export**, a charset of **UTF-8**, not forcing quotes, **double quotes** as the quote character, **DISPLAY_NAME** export name style and **AUTOMATIC** pivoting.

The PExportDefinition class contains getters and setters for all these variables.

Schedule the export

There are two methods available to schedule an export:

```
scheduleExport(String query, String jobName, PExportDefinition definition);
```

```
scheduleExport(PDrilldown drilldown, String jobName, PExportDefinition definition);
```

These methods will return a 0 if they failed to schedule the export, or the job id if they succeeded.

Fetch the file back to the client

As the export saves the exported file to the export directory on the server, it may be useful to fetch the file to the client. In order to do this, you need to use the following method:

```
fetchFile(String localFile, String remoteFile, DirName dir, boolean deleteOnSuccess);
```

Exporting a report

The flow of this is as follows:

1. Get the object (PObject)
2. Get the Document to export
3. Schedule the export using the Document

Get the object

There are a few methods which will help you do this:

```
PObject findObject(PObjectType type, String name);
```

```
PObject findObject(PObjectType type, String name, int version);
```

```
PObject findObject(PObjectType type, String name, PObject parent);
```

This allows you to find an object by its name. It also allows you to find a specific version of an object. The final signature allows you to find an object by parent. For example, you could have two tables with different names that both happen to have an email column, this method would allow you to get the correct column out.

```
List<PObject> getObjects(PObjectType type);
```

```
List<PObject> getObjects(PObjectType type, SortOrder order);
```

```
List<PObject> getObjects(PObjectType type, String pattern);
```

```
List<PObject> getObjects(PObjectType type, String pattern, SortOrder order);
```

This allows you to obtain a list of objects of a specified type. You can also have this list sorted by the specified sort order, which can be **UNSORTED**, **ASCENDING**, or **DESCENDING**. You can also filter the list using a regular expression.

The following method can be handy for obtaining a PObjectType:

```
PObjectType getObjectTypeForString(String type);
```

Get the document

There are methods to get each of the report types. The following methods each generate PDocument objects:

```
PDocument getDashboard(PObject object);
```

This will create a validation dashboard for the given object. This will only work if it is possible to generate a validation report for the object.

```
PDocument getMappingReport(PObject object);
```

```
PDocument getMappingReport(String query);
```

```
PDocument getMappingReport(PDrilldown drilldown);
```

These methods generate a mapping report for the specified object, query or drilldown, respectively. The first method will only work on a table or a view.

```
PDocument getRelationshipReport(PObject relationship);
```

This will create a relationship report document for the given relationship object.

The following methods start off a job to create a PDocument object:

```
int scheduleNoteDocumentJob(PObject object, long time);
```

```
int scheduleNoteDocumentJob(PObject object);
```

These methods will create a document containing all of the notes on the specified object. The optional time variable takes in the milliseconds of the time the job should start.

```
int scheduleQualityReportJob(PObject object, long time);
```

```
int scheduleQualityReportJob(PObject object);
```

These methods will create a quality report for the specified object, only if the object allows quality reports. The optional time variable takes in the milliseconds of the time the job should start.

Both of these methods will return either 0, if it failed to schedule the job, or the job id if it was successful.

If you have used one of the methods which kick off a job, the following method allows you to get the resultant PDocument object:

```
PDocument getDocumentJobResult(int jobId);
```

Note: you will need to make sure the job has completed before attempting to retrieve the document

You can use the following method on an object to get a list of supported report types:

```
object.getAllowedReportTypes();
```

Export the document

The following two methods are supplied to export the document:

```
exportDocument(PDocument document, ExportType type, String filename,  
boolean clip, PivotOption pivotOption, boolean mkdirs);
```

```
exportDocument(PDocument document, ExportType type, String filename,  
boolean clip, PivotOption pivotOption);
```

As the documents rely on presentation and images, it is only possible to use the **PDF** or **HTML** export types to export them. The mkdirs option tells the server whether or not to try and create directories in the filename's path if they do not already exist, this defaults to true.

Saving and Unsaving Relationships

The flow of this is as follows:

1. Get the objects (PObject)
2. Discover relationships (List<PObject>)
3. Save or unsave a relationship

Get the objects

There are a few methods which will help you do this:

```
PObject findObject(PObjectType type, String name);
```

```
PObject findObject(PObjectType type, String name, int version);
```

```
PObject findObject(PObjectType type, String name, PObject parent);
```

This allows you to find an object by its name. It also allows you to find a specific version of an object. The final signature allows you to find an object by parent. For example, you could have two tables with different names that both happen to have an email column, this method would allow you to get the correct column out.

```
List<PObject> getObjects(PObjectType type);
```

```
List<PObject> getObjects(PObjectType type, SortOrder order);  
  
List<PObject> getObjects(PObjectType type, String pattern);  
  
List<PObject> getObjects(PObjectType type, String pattern, SortOrder  
order);
```

This allows you to obtain a list of objects of a specified type. You can also have this list sorted by the specified sort order, which can be **UNSORTED**, **ASCENDING**, or **DESCENDING**. You can also filter the list using a regular expression.

The following method can be handy for obtaining a PObjectType:

```
PObjectType getObjectTypeForString(String type);
```

Discover relationships

The following method can be used to discover relationships between the two objects:

```
List<PObject> findRelationships(PObject lhs, PObject rhs, int minQuality,  
int minDomainQuality, boolean searchOutside, boolean ignoreDiscovered,  
boolean ignoreSelfJoins);
```

lhs and **rhs** are the left-hand and right-hand-side objects that you have just found.

minQuality is the lowest quality of relationship to find, so all relationships returned should have quality equal to or above this minimum quality.

minDomainQuality is the minimum domain quality for the column-to-column relationship.

searchOutside denotes whether or not to allow searching outside the objects provided. For example, where one of the objects appears on at least one side of the relationship.

ignoreDiscovered denotes whether or not to ignore joins that have not been saved.

ignoreSelfJoins denotes whether or not to ignore joins between the same object.

This method will return a list of relationships that meet the given parameters, or null if it failed to find any.

Save or unsave a relationship

In order to save a relationship you should call the following method:

```
boolean saveRelationship(PObject relationship);
```

In order to unsave a relationship you should call the following method:

```
boolean unsaveRelationship(PObject relationship);
```

API Objects and Commands

This section will detail the objects that Experian Pandora makes available to the scripts, and how to use them.

Example scripts are provided in the installation. Please refer to the example scripts directory in your installation for details.

Customising the Client UI

If you have defined a Java UI script, the container window can be customised by the script to change the banner text, icon, and even add new buttons. The following commands can be used to achieve this:

addToolBarComponent

This code will allow a JavaScript script to add user-defined components (i.e. buttons) to the container window, and allow them to define behaviour for them (i.e. when pressed).

```
importClass (com.experian.explorer.IconType);
importClass (com.experian.ui.util.Util);
...
container = pobjects.get(0);
...
tbbutton = Util.createToolBarButton(IconType.EXPORT);
var al = new ActionListener({
    actionPerformed: function(e) {
        print("toolbar button pressed");
    }
});
tbbutton.addActionListener(al);
container.addToolBarComponent(tbbutton);
```

setQueryTitle

This function allows the script to set the container window's title text. For example:

```
pobject.setQueryTitle("This title was script-generated");
```

setQueryInfo

This function allows the script to set the container window's sub-title text. For example:

```
pobject.setQueryInfo("This info was script-generated");
```

setQueryIconType

This function allows the script to set the icon displayed in the container window. For example:

```
pobject.setQueryIconType(com.experian.explorer.IconType.EXPORT);
```

setBackground

This function allows the script to set the background colour of the container window's canvas. For example:

```
pobject.getQueryContainer().setBackground(Color.WHITE);
```

PSession

The primary object for API communication with the server is the **PSession** object. This is created for you whenever you use the **PExecute** command and is available as the variable **session**. Using **PExecute** the session is disconnected automatically for you.

The following is a list of methods:-

abort

void abort()

Exits the running Java virtual machine (e.g. the PExecute command) and return the result code for **PSessionResult.ABORT**

addTableToContainer

boolean addTableToContainer(PObject container, PObject table)

Adds a repository Table object to a container, which can be a **SCHEMA**, **TABLE_VERSION_GROUP** or a **PROJECT**. Returns **true** if successful, **false** if an error occurred.

closeDrilldown

boolean closeDrilldown(PDrilldown drilldown)

Closes an active drilldown. Returns **true** if successful, **false** if an error occurred.

convertFixedToCsv

boolean convertFixedToCsv(String fixedFilename, String csvFilename, List<Integer> lengths, String charset)

Converts fixed files to comma separated.

createDrilldown

PDrilldown createDrilldown (String query)

PDrilldown createDrilldown(PObject object, PQueryType type)

Creates a new drilldown (**PDrilldown**) by executing an SQL query. This can be a formulated query string, or a known query for a given object defined by the **PQueryType** enumeration.

Valid query types for any PObject can be found by calling the **getAllowedQueryTypes()** method of the PObject.

createProject

PObject createProject(String name, String description)

Creates a new **Project** in the repository. The new Project is returned or **null** if the creation failed. Note that a name change may occur if the name supplied is already in use in the repository.

createSchema

PObject createSchema(String name, String description)

Creates a new **Schema** in the repository. The new Schema is returned or **null** if the creation failed. Note that a name change may occur if the name supplied is already in use in the repository.

createUnionQuery

String createUnionQuery(Collection<PObject> objects);

String createUnionQuery(PObject... objects);

String createUnionQuery(PObject object);

Creates a dynamic union query between any number of **objects**

deleteJob

boolean deleteJob(int jobId)

Deletes a job from the scheduler given the sequence id of the job. Returns **true** if the deletion worked or **false** if it did not.

deleteObject

boolean deleteObject(PObject object)

Deletes the **object** provided from the repository. Returns **true** if the deletion worked, or **false** if it did not. **This cannot be used for deleting tables, to do that you should use scheduleTableDeletion. This also cannot be used to delete a table group.**

didJobComplete

boolean didJobComplete(int jobId)

Checks whether or not the job with id **jobId** completed. Returns **true** if it did or **false** if it did not.

disableMultipleVersions

boolean disableMultipleVersions(PObject object)

Disables multiple versions for the specified **object**. This object can only be a table version group. Returns **true** if multiple versions were successfully disabled and **false** if they were not.

disableUser

boolean disableUser(PUser user)

Disables the **user** provided. A user can be obtained by using findObject to get the PObject of the user and then creating a new PUser object either from the name or id of the PObject. Returns **true** if the user was successfully disabled and **false** if it was not.

enableMultipleVersions

boolean enableMultipleVersions(PObject object)

Enables multiple versions for the specified **object**. This object can only be a table. Returns **true** if multiple versions were successfully enabled and **false** if they were not.

enableUser

boolean enableUser(PUser user)

Enables the **user** provided. Returns **true** if the user was successfully enabled and **false** if it was not.

encryptPassword

static String encryptPassword(String password)

Encrypts a plain password. Returns the encrypted password.

exit

void exit()

Exits the current java virtual machine with the result code of the last operation.

exitOk

void exitOk()

Exits the current java virtual machine with the **OK** result code.

exportDocument

boolean exportDocument(PDocument document, ExportType type, String filename, boolean clip, PivotOption pivotOption)

boolean exportDocument(PDocument document, ExportType type, String filename, boolean clip, PivotOption pivotOption, boolean makedirs)

Exports the supplied **document** and puts the resulting file into the path defined by **filename**. If the filename has no path, the file will be put into the script's working directory. **type** can be either **PDF** or **HTML**. If the filename does not have the correct extension, it will be appended on to the end. **pivotOption** can either be **AUTOMATIC**, **YES**, or **NO**. The optional boolean **makedirs** defines whether the directory in the filename should be created if it does not exist, this defaults to true. Returns **true** if the export succeeded and **false** if it failed.

fetchFile

boolean fetchFile(String localfile, String remotefile, DirName dir, boolean deleteOnSuccess)

Moves a file, defined by **remotefile**, from the specified server directory, defined by the **dir**, to a file on the client, defined by **localfile**. If **deleteonsuccess** is set to true the file will be deleted if the movement is a success, if it is set to false it will not.

fileExists

boolean fileExists(String filename)

boolean fileExists(String logicalDirName, String filename)

Returns **true** if a server file exists, false if it does not. Either a full qualified path name can be used or a relative file in a server logical directory. Relevant logical directories are:

Directory Name	Description
TEMP	Temporary files
IMPORT	Public import files.
EXPORT	Exported files.
CONTENT	Domain content files.
METADATA	Metadata files.
DRIVERS_JDBC	JDBC driver files.

findColumn

PObject findColumn(PObject table, String name)

Finds the column on the given **table** with the **name** provided. Returns **null** the column was not found or if an error occurred.

findDefinitionForFile

PLoadTableDefinition findDefinitionForFile(PDataSource dataSource, String name)

Finds the table definition for the file with the specified **name** in the specified **dataSource**.

findObject

PObject findObject(PObjectType type, String name)

PObject findObject(PObjectType type, String name, int version)

PObject findObject(PObjectType type, String name, PObject parent)

Finds a repository object with a given **name**, with an optional **version** for versioned objects and an optional **parent** for objects that can have the . Returns **null** if not found or if an error occurred.

For example, if you wanted to find a Table called Mountains you would use the following code:

```
mountains = psession.findObject(PObjectType.TABLE, "Mountains");
```

If Mountains had 3 versions and you wanted to find version 2, you would use the following code:

```
Mountains2 = psession.findObject(PObjectType.TABLE, "Mountains", 2);
```

Alternatively, you could use the following:

```
Mountains2 = psession.findObject(PObjectType.TABLE, "Mountains:2");
```

Finally, if you have saved a view on table mountains called "Mountains in Wales" and you wanted to find that, you would use the following:

```
height = psession.findObject(PObjectType.VIEW, "Mountains_in_Wales",  
mountains);
```

assuming that you used the code above to get mountains.

findRelationships

List<PObject> findRelationships(PObject lhs, PObject rhs, int minQuality, int minDomainQuality, boolean searchOutside, boolean ignoreDiscovered, boolean ignoreSelfJoins)

Find relationships between two objects. These can be projects, schemas, table version groups, tables or columns. Returns a list of the relationships found.

getAllDataSources

List<PDataSource> getAllDataSources()

Returns a list of all the datasources.

getArgument

Object getArgument(String name, Object defaultValue)

Returns the value associated with the command line argument with the given **name**. If an argument with that name does not exist, then **defaultValue** is returned.

getDashboard

PDocument getDashboard(PObject object)

Gets a document containing the validation dashboard for the supplied **object**. It must be possible to validate the supplied object. Returns the document.

getDataSource

PDataSource getDataSource(PObject table)

Gets the datasource for the **table** supplied. Returns the datasource. This method is useful for reloading a table.

getDataSourceTable

PDataSourceTable getDataSourceTable(PDataSource dataSource, String tableName)

PDataSourceTable getDataSourceTable(PObject table)

Gets the PDataSourceTable from the **dataSource** specified, using the **tableName** provided or gets the PDataSourceTable for the **table** provided. Returns the PDataSourceTable.

getDataSourceTableNames

List<String> getDataSourceTableNames(PDataSource dataSource)

Gets the names of the tables within the specified **dataSource** and returns a list of them.

getDataSourceTables

List<PDataSourceTable> getDataSourceTables(PDataSource dataSource)

Gets the PDataSourceTables from the specified **dataSource** and returns them as a list.

getDataSources

List<PDataSource> getDataSources(PDataSourceType type)

Gets all the datasources of the specified **type** and returns them as a list. If type is **null** then all the datasources are returned.

getDescribedDrilldown

PDrilldown getDescribedDrilldown(PDrilldown drilldown)

Returns a new drilldown that is a description of the drilldown specified. **Note that a drilldown must have been run or executed before it can be described.**

getDocumentJobResult

PDocument getDocumentJobResult(int jobId)

Returns the result of the document job with id **jobId**.

getHostname

String getHostname()

Returns the hostname used for the initial connection.

getImportAreaByName

PImportArea getImportAreaByName(String name)

Returns the import area for the given **name**. Defaults to **PImportArea.PUBLIC**.

getInternalDataSource

PDataSource getInternalDataSource(PDataSourceType type, PImportArea area)

Returns the datasource for the specified datasource **type** and import **area**. Type can be **DELIMITED_FILES**, **JDBC** or **EXCEL_FILES**. Area can be **PUBLIC**, **GROUP** or **PRIVATE**.

getJob

PJob getJob(int jobId)

Returns a PJob object for a scheduled job with the given sequence id. Returns null if an error occurred.

getJobProgress

int getJobProgress(int jobId)

Returns the progress of the job with the specified **jobId**. Returns -1 if an error occurred.

getJobs

List<PJob> getJobs(PUser user)

Returns a list of jobs for the specified user, or **null** if an error occurred.

getLastErrorMessage

String getLastErrorMessage()

Returns the last error message (from the last operation). This is cleared after each new operation.

getLastResult

PSessionResult getLastResult()

Returns the last result (PSessionResult) for the last operation. This is cleared after each new operation.

getLastResultCode

int getLastResultCode()

Returns the last result code for the last operation. This is cleared after each new operation.

getLicenseDetails

PLicense getLicenseDetails()

Returns a PLicense object with details of the server license. Returns null if an error occurred.

getLogicalDirPath

String getLogicalDirPath(String name)

String getLogicalDirPath(DirName name)

Returns the path for a named logical directory on the server.

getMappingReport

PDocument getMappingReport(PObject object)

PDocument getMappingReport(String query)

PDocument getMappingReport(PDrilldown drilldown)

Returns the mapping document for the specified **object**, **query** or **drilldown**. The object must be a **Table** or a **View**. Returns **null** if the operation failed.

getObjects

Collection<PObject> getObjects(PObjectType type)

Collection<PObject> getObjects(PObjectType type, SortOrder order)

Collection<PObject> getObjects(PObjectType type, String pattern)

Collection<PObject> getObjects(PObjectType type, String pattern, SortOrder order)

Returns a list of objects of the specified **type** with an optional sort **order** and conforming to an optional **pattern**. Order can be **ASCENDING**, **DESCENDING** or **UNSORTED** and will default to **UNSORTED**. If a pattern is specified, then the list will only contain objects of the specified type whose name matches the pattern.

For example, if you wanted to get a list of all the tables in the database, you would call:

```
tables = getObjects (PObjectType.TABLE) ;
```

If you wanted a list of all the tables in alphabetical order, you would call:

```
alphaTables = getObjects (PObjectType.TABLE, SortOrder.ASCENDING) ;
```

Finally, if you wanted a list of all the tables whose names begin with an 'A', you would call:

```
aTables = getObjects (PObjectType.TABLE, "^A.*") ;
```

getObjectTypeForString

PObjectType getObjectTypeForString(String typeName)

Returns the PObjectType enumeration for a given object type string. This is more forgiving than doing PObjectType.valueOf(). Returns **null** if no match can be found.

getPort

int getPort()

Returns the port used to connect to the server.

getProductInformation

PProductInfo getProductInformation()

Returns product information for the application.

getQueryForObject

String getQueryForObject(PObject object, PQueryType type)

Returns the SQL query string for a given query type (PQueryType) for a given repository object (PObject) if the query is valid for that object. Returns null if an error occurred.

getRelationshipReport

PDocument getRelationshipReport(PObject relationship)

Returns the relationship report for the specified relationship, or null if the operation fails.

getResultForCode

static PSessionResult getResultForCode(int code)

Returns the session result for the specified code.

getReturnValue

Object getReturnValue()

Returns the return value

getServerUptime

long getServerUptime()

Returns the number of milliseconds that the server application has been running since last restart.

getSqlTableName

String getSqlTableName(PObject object)

Returns the SQL name for the given **object**. The object must be a table.

getSummaryDrilldown

PDrilldown getSummaryDrilldown(PDrilldown drilldown)

Returns a new drilldown that is a summarisation of the drilldown provided, or null if the operation failed.

getTableJoin

PObject getTableJoin(PObject lhColumn, PObject rhColumn)

PObject getTableJoin(PObject lhColumn, PObject rhColumn, int delay)

PObject getTableJoin(PObject lhTable, String lhColumnName, PObject rhTable, String rhColumnName)

PObject getTableJoin(PObject lhTable, String lhColumnName, PObject rhTable, String rhColumnName, int delay)

Returns the relationship object for the join between the **lhColumn** and the **rhColumn** with an optional **delay**. The delay is designed to be used directly after a load to allow the server some time to find the joins for the newly loaded table(s). The columns can optionally be specified using the **lhTable** or **rhTable** object and their respective names. Returns **null** if no relationship could be found or an error occurred.

getTableNameForQuery

String getTableNameForQuery(PObject object, PQueryType type)

Returns the name of the table for the specified query **type** on the specified **object**. Returns **null** if the query type is not supported by the object, or if an error occurred.

getTimeNow

long getTimeNow()

Returns the time now in milliseconds on the server.

getURLPrefix

String getURLPrefix()

Returns the URL prefix for the server's REST engine

getUser

PUser getUser()

Returns the User object for the user you logged in as.

getUsername

String getUsername()

Returns the username used to login with.

isCompressionEnabled

void abort()

Returns whether compression is being used in the communication stream. This is handled automatically and cannot be enabled or disabled manually.

isDebug

boolean isDebug()

Returns whether or not the session is in debug mode

isError

boolean isError()

Returns whether or not the last request resulted in an error.

isJobRunning

boolean isJobRunning(int jobId)

Returns whether or not the job with the specified **jobId** is running.

killSession

boolean killSession(long id)

Kills the session with the specified session **id**. Returns **true** if the operation was successful or **false** if it was not.

logoutUser

boolean logoutUser(PUser user, String reason, int timeout)

Logs out the specified **user** after the specified **timeout**, giving them the specified **reason**. Returns **true** if the operation was successful or **false** if it was not.

pingServer

boolean pingServer()

Pings the server to see if it is up and to keep it alive. Returns **true** if the server responds or **false** if it does not.

print

void print(String message)

Prints a message out to standard output. Shortcut for System.out.println();

printResult

void printResult()

static void printResult(PSessionResult result, String message, String extraMessage)

void print Result(String extraMessage)

Prints a formatted result message for a given result (PSessionResult), with an optional message and extra message. The alternate form uses the last result and message automatically. If not specified, the **extraMessage** defaults to an empty String.

removeTableFromContainer

boolean removeTableFromContainer(PObject container, PObject table)

Removes the given **table** from the given **container**. The table can be either a Table or a Table Version Group. The container can be a Table Version Group, a Project or a Schema. If the container is a Table Version Group then the table can only be a Table.

requeueJob

boolean requeueJob(int jobId)

Requeues the job with the specified **jobId**. Returns **true** if the operation is successful, or **false** if it is not.

saveAsQuery

PObject saveAsQuery(PDrilldown drilldown, String name, String description, boolean overwrite)

Creates a Saved Query from the **drilldown** specified, with the **name** and **description** specified. If overwrite is true, then the object will be created regardless of whether or not an object with the

same name exists already, otherwise the creation will only succeed if no Saved Queries with the same name already exist. Returns the newly-created object, or **null** if the operation fails.

saveAsView

PObject saveAsView(PDrilldown drilldown, String name, String description, boolean overwrite)

Creates a View from the **drilldown** specified, with the **name** and **description** specified. If **overwrite** is true, then the object will be created regardless of whether or not an object with the same name exists already, otherwise the creation will only succeed if no Views with the same name already exist. Returns the newly-created object, or **null** if the operation fails.

saveRelationship

boolean saveRelationship(PObject relationship)

Saves the given **relationship** if it is not already saved. Returns **true** if the operation succeeds or **false** if it fails.

saveTableConfiguration

boolean saveTableConfiguration(PDataSourceType type, PDataSourceTable table)

Saves the load configuration specified in the PDataSourceTable for later use. **Note: if you are going to load the table anyway, then you do not need to call this method.**

scheduleExport

int scheduleExport(String query, String jobName, PExportDefinition definition)

int scheduleExport(PDrilldown drilldown, String jobName, PExportDefinition definition)

Schedules an export job with the given **jobName** on the given **query** or **drilldown** as per the export **definition**. Returns the job id if the job has been successfully scheduled, otherwise returns 0.

scheduleLoad

int scheduleLoad(PLoadTableDefinition definition)

int scheduleLoad(PDataSource dataSource, PLoadTableDefinition definition)

int scheduleLoad(PDataSource dataSource, PLoadTableDefinition definition, String tableName)

Schedules a load job for the specified load table **definition** from the specified **dataSource** with an optional **tableName**. Returns the job id if the job has been successfully scheduled otherwise returns 0.

`scheduleNoteDocumentJob`

`int scheduleNoteDocumentJob(PObject object)`

`int scheduleNoteDocumentJob(PObject object, long time)`

Schedules a job to create the note report for the specified object. The optional **time** parameter allows you to choose when to start the job by giving the milliseconds of the required date and time. The default is the current time. Returns the job id if the job has been successfully scheduled otherwise returns 0.

`scheduleQualityReportJob`

`int scheduleQualityReportJob(PObject object)`

`int scheduleQualityReportJob(PObject object, long time)`

Schedules a job to create a quality report for the specified object. The object must be a **Table**. The optional **time** parameter allows you to choose when to start the job by giving the milliseconds of the required date and time. Default is current time. Returns the job id if the job has been successfully scheduled otherwise returns 0.

`scheduleReload`

`int scheduleReload(PDataSource dataSource, PObject table, PLoadTableDefinition definition)`

Schedules a load job to reload the specified **table** from the specified **dataSource** with the specified **definition**. Returns the job id if the job has been successfully scheduled otherwise returns 0.

`scheduleSaveAsTable`

`int scheduleSaveAsTable(PObject schema, PObject project, PDrilldown drilldown, String name, String description, long startTimestamp, boolean enableTokens, TableStorageType type)`

`int scheduleSaveAsTable(PObject schema, PObject project, PDrilldown drilldown, String name, String description)`

`int scheduleSaveAsTable(PObject projectOrSchema, PDrilldown drilldown, String name, String description)`

`int scheduleSaveAsTable(PDrilldown drilldown, String name, String description)`

Schedules a load job to save the specified **drilldown** as a table with the specified **name** and **description**. The table can optionally be loaded into a the specified **project** or **schema** and the values can optionally be tokened. It can also be given an optional table storage **type**. This can be one of

READ_ONLY, **APPENDABLE**, **TRANSACTIONAL**, **ARCHIVED_INDEXED** or **ARCHIVED**. Returns the job id if the job has been successfully scheduled otherwise returns 0.

scheduleTableDeletion

int scheduleTableDeletion(String name, String reason)

int scheduleTableDeletion(String name, int version, String reason)

int scheduleTableDeletion(PObject table, String reason)

Schedules a delete job for Table. Either the name of a Table (with optional version) can be used, or a repository object for the Table can be used. Returns 0 if the scheduling did not occur. A mandatory reason for deletion must be supplied.

scheduleTableValidation

List<Integer> scheduleTableValidation(List<PObject> tables)

Schedules a validation job for each table in the specified list of **tables**. Returns the job id if the job has been successfully scheduled otherwise returns 0.

sendDomainFile

boolean sendDomainFile(String localfile, String remotefile, boolean overwrite)

Sends the **localfile** to the content folder on the server with the name **remotefile**. If remotefile already exists and **overwrite** is true then it will be overwritten, otherwise the operation will fail. Returns **true** if the operation succeeds or false if the operation fails.

sendFileForLoad

boolean sendFileForLoad(String localfile, String remotefile, PImportArea area, boolean overwrite)

Sends the **localfile** to the import **area** folder on the server with the name **remotefile**. If remotefile already exists and **overwrite** is true then it will be overwritten, otherwise the operation will fail. Area can be **PImportArea.PUBLIC**, **PImportArea.GROUP** or **PImportArea.PRIVATE**. Returns **true** if the operation succeeds or false if the operation fails.

setDebug

void setDebug(boolean debug)

Allows the user to enable or disable debug mode on the session. In debug mode, exceptions will be printed out. This is false by default.

setReturnValue

void setReturnValue(Object value)

Sets the return value.

shutdownServer

boolean shutdownServer(int delay)

Shuts down the server after **delay** seconds.

startJob

boolean startJob(int jobId)

Starts a scheduled job which is waiting for a start time that has not been reached yet. Returns **false** if the operation failed and **true** if it worked.

stopJob

boolean stopJob(int jobId)

Stops a running scheduled job. Returns **false** if the operation failed and **true** if it worked.

unsaveRelationship

boolean unsaveRelationship(PObject relationship)

Unsave the specified **relationship**.

waitForJob

PJob waitForJob(int jobId)

Returns the job with the given **jobId**. Returns **null** if the job failed, was stopped or aborted, or if an error occurred.

PObject

This is a generic object representing a repository metadata object. PObjects implement the Comparable interface for easy comparison. As the information recorded for repository objects varies with the type of object, the different parameters are implemented as a map of properties (defined by the **PProperty** enumeration). A PProperty has a type defined by the **PPropertyType** enumeration. Property values themselves are plain objects, and it is up to the caller to cast them appropriately.

addPropertyToJSONObject

void addPropertyToJSONObject(JSONObject jsonObject, PProperty key, Collection<String> upropertyKeys)

void addPropertyToJSONObject(JSONObject jsonObject, PProperty key, Object value, Collection<String> upropertyKeys)

static void addPropertyToJSONObject(PProperty pobj, JSONObject jsonObject, PPropertyKey key, Object value, Collection<String> upropertyKeys)

throws JSONException

Adds the **key** to the **jsonObject** with an optional **value**. The list of **upropertyKeys** is used to filter user_property by the supplied keys, this can be **null**.

compareTo

int compareTo(PObject that)

Compares the object with another PObject return -1, 0 or 1 depending on whether this object is smaller, equal to or greater than the other one.

getAllowedQueryTypes

List<PQueryType> getAllowedQueryTypes()

Returns a list of PQueryType enumerations which are the types of query allowed by the object.

getAllowedReportTypes

List<PReportType> getAllowedReportTypes()

Returns a list of PReportTypes. These are the reports that it is possible to get for the object.

getById

static PObject getById(long id)

throws PException

Returns the repository object with the specified **id**.

getChildren

Collection<PObject> getChildren(PObjectType type, String pattern, SortOrder order)

throws PException

Returns a collection of child objects of a given type for an object, optionally sorted and optionally with their names filtered by a regular expression pattern. For example, this can be used to get the Columns for a Table. Valid values of SortOrder are:-

ASCENDING, DESCENDING, UNSORTED

getCreatedAction

PUserAction getCreatedAction()

Returns the PUserAction for the creation event for this object. A PUserAction provides details of who, when and why something was done.

getDefaultQuery()

String getDefaultQuery()

throws PException

Returns the default query action for the object. This is the same as the query provided in the user interface when the object icon is clicked on.

getId()

long getId()

throws PException

Returns the unique 64 bit identifier for the repository object.

getObjects

static List<PObject> getObjects(PObjectType type)

static List<PObject> getObjects(PObjectType type, SortOrder order)

static List<PObject> getObjects(PObjectType type, String pattern)

static List<PObject> getObjects(PObjectType type, String pattern, SortOrder order)

throws PException

Static methods to return all objects of a given type from the repository, with an optional regular expression filter pattern and an optional sort order. Valid values of SortOrder are:-

ASCENDING, DESCENDING, UNSORTED

getProperty

Object getProperty (PPROPERTY type)

Returns the value of a repository object property

getQuery

String getQuery(PUser user, PQueryType type)

throws PException

Returns the SQL query string for a given enumerated query type.

getType

PObjectType getType()

Returns the type of repository object as a PObjectType. A list of these can be found at the </types/object.html> REST url, or can be obtained by calling PObjectType.values().

iterator

Iterator<PPROPERTY> iterator()

Allows the user to iterate through the properties in a PObject. Returns the iterator for the properties.

toJSON

JSONObject toJSON(Collection<PPROPERTY> keys, Collection<String> upropertyKeys)

JSONObject toJSON()

throws JSONException

Converts the object to a JSON representation. The optional parameters, **keys** and **upropertyKeys**, are used to specify which properties and user properties the user wishes to include in the object.

toJSONArray

static JSONArray toJSONArray(Collection<PObject> objects)

Returns a JSONArray containing the JSON representation for each of the **objects** provided.

toJSONBrief

JSONObject toJSONBrief()

Converts the object to a JSON representation in brief form.

PObjectType

This object defines the types of object found in the server. It is used when calling other functions that require a PObjectType to be passed. For example, the findObject function requires the name and type of an object in order to find it, and the type needs to be defined in the PObjectType object:

```
scriptObj = psession.findObject(PObjectType.SCRIPT, scriptName);
```

A list of valid PObjectTypes can be found at the `/types/object.html` REST url, or can be obtained by calling `PObjectType.values()`.

PQueryType

This object defines the query types that can be found in the server. It is used when calling other functions that require a PQueryType to be passed. For example, when creating a drilldown, the type of drilldown needs to be defined along with the object that contains the drilldown:

```
var drilldown = psession.createDrilldown(pobject, PQueryType.ROW)
```

A list of all possible PQueryTypes can be found by either going to the `/queries/type.html` url on the REST server, or printing out the values returned by `PQueryType.values()`.

PDrilldown

This manages the execution of SQL queries to the database providing access to the results.

close

void close()

throws PException

Closes an active drilldown

execute

void execute()

throws PException

Executes the drilldown in a separate thread.

forwardIterator

Iterator<PDataRow> forwardIterator

Provides an iterator for stepping through the rows of a query in forward order

getAllColumns

List<ColumnDefinition> getAllColumns()

Returns a list of all columns in the database table being queried, whether visible or not.

getByName

ColumnDefinition getByName(String name)

Returns a column in the query by name

getCell

PDataCell getCell(long row, int column)

Returns a data cell at a given row/column position.

getColumnCount

int getColumnCount()

Returns the count of visible columns in the table being queried.

getColumns

List<ColumnDefinition> getColumns()

Returns the list of visible columns in the query

getDescription

String getDescription()

Returns the description for the query.

getFilter

ColumnFilter getFilter()

Returns the filter that the drilldown will use when deciding which columns to return.

getFilteredColumns()

List<ColumnDefinition> getFilteredColumns()

Returns a list of columns, filtered using the column's filter.

getProgress

int getProgress()

If the drilldown is being executed, then this method returns the progress of the execution, otherwise it returns 0.

getQuery

String getQuery()

Returns the query for this drilldown.

getRow

PDatRow getRow(long rowId)

Returns an entire row at a given row number.

getRowCount

long getRowCount()

Returns the count of rows for the query.

getTableName

String getTableName()

Returns the name of the table for the executed drilldown.

getTimestamp

long getTimestamp()

Returns the timestamp for the query which can be used to detect changes on reapplication.

getTitle

String getTitle()

Returns the descriptive title for the query.

hasExecuted

boolean hasExecuted()

Returns **true** if the drilldown has been executed, or **false** if it has not.

isBusy

Boolean isBusy()

Denotes if the query activity is currently busy, i.e. still executing

isReverseOrder

boolean isReverseOrder()

Denotes if default row iteration is in reverse order.

iterator

Iterator<PDataRow> iterator()

Returns the current iterator (which may or may not be reverse ordered).

reverseIterator

Iterator<PDataRow> reverseIterator

Provides an iterator to stepping through the the rows of a query in reverse order.

run

void run()

executes the drilldown in the same thread

setFilter

void setFilter(ColumnFilter filter)

Sets the filter that the drilldown will use when deciding which columns to return.

setReverseOrder

void setReverseOrder(boolean reverse)

Sets default iteration to be reverse or not reverse (forward).

toDocumentTable

PDocTable toDocumentTable(Color headingColour, DocumentTableStyle style, long startRow, int rowCount, boolean wordWrap, int columnCount, boolean addRows)

PDocTable toDocumentTable(Color headingColour, DocumentTableStyle style, long startRow, int rowCount, boolean wordWrap, int columnCount)

PDocTable toDocumentTable(long startRow, int rowCount, int columnCount, boolean addRows)

PDocTable toDocumentTable(long startRow, int rowCount)

PDocTable toDocumentTable(int rowCount)

Creates a table that can be used to export this drilldown as a part of a document.

PDataRow

This is the representation of a row of data in the results of a drilldown/query. The data row is mutable.

getDataCell

PDataCell getDataCell(int column)

Returns the cell value for a given column index.

iterator

Iterator<PDataCell> iterator

Returns an iterator to iterator over the cells in the row.

size

int size()

Returns the size of the row, in numbers of cells.

toCsv

String toCsv()

Converts the row to a line of comma-separated-values.

toJSON

JSONObject toJSON(List<ColumnDefinition> columns)

Converts the row to a JSON object.

PDataCell

This object represents a generic value at a cell location (row, column position) in the results of a query.

clear

void clear()

Resets the cell to its initial state.

getBackground

Color getBackground()

Return the background color for the cell

getDatatype

byte getDatatype()

Return the datatype for the cell. Valid values are:-

Type	Definition
Datatype.UNKNOWN	Unknown or not set, used for auto-typing
Datatype.INTEGER	Integer values
Datatype.DATE	Date/Time values
Datatype.DECIMAL	Decimal values
Datatype.MONEY	Monetary values
Datatype.ALPHANUMERIC	Any value that does not fit into INTEGER, DATE, DECIMAL or MONEY types
Datatype.NULL	Null

getForeground

Color getForeground()

Returns the foreground color for the cell.

getIcon

IconType getIcon()

Returns the icon enumeration for the cell.

getId

long getId()

Returns the id of a repository object that is associated with the cell value.

getMode

RenderMode getMode()

Returns the rendering mode for the cell.

getPresentation

String getPresentation()

Returns the display presentation for the value, i.e. what is seen by the user.

getType

CellType getType()

Returns the type of the cell. Valid values of the CellType enumeration are:-

VALUE, WARNING, ERROR

getValue

Object getValue()

Returns the value in the cell

isAutocasted

boolean isAutocasted

Returns whether or not the datacell has been autocasted.

isObfuscated

boolean isObfuscated()

Denotes if the cell is obfuscated

set

void set(byte datatype, Object object, String presentation)

void set(PDataCell cell)

void set(String value)

void set(String value, String presentation)

void set(BigInteger value)

void set(BigInteger value, String presentation)

void set(Long value)

void set(Long value, String presentation)

void set(Integer value)

void set(Integer value, String presentation)

void set(Short value)

void set(Short value, String presentation)

void set(Byte value)

void set(Byte value, String presentation)

void set(BigDecimal value)

void set(Double value)

void set(Double value, String presentation)

void set(Float value)

void set(Float value, String presentation)

void set(DateAndTime value)

void set(DateAndTime value, String presentation)

void set(Money value)

void set(Money value, String presentation)

Sets the value of the cell using a variety of signatures for different object types with an optional presentation that is different to that of the stored value.

setAutoCast

void setAutoCast(boolean setting)

Sets whether or not to autocast the cell, using **setting**.

setBackground

void setBackground(Color background)

Sets the background color for the cell.

setError

void setError(String message)

Sets the cell value to be an error message.

setForeground

void setForeground(Color foreground)

sets the foreground color for the cell

setIcon

void setIcon(IconType icon)

Sets the icon image using the IconType enumeration

setId

void setId(long id)

Sets the associated object id for the cell value.

setMode

void setMode(RenderMode mode)

Sets the rendering mode for the cell.

setObfuscated

void setObfuscated(boolean obfuscated)

Sets the obfuscated flag for the cell.

setPresentation

void setPresentation(String presentation)

Sets the presentation for the cell which can be different to the actual underlying value.

setWarning

void setWarning(String message)

Sets the cell value to be a warning message.

toCsvString

String toCsvString()

Converts the cell to a comma-separated value.

toJSON

JSONObject toJSON()

Converts the cell to a JSON representation.

ColumnDefinition

This is the definition of a column in a drilldown.

getAlias

String getAlias()

Returns the alias of the column.

getBackground

Color getBackground()

Returns the background color of the column.

getDatatype

byte getDatatype()

Returns the datatype of the column.

getDisplayFormat

String getDisplayFormat()

Returns the display format of the column. This should be used to format the values of the cells for that column.

getDisplayName

String getDisplayName()

Returns the display name for the column.

getExpectedFormat

String getExpectedFormat()

Returns the expected format of the column.

getFalseText

String getFalseText()

Returns the text to display if the value is a boolean **false** value.

getFontDefinition

String getFontDefinition()

Returns the definition of the font used in the column.

getForeground

Color getForeground()

Returns the foreground color for the column.

getHeadingBackground

Color getHeadingBackground()

Returns the background color for the column's heading.

getHeadingForeground

Color getHeadingForeground()

Returns the foreground color for the column's heading.

getIndex

int getIndex()

Returns the column's index.

getLineWeight

LineWeight getLineWeight(LinePosition side)

Returns the line weight of the line on the specified **side**. Possible values for LinePosition are **TOP**, **LEFT**, **BOTTOM** or **RIGHT**. Possible values for LineWeight are **NONE**, **NORMAL**, **MEDIUM**, **HEAVY**. Each line weight has a color associated with it that can be obtained by calling `getColor()`.

`getLocale`

Locale `getLocale()`

Returns the Locale of the column.

`getMaxLength`

int `getMaxLength()`

Returns the length of the longest value in the column.

`getMaxValue`

Object `getMaxValue()`

Returns the maximum value in the column.

`getMinValue`

Object `getMinValue()`

Returns the minimum value in the column.

`getMode`

RenderMode `getMode()`

Returns the render mode for the column.

`getName`

String `getName()`

Returns the name of the column

`getPrecision`

int `getPrecision()`

Returns the precision of the column.

getScale

int getScale()

Returns the scale of the column.

getTrueText

String getTrueText()

Return the text to display if the value is a boolean **true** value.

getUniqueThreshold

int getUniqueThreshold()

Returns the unique threshold of the column.

getWidth

int getWidth()

Returns the width of the column.

isNullable

boolean isNullable()

Returns whether or not the column is allowed to contain null values.

isShowHints

boolean isShowHints()

Returns whether or not hints should be shown

isShowingHints

boolean isShowingHints()

Returns whether or not hints are being shown

isVisible

boolean isVisible()

Returns whether or not the column is visible.

setAlias

void setAlias(String alias)

Sets the alias of the column.

setBackground

void setBackground(Color background)

Sets the background color for the column.

setDatatype

void setDatatype(byte datatype)

Sets the datatype of the column.

setDisplayFormat

void setDisplayFormat(String displayFormat)

Sets the format to use when displaying the column's values.

setExpectedFormat

void setExpectedFormat(String expectedFormat)

Sets the expected format.

setFalseText

void setFalseText(String falseText)

Sets the text to display if the value is a boolean value **false**.

setFontDefinition

void setFontDefinition(String fontDefinition)

Sets the font definition.

setForeground

void setForeground(Color foreground)

Sets the foreground color of the column.

setHeadingBackground

void setHeadingBackground(Color headingBackground)

Sets the background color for the column's heading.

setHeadingForeground

void setHeadingForeground(Color headingForeground)

Sets the foreground color for the column's heading.

setIndex

void setIndex(int index)

Sets the index of the column.

setLineWeight

void setLineWeight(LineWeight left, LineWeight top, LineWeight right, LineWeight bottom)

Sets the line weight that should be used when rendering the border on each side.

setLocale

void setLocale(Locale locale)

Sets the locale of the column.

setMaxLength

void setMaxLength(int maxLength)

Sets the maximum value length of the column.

setMaxValue

void setMaxValue(Object maxValue)

Sets the maximum value of the column.

setMinValue

void setMinValue(Object minValue)

Sets the minimum value of the column.

setMode

void setMode(RenderMode mode)

Sets the render mode of the column.

setName

void setName(String name)

Sets the name of the column.

setNullable

void setNullable(boolean nullable)

Sets whether or not the column can contain null values.

setPrecision

void setPrecision(int precision)

Sets the precision of the column.

setScale

void setScale(int scale)

Sets the scale of the column.

setShowHints

void setShowHints(boolean showHints)

Sets whether or not hints should be shown.

setTrueText

void setTrueText(String trueText)

Sets the text to display if the value is a boolean value **true**.

setUniqueThreshold

void setUniqueThreshold(int uniqueThreshold)

Sets the unique threshold on the column.

setVisible

void setVisible(boolean visible)

Sets whether or not the column is visible.

setWidth

void setWidth(int width)

Sets the width of the column.

toJSON

JSONObject toJSON()

Returns a JSON representation of the column definition.

PUser

This defines a user of the product

Constructor

PUser(String username)

PUser(long id)

These constructors find the user using either their name or id.

getDisplayName

String getDisplayName()

Returns the display name of the user.

getGroupId

long getGroupId()

Returns the id of the user's group.

getId

long getId()

Returns the id of the user.

getName

String getName()

Returns the name of the user.

getProfileLimit

int getProfileLimit

Returns the profile limit for the user.

toJSON

JSONObject toJSON()

Returns a JSON representation of the user.

PUserAction

This provides all information necessary for an action made by a user.

getReason

String getReason()

Returns the reason for the action

getTime

PTimestamp getTime()

Returns the time the action took place.

getTimestamp

long getTimestamp()

Returns the milliseconds of the time the action took place.

getUser

PUser getUser()

Gets the user who made the action.

toJSON

JSONObject to JSON()

Returns a JSON representation of the user action.

ProcessState

This enumeration tells you what state a job is in. Possible values are:

CREATING, INITIALISING, RUNNING, WAITING, FINISHING, TIDYING, ABORTED, FAILED, COMPLETED, STOPPED, QUEUED, LOGGING or QUEUE_FULL.

getDescription

String getDescription()

Returns a human-readable description of the state.

sendCompletionMessage

boolean sendCompletionMessage()

Returns whether or not the process state will send a message on completion.

toJSON

JSONObject to JSON()

Returns a JSON representation of the process state.

PJob

This object is a representation of a job in the scheduler. **Note that you will need to keep fetching it in order to get updated progress and process state.**

getEndTime

long getEndTime()

Returns the time the job ended.

getId

int getId()

Returns the sequence id of the job.

getName

String getName()

Returns the name of the job.

getOwner

PUser getOwner()

Returns the user that kicked off the job.

getProgress

int getProgress()

Returns the progress of the job.

getScheduledTime

long getScheduledTime()

Returns the milliseconds for the time the job was scheduled to start.

getStartTime

long getStartTime()

Returns the milliseconds for the time the job actually started.

getState

ProcessState getState()

Returns the state of the job.

getTableId

long getTableId()

Returns the table id associated with the Job, if any

getType

JobType getType()

Returns the type of the job.

setEndTime

void setEndTime(long endTime)

Sets the time the job finished.

setId

void setId(int id)

Sets the id of the job.

setName

void setName(String name)

Sets the name of the job.

setOwner

void setOwner(PUser owner)

Sets the owner of the job.

setProgress

void setProgress(int progress)

Sets the progress of the job.

setScheduledTime

void setScheduledTime(long scheduledTime)

Sets the time the job was scheduled to start, in milliseconds.

setStartTime

void setStartTime(long startTime)

Sets the time the job actually started, in milliseconds.

setState

void setState(ProcessState state)

Sets the state of the job.

setTableId

void setTableId()

Sets the table id of the job.

setType

void setType(JobType type)

Sets the type of the job.

toJSON

JSONObject toJSON()

Returns a JSON representation of the job.

PLicense

This object contains information about the product license.

getCapabilities

String getCapabilities()

Returns a String representation of the capabilities of the license.

getCpus

int getCpus()

Returns the maximum number of cpus that the server is allowed to use.

getExpiryDate

String getExpiryDate()

Returns the expiry date of the license.

getLicenseKey

String getLicenseKey()

Returns the license's key.

getLicensedEthernetAddress

String getLicensedEthernetAddress

Returns the Ethernet address that this license is for.

getLicensedSessions

int getLicensedSessions()

Returns the number of concurrent sessions that the license will allow.

getMaximumRows

int getMaximumRows()

Returns the maximum number of rows of a table that can be loaded into the database.

getMaximumTables

int getMaximumTables()

Returns the maximum number of tables that can be loaded into the database.

getPort

int getPort()

Returns the port that this license is for.

toJSON

JSONObject toJSON()

Returns a JSON representation of the license.

PPermissions

This object contains information about the various permissions on an object.

getGroup

List<AccessRight> getGroup()

Returns the access rights on the object for users in the group which contains the user who created it.

getUser

List<AccessRight> getUser()

Returns the access rights on the object for the user who created it.

getWorld

List<AccessRight> getWorld()

Returns the access rights on the object for any user who is not the user that created the object, or in that user's group.

toJSON

JSONObject toJSON()

Returns a JSON representation of the permissions.

PScriptExecutor

This object is used in the execution of a script.

getException

Exception getException()

Returns the last exception that was caused by the script.

getInfo

String getInfo()

Returns information about the executor.

getLanguageFromFile

static ScriptLanguage getLanguageFromFile(String scriptFile)

Uses the file extension of the given **scriptFile** to calculate the language used. Returns the calculated language.

getMessage

String getMessage()

Returns the message.

getResult

PSessionResult getResult()

Returns the result of the script.

getScriptResult

Object getScriptResult()

Returns the object result of the script.

PAccessControl

This object describes the access control rights that a user has on an object.

getRights

List<AccessRight> getRights()

Returns the access rights.

getUser

PUser getUser()

Returns the user.

setRights

void setRights(List<AccessRight> rights)

Sets the rights. An access right can be one of **READ**, **READ_OBFUSCATED**, **MODIFY**, **DELETE**, **EXPORT**, **NONE**, **ALL** or **ANNOTATE**.

setUser

void setUser(PUser user)

Sets the user.

toJSON

JSONObject to JSON()

Returns a JSON representation of the access control.

PDataSource

This object describes a datasource. These are used in the loading of data into Experian Pandora.

getDataSourceType

PDataSourceType getDataSourceType()

Returns the type of the datasource.

getDisplayName

String getDisplayName()

Returns the display name of the datasource.

getId

long getId()

Returns the id of the datasource.

getName

String getName()

Returns the name of the datasource.

toJSON

JSONObject toJSON()

Returns a JSON representation of the datasource.

PDataSourceColumn

This object describes a column on a data source table. It can be changed to change how that column is loaded.

getColumnMetadata

ColumnMetadata getColumnMetadata()

Returns the metadata of the column.

getName

String getName()

Returns the name of the column.

setName

void setName(String name)

Sets the name of the column.

toJSON

JSONObject toJSON()

Returns a JSON representation of the column

ColumnMetadata

This object describes a column's metadata.

canHaveNulls

boolean canHaveNulls()

Returns whether or not the column can contain null values.

getBackground

Color getBackground()

Returns the background color for the column.

getDatatype

byte getDatatype()

Returns the datatype of the column.

getDefaultMetadata

ColumnMetadata getDefaultMetadata()

Returns the default metadata of the column.

getDefaultValueWhenNull

Object getDefaultValueWhenNull()

Returns the value to display if the value is **null**.

getDisplayFormat

String getDisplayFormat()

Returns the display format for the column.

getFalseText

String getFalseText()

Returns the text to display if a value is a boolean **false** value.

getFontDefinition

String getFontDefinition()

Returns the definition of the font for that column.

getForeground

Color getForeground()

Returns the foreground color of the column.

getFormat

byte[] getFormat()

Returns the format of the column.

getHeadingBackground

Color getHeadingBackground()

Returns the background color for the column's heading.

getHeadingForeground

Color getHeadingForeground()

Returns the foreground color for the column's heading.

getKeyUniqueLevel

int getKeyUniqueLevel()

Returns the key unique level for the column

getLength

int getLength()

Returns the length of the column.

getLocale

Locale getLocale()

Returns the locale of the column.

getMaxValue

Object getMaxValue()

Returns the highest value in the column.

getMinValue

Object getMinValue()

Returns the lowest value in the column.

getNameForMetadataSql

String getNameForMetadataSql(boolean beautify)

Returns the name for that column that should be used in metadata sql.

getNumberPrecision

int getNumberPrecision()

Returns the precision of numbers in the column.

getNumberScale

int getNumberScale()

Returns the scale of numbers in the column.

getObjectId

long getObjectId()

Returns the object id for the column.

getRawLineWeight

int getRawLineWeight()

Returns the raw line weight of the column.

getRenderMode

RenderMode getRenderMode()

Returns the render mode of the column.

getSortBehaviour

SortBehaviour getSortBehaviour()

Returns the sort behaviour of the column. This defines whether or not a sort of the column should be case sensitive (found out by calling **isCaseSensitive()** on the SortBehaviour) and also whether or not the sort should use collation (found out by calling **isUsingCollation()** on the SortBehaviour).

getStandardisationRule

int getStandardisationRule()

Returns the standardisation rule for the column.

getTrueText

String getTrueText()

Returns the text to display if a value is a boolean **true** value.

getWidth

int getWidth()

Returns the width of the column.

isDefaultAppliedToLeft

boolean isDefaultAppliedToLeft()

Returns whether or not the default null value is applied if the column is on the left-hand side of a join.

isDefaultAppliedToRight

boolean isDefaultAppliedToRight()

Returns whether or not the default null value is applied if the column is on the right-hand side of a join.

isWidthInPixels

boolean isWidthInPixels()

Returns whether or not the result of **getWidth()** is in pixels.

setBackground

void setBackground(Color background)

Sets the background color of the column.

setCanHaveNulls

void setCanHaveNulls(boolean nullable)

Sets whether or not the column can contain null values.

setDatatype

void setDatatype(byte datatype)

Sets the datatype of the column.

setDefaultAppliedToLeft

void setDefaultAppliedToLeft(boolean setting)

Sets whether or not the default null value is applied if the column is on the left-hand side of a join.

setDefaultAppliedToRight

void setDefaultAppliedToRight(boolean setting)

Sets whether or not the default null value is applied if the column is on the right-hand side of a join.

setDefaultValueWhenNull

void setDefaultValueWhenNull(Object value)

Sets the default value to use when the value is null.

setDisplayFormat

void setDisplayFormat(String displayFormat) Sets the display format for the column

setFalseText

void setFalseText(String text) Sets the text to display if a value is a boolean **false** value.

setFontDefinition

void setFontDefinition(String definition) Sets the font definition for the column.

setForeground

void setForeground(Color foreground) Sets the foreground color for the column.

setHeadingBackground

void setHeadingBackground(Color c) Sets the background color to use in the heading of this column.

setHeadingForeground

void setHeadingForeground(Color c) Sets the foreground color to use in the heading of this column.

setKeyUniqueLevel

void setKeyUniqueLevel(byte keythreshold) Sets the key unique level.

getLength

void setLength(int length) Sets the length of the column.

setLineWeight

void setLineWeight(LineWeight left, LineWeight top, LineWeight right, LineWeight bottom)

void setLineWeight(int lineWeight)

Sets the line weight of the column.

setLocale

void setLocale(Locale locale) Sets the locale of the column

setRenderMode

void setRenderMode(RenderMode mode) Sets the render mode of the column.

setTrueText

void setTrueText(String text) Sets the text to display if a value is a boolean **true** value.

setWidth

void setWidth(int width) Sets the width of the column.

setWidthInPixels

void setWidthInPixels(boolean b) Sets whether or not the column's width is in pixels.

PDataSourceTable

This object describes a data source table. It can be changed to change how the table is loaded.

addNewColumn

void addNewColumn() Adds a new column to the table.

deleteLastColumn

void deleteLastColumn() Deletes the last column.

getColumns

List<PDataSourceColumn> getColumns() Returns the columns.

getExternalName

String getExternalName() Returns the table's external name.

getLocale

Locale getLocale() Returns the locale of the table.

getName

String getName() Returns the name of the table.

setLocale

void setLocale(Locale locale) Sets the locale of the table.

setName

void setName(String name) Sets the name of the table.

toJSON

JSONObject toJSON() Returns a JSON representation of the data source table.

PLoadPermissions

This object describes the load permissions on a table.

Constructor

PLoadPermissions()

getAccessControlList

List<PAccessControl> getAccessControlList()

Returns the list of access controls that the table should have once it is loaded.

hasAccessRight

boolean hasAccessRight(PUser user, AccessRight requestedRight)

Returns whether or not the specified **user** will have the **requestedRight** on the table.

setAccessControlList

void setAccessControlList(List<PAccessControl> acl) Sets the access control list to **acl**.

setAccessRights

void setAccessRights(AccessLevel level, List<AccessRight> rights)

Sets the access rights for the specified access level. Access level can be one of **USER**, **GROUP**, or **WORLD**. An access right can be one of **READ**, **READ_OBFUSCATED**, **MODIFY**, **DELETE**, **EXPORT**, **NONE**, **ALL** or **ANNOTATE**.

PLoadTableDefinition

This object holds all the information necessary to load a table into Experian Pandora.

Constructor

PLoadTableDefinition()

PLoadTableDefinition(PObject schema, PObject project, Locale locale, long scheduledTime, PLoadPermissions permissions, TableStorageType storageType, boolean ignoreEmptyTables)

schema – The schema the table should go into, can be null.

project – The project the table should go into, can be null.

locale – The locale of the table, can be null.

scheduledTime – The time the load is scheduled to start.

permissions – Permissions the table will have.

storageType – The type of storage to use for the table.

ignoreEmptyTables – Whether or not to ignore empty tables in the load.

getLocale

Locale getLocale() Returns the locale that the table will be given on load.

getPermissions

PLoadPermissions getPermissions() Returns the permissions that the table will be given on load.

getProject

PObject getProject() Returns the project that the table will be put into on load, can be **null**.

getScheduledTime

long getScheduledTime() Returns the milliseconds of the time that the load was scheduled.

getSchema

PObject getSchema() Returns the schema that the table will be put into on load, can be **null**.

getStorageType

TableStorageType getStorageType()

Returns the storage type of the table. This can be one of: **READ_ONLY**, **APPENDABLE**, **TRANSACTIONAL**, **ARCHIVED_INDEXED**, or **ARCHIVED**.

getTable

PDataSourceTable getTable() Returns the table to be loaded.

ignoreEmptyTables

boolean ignoreEmptyTables() Returns whether or not the loader should ignore empty tables.

setIgnoreEmptyTables

void setIgnoreEmptyTables(boolean ignoreEmptyTables)

Sets whether or not the loader should ignore empty tables.

setLocale

void setLocale(Locale locale) Sets the locale of the table to be loaded.

setPermissions

void setPermissions(PLoadPermissions permissions)

Sets the permissions on the table to be loaded.

setProject

void setProject(PObject project) Sets the project that the loaded table will go into.

setScheduledTime

void setScheduledTime(long scheduledTime) Sets the time the load is scheduled to start.

setSchema

void setSchema(PObject schema) Set the schema that the loaded table will go into.

setStorageType

void setStorageType(TableStorageType storageType) Sets the storage type of the table.

setTable

void setTable(PDataSourceTable table) Sets the table to be loaded.

toJSON

JSONObject toJSON() Returns a JSON representation of the load definition.

PDocument

This object describes a document to be exported. These are used to create reports.

Constructor

PDocument()

addDocument

void addDocument(PDocument doc) Adds a document to the PDocument.

addPageBreak

void addPageBreak() Adds a page break to the document.

addSection

void addSection(PDocSection section) Adds a section to the document.

addSpacing

void addSpacing(int rows) Adds spacing to the document.

addTable

void addTable(PDocTable table) Adds a table to the document.

addText

void addText(PDocText text) Adds text to a document.

isBook

boolean isBook() Returns whether or not the document contains a series of documents and, therefore, is a document book.

setDescription

void setDescription(String description) Sets the description for the document.

setTitle

void setTitle(String title) Sets the title of the document.

PDocSection

This object describes a section of a document to be exported.

Constructor

PDocSection(String title, String description)

addTable

void addTable(PDocTable table) Adds a table to the section.

addText

void addText(PDocText text) Adds text to the section.

PDocTable

This object describes a table in a document to be exported.

setObjectId

void setObjectId(long id) Sets the object id on the table.

PDocText

This object describes a block of text in a document to be exported.

Constructor

PDocText(String title, Collection<String> text)

PDocText(String title, File file)

PDocText(String title, String text)

title – The title of the area of text.

text – The text to go into the area of text.

file – The file to read the text from.

PExportDefinition

This object holds the information required for an export.

Constructor

PExportDefinition()

PExportDefinition(String serverFileName)

PExportDefinition(ExportType exportType, String serverFilename, String charsetName, boolean forceQuotes, char quoteCharacter, ExportColumnNameStyle columnNameStyle, PivotOption pivotOption)

exportType – The export type to use. This can be one of the following: **CSV, TAB, PIPE, HTML, XLS, XLSX, PDF** or **DDL**.

serverFilename – The name to give the export file on the server.

charsetName – the name of the character set to use in the export.

forceQuotes – whether or not to force quotes in the export.

quoteCharacter – the quote character to use.

columnNameStyle – the style to use for column names.

This can be one of:

- **DISPLAY_NAME** – which uses the display name of the column
- **EXTERNAL_NAME** – which uses the external name of the column
- **ALIAS** – which uses the alias of the column
- **CAMEL_CASE** – which prints out the column's display name in camel case
- **TITLE_CASE** – which prints out the column's display name in title case
- **HUMANIZED** – which humanises the column's display name.
- **DATABASE_STYLE** – which prints out the column's display name in database style
- **LOWER_CASE** – which prints out the column's display name in lower case
- **UPPER_CASE** – which prints out the column's display name in upper case
- **NONE** – no headings are printed out

pivotOption – the option to use when deciding whether or not to pivot. This can be one of: **AUTOMATIC, YES** or **NO**.

Note: Only HTML and PDF exports support pivoting.

getCharsetName

String getCharsetName() Returns the name of the character set to be used in the export.

getColumnNameStyle

ExportColumnNameStyle getColumnNameStyle()

Returns the name style to be used for the columns. This can be one of: **DISPLAY_NAME**, **EXTERNAL_NAME**, **ALIAS**, **NAME**, **CAMEL_CASE**, **TITLE_CASE**, **HUMANIZED**, **DATABASE_STYLE**, **LOWER_CASE**, **UPPER_CASE**, **NONE**.

getPivotOption

PivotOption getPivotOption()

Returns the pivot option for the export. This can be one of **AUTOMATIC**, **YES** or **NO**.

getQuoteCharacter

char getQuoteCharacter() Returns the quote character for the export.

getServerFilename

String getServerFilename() Returns the name of the file that will be created on the server.

getType

ExportType getType()

Returns the type of the export. This can be either **PDF** or **HTML** for a document or **PDF**, **HTML**, **CSV**, **TAB**, **PIPE**, **XLS**, **XLSX**, **JDBC** or **DDL** for a table.

isForceQuotes

boolean isForceQuotes() Returns whether or not quotes will be forced in the export.

setCharsetName

void setCharsetName(String charsetName)

Sets the name of the character set to be used in the export.

setColumnNameStyle

void setColumnNameStyle(ExportColumnNameStyle columnNameStyle)

Sets the name style to be used for the columns. This can be one of: **DISPLAY_NAME**, **EXTERNAL_NAME**, **ALIAS**, **NAME**, **CAMEL_CASE**, **TITLE_CASE**, **HUMANIZED**, **DATABASE_STYLE**, **LOWER_CASE**, **UPPER_CASE**, **NONE**.

setForceQuotes

void setForceQuotes(boolean forceQuotes) Sets whether or not quotes will be forced in the export.

setPivotOption

void setPivotOption(PivotOption pivotOption)

Sets the pivot option for the export. This can be one of **AUTOMATIC**, **YES** or **NO**.

setQuoteCharacter

void setQuoteCharacter(char quoteCharacter) Sets the quote character for the export.

setServerFilename

void setServerFilename(String serverFilename)

Sets the name of the file that will be created on the server.

setType

void setType(ExportType type) Sets the type of the export. This can be either **PDF** or **HTML**.

PFunctionVersionInfo

This object contains information about a specific version of a function.

getAction

PUserAction getAction() Returns the user action for the version's creation.

getExpression

String getExpression() Returns the function expression for the version.

setAction

void setAction(PUserAction action) Sets the user action for the version's creation.

setExpression

void setExpression(String expression) Sets the expression for the version

toJSON

JSONObject toJSON() Returns a JSON representation of the version.

PGroup

This object defines a user group

Constructor

PGroup(String groupName)

PGroup(long id)

These constructors find the group using either its name or id.

getDisplayName

String getDisplayName() Returns the display name of the group

getId

long getId() Returns the id of the group.

getName

String getName() Returns the name of the group.

getUsers

Collection<PUser> getUsers() Returns a collection of the users that are part of the group.

toJSON

JSONObject toJSON() Returns a JSON representation of the group.

PProperties

This object defines a list of properties

Constructor

PProperties(String name)

add

void add(PProperty prop, Object value) Adds a property to the properties list.

containsKey

boolean containsKey(PProperty key)

Returns whether or not the list of properties contains the given **key**.

containsValue

boolean containsValue(Object value)

Returns whether or not the list of properties contains the given **value**.

getName

String getName() Returns the name of the property list.

getValue

Object getValue(PProperty property) Returns the value associated with the given **property**.

isEmpty

boolean isEmpty() Returns whether or not the property list is empty.

iterator

Iterator<PProperty> iterator() Returns an iterator for the properties.

keySet

Set<PProperty> keySet() Returns a set of the properties in this property list.

setName

void setName(String name) Sets the **name** of the group of properties.

size

int size() Returns the size of the list of properties.

toJSON

JSONObject toJSON() Returns a JSON representation of the list of properties.

values

Collection<Object> values() Returns a collection of the values of all the properties in the property list.

PPropertiesGroup

This object defines a group of properties.

Constructor

PPropertiesGroup(String name)

add

boolean add(PProperties) Adds a list of properties to the property group.

clear

void clear() clears the properties group

get

PProperties get(int index) Returns the list of properties at the specified **index**.

getName

String getName() Returns the name of the property group.

isEmpty

boolean isEmpty() Returns whether or not the list of properties is empty.

iterator

Iterator<PProperties> iterator() Returns the iterator for the group.

listIterator

ListIterator<PProperties> listIterator() Returns the list iterator for the group.

setName

void setName(String name) Sets the name of the property group.

size

int size() Returns the size of the property group

subList

List<PProperties> subList(int fromIndex, int toIndex)

Returns a sublist of the properties in the group that are between **fromIndex** and **toIndex**.

toJSON

JSONObject toJSON() Returns a JSON representation of the property group.

PProperty

This enumeration defines the properties that are used in PObjects.

Possible Properties are:

Property	Type	Description	Used In
AUTHOR	User	The author of the object	Note_Detail
TEXT	String	The text in the object	Note_Detail
HTML_TEXT	String	An HTML representation of the text in the object	Note_Detail
NAME	String	The name of the object	All Objects
DISPLAY_NAME	String	The display name of the object	All Objects
DESCRIPTION	String	The description of the object	All Objects
STEWARD	User	The user who stewards the object	All Objects
STEWARD_GROUP	Object	The group of the user who stewards the object.	All Objects
LAST_MODIFIED	User Action	Information about when the object was last modified	All Objects
CREATED	User Action	Information about when the object was created	All Objects
OBJECT_TYPE	Object Type	The type of the object	All Objects

ID	String	The id of the object	All Objects
PARENT	Object	The parent of the object	All Objects
NOTE_COUNT	Integer	The number of notes on the object	All Objects
PERMISSIONS	Permissions	The permissions on the object	All Objects
COLUMN_COUNT	Integer	The number of columns in the object	Table, View_Table
VALIDATION_STATUS	String	The validation status of the object	Table
DATASOURCE	String	The datasource the table came from	Table
DEPENDENCY_COUNT	Integer	The number of dependencies on the table	Table
KEY_COUNT	Integer	The number of keys on the table	Table
ROW_COUNT	Integer	The number of rows in the object	Table, View_Table
KEY_THRESHOLD	Integer	Limit the maximum number of keys that the process will find	Table
KEY_LEVEL	Integer	The Maximum key level	Table
DEPENDENCY_THRESHOLD	Integer	Limit the maximum number of dependencies that the process will discover.	Table
DEPENDENCY_LEVEL	Integer	The maximum dependency level	Table
LAST_KEY_TIME	Timestamp	A timestamp for the last time a key was found on the table	Table
LAST_DEPENDENCY_TIME	Timestamp	A timestamp for the last time a dependency was found on the table	Table
LAST_KEY_USER	User	The last user to find a key on the table	Table
LAST_DEPENDENCY_USER	User	The last user to find a dependency on the table	Table
LAST_VALIDATION_TIME	Timestamp	A timestamp for the last validation time.	Table, Column
LAST_VALIDATION_USER	User	The user who last started a validation on the object.	Table, Column
LOAD_DURATION	Integer	The time it took a table to load	Table
LOW_VALIDATION_THRESHOLD	Integer	The threshold between an ok result and a failed result in validation	Table
HIGH_VALIDATION_THRESHOLD	Integer	The threshold between a pass result and an ok result in a validation	Table
SCORE	Decimal	The value that the object scored in validation	Table, Column
VOLUME_SCORE	Decimal	The volume score on a column	Column
SUBJECT_AREA	Object	The Schema for the table	Table
VERSION	Integer	The version of the object	Note_Detail, Table
CHILD_COUNT	Integer	The number of children the object has	All Objects
ABBREVIATIONS	String	A comma-separated list of abbreviations for the business term	Business_Term
DEFINITION	String	The definition of the business term	Business_Term
EXAMPLE	String	An example of the business term	Business_Term
SENTENCE	String	An example of the business term being used in a sentence	Business_Term
BROADER_TERM	String	A broader term for the business term	Business_Term
USAGE	String	The usage of the business term	Business_Term
SOURCE	String	The source of the object	Business_Term, Table_Version_Group

STATUS	String	The status of the object	Business_Term, Relationship, User
LAST_STATUS_CHANGE	User Action	Information about the last status change on the business term.	Business_Term
SYNONYMS	Object List	Returns a list of business terms that are synonymous with the business term	Business_Term
RELATED_TERMS	Object List	Returns a list of business terms that are related to the business term	Business_Term
RELATED_COLUMNS	Object List	Returns a list of columns that are related to the business term	Business_Term
TYPE	String	The type of the object	Domain, Saved_Query, Function
SKIP_FIRST_ROW	Boolean	Whether or not to skip the first row when loading a domain	Domain
IS_SENSITIVE	Boolean	Whether or not the domain contains sensitive information	Domain
FILENAME	String	The name of the file that contains the domain data	Domain
VALUE	String	The value of a user property or named constant	All Objects
ROLE	Object	The User's Role	User
EMAIL	String	The email address of the user	User
PHONE1	String	The primary phone number of the user	User
PHONE2	String	The secondary phone number of the user	User
LAST_LOGGED_IN_TIME	Timestamp	A timestamp for when the user last logged in	User
LAST_LOGGED_IN	String	A textual representation of the timestamp for when the user last logged in	User
CURRENT_PROJECT	Object	The User's current project	User
IS_DISABLED	Boolean	Whether or not the user is disabled in Experian Pandora	User
IS_ADMIN	Boolean	Whether or not the user is the system administrator	User
BUSINESS_IMPACT	Object	The business impact on a note	Note
TECHNICAL_IMPACT	Object	The technical impact on a note	Note
NOTED_OBJECT	Object	The object the note is for	Note
ASSIGNED_TO	User	The user the note is assigned to	Note
ASSIGNED_BY	User	The user who assigned the note	Note
ASSIGNED_WHEN	Timestamp	A timestamp for when the note was assigned	Note
NOTE_ID	Integer	The sequence id of the note	Note
INDEX	Integer	The index of the note detail	Note_Detail
QUERY	String	The query on the object	Note_Detail, Saved_Query, View_Table
QUERY_DESCRIPTION	String	The description of the query on a note detail	Note_Detail
LHS_COLUMNS	Object List	The left-hand columns in a key	Key

RHS_COLUMN	Object	The right-hand column in a key or dependency	Key, Dependency
COVERAGE	Integer	The amount of a column that follows the dependency	Dependency
BOTH_NULL_COUNT	Integer	The number of rows where both the left-hand column and right-hand column are null	Dependency
SINGLETON_NULL_COUNT	Integer	The number of rows where either the left-hand column or the right-hand column, but not both, contain a null value	Dependency
NULL_COUNT	Integer	The number of null values in the key	Key
QUALITY	Integer	The quality of the key	Key
ERROR_COUNT	Integer	The number of errors in the key	Key
IS_APPROXIMATE	Boolean	Whether or not the key is approximate	Key
IS_SAMPLED	Boolean	Whether or not the key is sampled	Key
TESTED_QUALITY	Integer	The quality of the key when tested	Key
VERSIONS	Object List	The versions of the object	Function, Table_Version_Group
RELOAD_METHOD	String	The reload method to use for the table version group	Table_Version_Group
RELOAD_PERIOD	String	The period after which to check whether the table needs reloading	Table_Version_Group
VERSIONS_TO_KEEP	Integer	The number of versions to keep before starting to archive them	Table_Version_Group
VERSIONS_TO_ARCHIVE	Integer	The number of versions to archive before starting to delete them	Table_Version_Group
WORKING_DAYS_ONLY	Boolean	Whether or not to only reload in working days	Table_Version_Group
LAST_RELOADED	Timestamp	A timestamp for when the table was last reloaded	Table_Version_Group
AUTO_REVALIDATE	Boolean	Whether or not a new version of the table should be automatically validated	Table_Version_Group
DATATYPE	String	The datatype of the object	Table, Function
COUNT	Integer	The number of arguments the function has	Function
CARDINALITY	String	The cardinality of the relationship	Relationship
DOCUMENTED_CARDINALITY	String	The documented cardinality of the relationship	Relationship
CARDINALITY_TYPE	String	The type of cardinality the relationship has	Relationship
LHS_DOMAIN_QUALITY	Decimal	The quality of the domain on the left-hand side of the relationship	Relationship
RHS_DOMAIN_QUALITY	Decimal	The quality of the domain on the right-hand side of the relationship	Relationship
DOMAIN_QUALITY	Decimal	The quality of the domain of the relationship has a whole	Relationship
LHS_JOIN_QUALITY	Decimal	The quality of the join on the left-hand side of the relationship	Relationship

RHS_JOIN_QUALITY	Decimal	The quality of the join on the right-hand side of the relationship	Relationship
JOIN_QUALITY	Decimal	The quality of the join on the relationship as a whole	Relationship
LHS_UNMATCHED_VALUES	Integer	The number of unmatched values in the left-hand column	Relationship
RHS_UNMATCHED_VALUES	Integer	The number of unmatched values in the right-hand column	Relationship
LHS_UNMATCHED_ROWS	Integer	The number of unmatched rows in the left-hand column	Relationship
RHS_UNMATCHED_ROWS	Integer	The number of unmatched rows in the right-hand column	Relationship
DUPLICATE_MATCHED_VALUES	Integer	The number of duplicate matched values in the relationship	Relationship
DUPLICATE_MATCHED_ROWS	Integer	The number of duplicate matched rows in the relationship	Relationship
COMMON_VALUES	Integer	The number of common values in the relationship	Relationship
VALUES_NOT_IN_COMMON	Integer	The number of values not in common in the relationship	Relationship
COMMON_ROWS	Integer	The number of common rows in the relationship	Relationship
TOTAL_VALUES	Integer	The total number of values in the relationship	Relationship
TOTAL_ROWS	Integer	The total number of rows in the relationship	Relationship
LHS_NULL_COUNT	Integer	The number of nulls in the left-hand side of the relationship	Relationship
RHS_NULL_COUNT	Integer	The number of nulls in the right-hand side of the relationship	Relationship
LHS_MATCHED_ROWS	Integer	The number of matched rows on the left-hand side of the relationship	Relationship
RHS_MATCHED_ROWS	Integer	The number of matched rows on the right-hand side of the relationship	Relationship
TOTAL_DUPLICATE_VALUES	Integer	The total number of duplicate values in the relationship	Relationship
TOTAL_DUPLICATE_ROWS	Integer	The total number of duplicate rows in the relationship	Relationship
LHS_TABLE	Object	The table on the left-hand side of the relationship	Relationship
RHS_TABLE	Object	The table on the right-hand side of the relationship	Relationship
USER_PROPERTY	Property	A user property on the object	All Objects
USER_PROPERTIES	Properties	A list of user properties on the object	All Objects
VERSIONED	Boolean	Whether or not the function is versioned	Function
EXPRESSION_ARGUMENTS	Object List	The arguments in the function	Function
IS_VARIABLE	Boolean	Whether or not the function is variable	Function
STATISTICS	Object List	The statistics for the table or column	Table, Column

getByName

static PProperty getByName(String value)

Returns the PProperty whose name matches the given **value**.

getKey

String getKey() Returns the property's key.

getPropertyType

PPropertyType getPropertyType() Returns the type of the property.

PTimestamp

This object defines a timestamp for when something happened, such as a table load.

getDate

Data getDate() Returns the Date object for this timestamp.

getTimestamp

long getTimestamp() Returns the timestamp in milliseconds.

toJSON

JSONObject toJSON() Returns a JSON representation of the timestamp.

PProductInfo

This object defines information about the product.

getBuildDate

String getBuildDate() Returns a String representation of the date when the product was built.

getCommonVersion

String getCommonVersion() Returns the version of the PCommon jar.

getCopyright

String getCopyright() Returns the copyright information for the product.

getDescription

String getDescription() Returns a description of the product.

getName

String getName() Returns the name of the product.

getServerVersion

String getServerVersion() Returns the version of the PServer jar.

getSupportEmail

String getSupportEmail() Returns the email address that any support queries should be sent to.

getTitle

String getTitle() Returns the title of the product.

toJSON

JSONObject toJSON() Returns a JSON representation of the product information.

RESTful API

The REST API is a web server embedded in the database server. It is predominantly to provide REST api access to data and metadata resources, but can also serve interactive pages (for example, sharing a quality report for a Table with another person who is not a user of the application).

The majority of resources require authentication.

Call Pattern

Resources can either be used atomically, passing credentials with each call, which is more in keeping with the REST ideology, or can be part of an overall session with a traditional login/logout. Note that a client concurrent license is consumed for the duration of the result only if the request is atomic, or for the entire duration of the session if the login/action.../logout pattern is used.

Authentication

All requests have the option of username and password parameters being supplied in the URI, or an auth key. The auth key is tied to a live session, created by using the **login** resource and is valid provided the session continues and the calls are from the same client IP address.

Although the REST engine allows connections with plain passwords, we do not recommend this is used other than in test environments. The login resource will return the encrypted password as well as the auth key, so future atomic requests can be performed using the encrypted password.

Should a resource request be interactive (pdf or html) and no credentials be supplied, the web server will invoke an html login screen and (optionally) store the credentials in an encrypted manner in a client-side cookie. Auth keys are not used for interactive sessions, instead the cookie is the favoured option.

General Conventions

Result tags for XML and JSON requests should all be viewed as optional. Boolean parameters are in general only presented when true (set). Their absence indicates they are false. This is a deliberate design decision in an attempt to minimise the size of XML and JSON result sets.

General REST URI Parameters

Many of the API requests respond to URI query parameters.

Parameter types

Type	Allowed values
Boolean	true, false, 1, 0, yes, no, y or n, case is insensitive
String	Any value
Integer	An integer number
Double	A decimal number
ObjectType	A enumeration of repository object types
QueryType	An enumeration of query types
IconType	An enumeration of the types of icon for the <i>icons</i> resource
SortOrder	A sorting order. Allowed values are: Ascending (asc) Descending (desc) Unsorted (none)
ImageState	The state of an image render. Allowed values are: DISABLED – Greyed out and dimmer ROLLOVER – Brighter for when mouse is rolled over the image NORMAL – Normally rendered image
StringList	A list of strings, comma separated
IntegerList	A list of integers, comma separated
PropertyList	A list of object properties
SqlQuery	A formal SQL query string
Path	A relative REST resource path
ExplorerType	An enumeration of the type of nodes for the <i>explorer</i> resource
Color	An HTML color string
AllowedValueList	A description of an allowed value, used for functions, which is a value with a descriptive name and a description.

Parameters used in all URIs

Parameter	Type	Usage
username	String	The username to use to log in to the server
password	String	The password to use to log in to the server
auth_key	String	The authorisation key given on login (used instead of username and password)

RESTful GET Resources

/manual

The /manual resource retrieves the application user manual in pdf format. This does not require authentication.

/api

The /api resource retrieves this manual in pdf format. This does not require authentication.

/explorer/node/subNode/...

This resource provides access to the items seen in the Experian Pandora Explorer interface, each of which is an explorer *node*. This is a tree of nodes, and hence a tree of REST resources, the base of which is the **explorer** resource which gives you the root of the node. These resources require authentication. The starting point is **/explorer**.

Supported Formats

- json, xml, text, txt

Supported Parameters

Parameter	Type	Usage
Expand	Boolean	Expand the results to include the children of the required node if there are any. Default is false.
Names	StringList	The list of names of child nodes to return, can be regular expressions. Default is to return all children. Setting this parameter implicitly sets expand to true.

Result Tags

Tag	Type	Description
can_drilldown	Boolean	Whether or not it is possible to drilldown on the node. This is to allow an application to display the node as enabled or not for drilldown.
Count	Integer	The number of children returned when expanded either using expand or names
description	String	A human readable description of the node.
display_name	String	The name of the node for display purposes.
drag_query	SqlQuery	The SQL query to run when the node is dragged onto the Experian Pandora desktop. This may differ to the <i>query</i> tag which is the SQL query that is executed when you drilldown on the node.
explanation	String	The text explanation for the query invoked from the node
foreground_color	Color	The foreground color
has_children	Boolean	Whether the node has children or not.
icon_type	IconType	The type of icon for the node. The image for this can be accessed using the <i>icons</i> resource.
Id	Integer	The id of the repository object represented by this node.
identifier	Integer	The unique identifier for the node. This is guaranteed to be unique across the entire hierarchy of nodes and can be used as a key to recall remembered nodes by the calling application.
Index	Integer	The index of the node, starting at zero. This is simply the position of the node at its current level in the hierarchy.
Name	String	The name of the node which may differ to the human readable <i>display_name</i> tag.
note_count	Integer	Number of notes attached to the repository object represented by this node.
obfuscated	Boolean	Denotes if the node is obfuscated (scrambled)
object_type	ObjectType	The type of repository object represented by the node.
parent_id	Integer	The id of the repository object represented by the parent node for this node.
Path	String	The relative url path to this node in the REST api
Query	SqlQuery	The SQL Query that results when drilling down to the node.
Type	ExplorerNodeType	The type of node.

/type/enumeration

This resource provides information on the enumerated types used in REST interactions. These resources require authentication.

Supported Formats

- json, xml, text, txt

Valid Enumerations

Name	Description
explorer	An enumeration of the explorer node types
expression_argument	An enumeration of types of arguments for functions
Icon	Types of icon
Job	An enumeration of job types for the scheduler
job_priority	An enumeration of job priorities for the scheduler
job_state	An enumeration of job states for the scheduler
Object	A repository type
Query	A query type name
Report	A type of report

Supported Parameters

Parameter	Type	Usage
name	String	The regular expression pattern to use when retrieving types in order to determine which results to return by filtering on type name
count	Integer	The maximum number of results to return

Result Tags (Object)

Tag	Type	Description
count	Integer	The number of objects of that object type in the repository
description	String	The description of the object type
name	String	The name of the object type
timestamp	Object	The timestamp an object of that type was last created or modified
type	String	The type of object

Result Tags (Query)

Tag	Type	Description
name	String	The name of the query type
display_name	String	The display name for the query type

Result Tags (Icon)

Tag	Type	Description
icon_type	String	The name of the icon type

Result Tags (Explorer)

Tag	Type	Description
always_has_folders	Boolean	This node type always has folders only below
always_leaf	Boolean	This node type is always a leaf
display_name	String	The display name for the explorer node type
name	String	The name of the explorer node type

Result Tags (Expression Argument)

Tag	Type	Description
display_name	String	The display name for the expression argument type
name	String	The name of the expression argument type

Result Tags (Global Query)

Tag	Type	Description
display_name	String	The display name for the global query
name	String	The name of the global query
query	String	The SQL query string for the global query

Result Tags (Job State)

Tag	Type	Description
description	String	The description of the job state
name	String	The name of the job state
send_completion	Boolean	True if the job should send a completion message on completion

Result Tags (Job Type)

Tag	Type	Description
class	String	The scheduler class for the job type
display_name	String	The display name for the job type
group	String	The name of the job group, itself a job type

Tag	Type	Description
name	String	The name of the job type

Result Tags (Job Priority)

Tag	Type	Description
name	String	The name of the job type

Result Tags (Report Type)

Tag	Type	Description
name	String	The name of the report type
description	String	The description for the report type
display_name	String	The display name for the report type

/function

This resource provides information about all of the functions, including custom functions.

Supported Formats

- json, xml, txt, text, html, pdf

Supported Parameters

Parameter	Type	Usage
name	String	The regular expression pattern to use when retrieving function information in order to determine which function(s) to display, by function name
library	String	The regular expression pattern to use when retrieving function information in order to determine which function(s) to display, by library

Result Tags

Tag	Type	Description
count	Integer	The number of arguments the function expects. This is not set if the function is variable.
datatype	String	The datatype for the return value from the function, or AUTO if not strictly typed
description	String	The description for the function
display_name	String	The display name for the function
expression_arguments	ObjectList	A list of argument objects for the function
id	String	The id for the function
is_variable	Boolean	Whether the function has a variable number of arguments
library	Object	Name and description of the parent function library

Tag	Type	Description
name	String	The function name
parent	String	The parent library name for the function
type	String	INTERNAL for internal functions or CUSTOM for glossary user-defined ones
versioned	Boolean	Whether the function has versions

The **expression_arguments** could have the following result tags:

Tag	Type	Description
allowed_values	AllowedValueList	An object describing the value, name and description for allowed values
auto_populate	Boolean	Whether the argument should be auto-populated
datatypes	StringList	The possible datatypes for the argument
description	String	The description of the argument
is_column_reference	Boolean	Whether the argument is a reference to a column
is_lookup_table	Boolean	Whether the argument is a lookup table
is_property_key	Boolean	Whether it is a property key
is_reference	Boolean	Whether the argument is a reference
is_validation_arg	Boolean	Whether it is a validation argument
name	String	The name of the argument
nullable	Boolean	Whether the argument is allowed to be null
type	ArgumentType	The type of the argument
verbatim	Boolean	Whether the argument should be treated verbatim and not auto-casted

/function/execution/function_name

This resource provides access to execution of custom functions in the glossary from external applications. The name of the function is unique in the glossary so can be referenced without a qualifying function library name.

The parameters are named the same as the parameter names for the function itself, in lower case with underscores instead of spaces. The values of the parameters are taken verbatim and are typeless. For alpha numeric values you should encapsulate them with single quotes. To include a single quote within the value, you should put two single quotes (i.e. escape the quote with another quote). This is standard SQL syntax.

Either transformation or validation functions can be invoked with this mechanism.

Supported Formats

- json, xml, text, txt

Result Tags

Tag	Type	Description
background_color	Color	The HTML colour for the background
datatype	String	The datatype of the value
foreground_color	Color	The HTML colour for the foreground
html_value	String	The value in html if it is subject to formatting
icon_type	IconType	The type of icon for the value if there is one
is_blank	Boolean	The value returned is empty (blank)
is_error	Boolean	Denotes the result value is an error
is_null	Boolean	The value returned is NULL
is_warning	Boolean	Denotes the result value is a warning
length	Integer	The length of the value
message	String	The warning or error message return if it is a warning or error
obfuscated	boolean	True if the value is obfuscated (scrambled)
render_mode	RenderMode	The rendering mode for the value for presentation purposes. Only set if different to the default, which is normal text in colour.
value	String	The value in string form

/icon/icon_type

This resource provides the images for all of the available icon types. It does not require authentication. Default is 16x16, normal image state and png format. A full list of icon types can be obtained from the [/type/icon](#) resource.

Supported Formats

- bmp, png, jpg, gif

Note: **png** is the only format that supports transparency

Supported Parameters

Parameter	Type	Usage
size	Integer	The size of the image from 8 to 256 pixels square
render	ImageState	The state of the image to be rendered. Default is <i>NORMAL</i>

The possible values for **render** are: NORMAL, ROLLOVER and DISABLED.

/global

This resource is a synonym for [/type/global_query](#)

/global/global_query_type

This resource provides access to global query definitions that report on aspects of the server that are independent of a specific context (e.g. All Tables, All Columns etc.). A full list of *global_query_type* definitions can be obtained from the */type/global_query* resource.

Supported Formats

- json, xml, txt, text, html, pdf

Result Tags

Tag	Type	Description
display_name	String	The display name for the global query type
name	String	The name of the global query type
query	String	The SQL query string for the global query

/images/

This resource is provided for the generation of dynamic images produced in, for example, mapping reports which are not fixed objects in the REST schema, but are on-demand generated. The images themselves are created in the REST_IMAGE directory of the server and given a name relative to the user's session. There is currently no intelligent caching of images. It is not really possible to use this resource directly, more that it will be used indirectly during dynamic generation of html.

Supported Formats

- bmp, png, jpg, gif

/login

This resource allows a user to log in to the REST API. If the specified format is interactive (html or pdf) the user will be presented with an HTML login if no credentials are provided on the command line. The response from this resource is currently information about the user logged in.

Supported Formats

- json, xml, txt, text, html, pdf

Result Tags

Tag	Type	Description
display_name	String	The display name of the User
id	Integer	The internal id of the User
identifier	String	The identifier for the User's REST Session (to be used as the Auth Key)
name	String	The name of the User
password	String	The encrypted password for the user

/logout

This resource allows a user to log out of the REST API

Supported Formats

- json, xml, txt, text, html, pdf

Result Tags

Tag	Type	Description
Message	String	A Goodbye message to the user

/object/object_type

This resource allows provides access to the majority of repository objects. The resource can be called without an object_type to return the *entire* repository in a single call. For a list of object types use the **/type/object** resource.

Supported Formats

- json, xml, txt, text, html, pdf

Supported Parameters

Parameter	Type	Usage
child_match	Boolean	Only return child objects that had properties that matched the Child_Property query parameter (if used)
child_property	String	Specifies the list of properties to return for children if returning children
children	String	Specifies the list of object types of children that should be returned. Default is not to return children.
default_query	Boolean	The default SQL query string
id	Integer	Returns objects of the requested type with a repository id matching one of the ids in the list.
match	StringList	Only return objects that had properties that matched the Property query parameter (if used)
name	String	Returns only objects of the requested type whose display_name attribute matches this supplied regular expression. Default is to match all.
order	SortOrder	Controls the sort order of returned objects using their name for the sort.
property	String	Returns only the specified object properties. This can be a list of specific properties and/or regular expression patterns.
user_property	String	Specifies a list of repository object properties to return. * denotes all as does the lack of this parameter.
user_property_match	Boolean	Only return objects that had properties that matched the User_Property query parameter (if used)
object_types	ObjectTypeList	List of types to return when returning all objects

Result Tags (All Objects)

Tag	Type	Description
child_count	Integer	The number of children the object has
created	User Action	Information about the creation of the object. Including the name and id of the user who created it, the reason why they created it and the timestamp to show when it was created.
description	String	The object's description
display_name	String	The display name of the object
id	Integer	The id of the object

Tag	Type	Description
last_modified	User Action	Information about the last time this object was modified. Including the name and id of the user that modified it, the reason why they modified it and the timestamp to show when it was modified.
name	String	The name of the object
note_count	Integer	Number of notes attached to the object
parent	Integer	The id of the object's parent
permissions	Permissions	The access rights for the object
steward	String	The user that stewards this object
steward_group	String	The group who steward this object
user_property	Property	A user defined property on the object

Result Tags (Business Constants)

Tag	Type	Description
value	String	The value of the business constant

Result Tags (Business Terms)

Tag	Type	Description
abbreviations	StringList	A list of abbreviations for the business term
broader_term	String	General class to which the term belongs
definition	String	The definition of the business term
example	String	An example of the use of the business term
last_status_change	User Action	The last status change
related_terms	Business Term	Terms that are related to the business term
related_columns	Column List	Columns that are related to the business term
sentence	String	A sample sentence that contains the term
source	String	Where the business term comes from
status	String	The status of the business term
synonyms	StringList	Synonyms for the business term
usage	String	Description of how the item defined by the business term is used

Result Tags (Columns)

Tag	Type	Description
atypical_datatypes	StringList	Any present datatypes that are not the documented datatype
auto_parse_dates	Boolean	Whether date values in this column should be automatically parsed
auto_parse_money	Boolean	whether money values in this column should be automatically parsed
auto_remove_erroneous_quotes	Boolean	Whether erroneous quotes should be automatically removed
average	Object	The average of the values in the column
average_format_frequency	Double	The average of the frequencies of all formats on the column
average_frequency	Double	The average frequency of values on the column
average_length	Integer	The average of the lengths of all the values in the column
blank_count	Integer	The number of blank values
blanks_treated_as_null	Boolean	Whether blank values are treated as nulls
broken_key	Boolean	Whether or not the column is a broken key
checksum	Integer	The checksum for the column
completeness	Double	The completeness of the column
count	Integer	The number of values in the column
datatype	String	The predominant datatype in this column
decimals_are_standardised	Boolean	Whether decimals are standardised
display_steward	String	The display name of the steward
display_steward_group	String	The display name of the steward group
distribution	Double	The distribution of values in the column
documented_format	String	The documented format
documented_length	Integer	The documented length
documented_max	Object	The documented maximum value
documented_min	Object	The documented minimum value
documented_precision	Integer	The documented precision
documented_scale	Integer	The documented scale
documented_type	String	The documented datatype
external_name	String	The external name of the column
false_text	String	The text to show for a false value
format_frequency_deviation	Double	The standard deviation of format frequencies
frequency_deviation	Double	The standard deviation of value frequencies
high_format_frequency	Boolean	Whether the column has an abnormally high format frequency
high_frequency	Boolean	Whether the column has an abnormally high value frequency
high_number	Boolean	Whether the column has an abnormally high number
high_validation_threshold	Integer	The threshold between an ok result and a passed result in validation
index	Integer	This column's index
integers_are_standardised	Boolean	Whether integers are standardised
is_atypical	Boolean	Whether or not the column has more values of a different datatype than expected

Tag	Type	Description
last_validation_time	Timestamp	A timestamp for the last time the column was validated
last_validation_user	String	Information about the user who last validated this column
leading_spaces_removed	Boolean	Whether leading spaces should be removed
leading_zero_as_number	Boolean	Whether leading zeros should be counted as a number
least_common	Object	The least common value in the column
least_common_count	Integer	The number of times the least common value occurs
length_deviation	Double	The standard deviation of value lengths of the values in the column
length_sum	Integer	The sum of the lengths of the values in the column
length_sum_squared	Integer	The sum of the lengths of the values in the column, squared
long_values	Boolean	Whether the column has long values
longest_length	Integer	The length of the longest value in the column
low_format_frequency	Boolean	Whether the column has an abnormally low format frequency
low_frequency	Boolean	Whether the column has an abnormally low value frequency
low_nulls	Boolean	Whether the column has a low distribution of null values
low_number	Boolean	Whether the column contains an abnormally low number
low_validation_threshold	Integer	The threshold between a failed result and an ok result in validation
max	Object	The maximum value in the column
max_count	Integer	The number of times the maximum value occurs
max_expected_format_frequency	Double	The largest number of duplicate formats expected
max_expected_frequency	Double	The largest number of duplicate values expected
max_expected_length	Integer	The longest a value is expected to be
max_expected_number	Double	The maximum expected numeric value
min	Object	The minimum value in the column
min_count	Integer	The number of times the minimum value occurs
min_expected_format_frequency	Double	The smallest number of duplicate formats expected
min_expected_frequency	Double	The smallest number of duplicate values expected
min_expected_length	Integer	The shortest a value is expected to be
min_expected_number	Double	The minimum expected numeric value
most_common	Object	The most common value in the column
most_common_count	Integer	The number of times the most common value occurs
most_common_format	String	The most common format in the column
most_common_format_count	Integer	The number of times the most common format in the column occurs
native_type	String	The native type of this column

Tag	Type	Description
negative_count	Integer	The number of negative values
note_count	Integer	The number of notes attached to this column
null_count	Integer	The number of nulls in this column
null_distribution	Double	The distribution of null values
precision	Integer	The precision of the column
prevalent_type	String	The prevalent type of the column
relationship_count	Integer	The number of relationships that involve this column
row_count	Integer	The number of rows in this column
scale	Integer	The scale of the column
scientific_numbers_are_standardised	Boolean	Whether scientific numbers are standardised
score	Double	The validation score for this column
sequence	String	The sequence of the column
short_values	Boolean	Whether the column has abnormally short values
shortest_length	Integer	The length of the shortest value in the column
standard_deviation	Double	The standard deviation of values in the column
standard_deviation_multiplier	Double	The standard deviation multiplier on the column
sum	Object	The sum of the values in the column
sum_squared	Double	The square of the sum of the values in the column
sum_squared_of_format_frequency	Double	The sum of the frequencies of the formats in the column, squared
sum_squared_of_frequency	Double	The sum of the frequencies of the values in the column, squared
table_id	Integer	The id of this column's table
table_name	String	The name of this column's table
trailing_spaces_removed	Boolean	Whether trailing spaces should be removed
true_text	String	The text to show for a true value
uncommon_divisor	Integer	The uncommon divisor of the column
unique_formats	Integer	The number of unique formats in the column
unique_key_threshold	Integer	The unique key threshold of the column
unique_values	Integer	The number of unique values in the column
values_were_tokenised	Boolean	Whether or not the values were tokenised
volume_score	Double	The volume score for this column
zero_count	Integer	The number of zero values

Result Tags (Dependencies)

Tag	Type	Description
both_null_count	Integer	The number of times both sides of the dependency are null
coverage	Integer	The coverage of the dependency
rhs_column	Integer	The right-hand column in the dependency
singleton_null_count	Integer	The number of times one side or the other, but not both, contains a null.

Result Tags (Domains)

Tag	Type	Description
filename	String	The name of the file that contains the domain information
is_sensitive	Boolean	Whether the content of the domain is sensitive
skip_first_row	Boolean	Whether the first row should be skipped (As it is titles)
type	String	The type of domain

Result Tags (Functions)

Tag	Type	Description
expression_arguments	Object Kist	A list of expression arguments objects
count	Integer	The number of arguments passed
datatype	String	The datatypes that are compatable with this function
description	String	The description name of the function
display_name	String	The display name of the function
id	Integer	The id of the function
is_variable	Boolean	Whether this function has a variable number of arguments
name	String	The name of the function
parent	Object	The parent function
type	String	The type of function
versions	Object List	A list of the versions of this function
versioned	Boolean	Whether the function is versioned

Result Tags (Keys)

Tag	Type	Description
error_count	Integer	Number of errors
is_approximate	Boolean	The key is approximate
is_sampled	Boolean	Keys have been calculated using a sampled dataset
lhs_columns	Object List	A list of the columns that form the key
null_count	Integer	Number of nulls
quality	Integer	The quality of the calculated key in percent
tested_quality	Integer	The tested quality of the key.

Result Tags (Notes)

Tag	Type	Description
assigned_by	Integer	Who the note was assigned by
assigned_to	Integer	Who the note has been assigned to
assigned_when	Integer	When the note was assigned
business_impact	Integer	The business impact of the note
note_id	Integer	The id of the note
noted_object	Integer	The id of the object that the note is on
technical_impact	Integer	The technical impact of the not

Result Tags (Note Details)

Tag	Type	Description
author	User	The user who created the note detail
created	User Action	The action that describes the creation of the note detail
html_text	String	The html representation of the text in the note detail
index	Integer	The index of the note detail within its parent note
last_modified	User Action	The action that describes the last time the note detail was modified
query	String	The query attached to the note detail
query_description	String	The description of the query
text	String	The text in the note detail, as plain text
version	Integer	The version number for the note detail

Result Tags (Queries)

Tag	Type	Description
type	String	The type of object this query will work for
query	String	The query

Result Tags (Relationships)

Tag	Type	Description
cardinality	String	The cardinality of the join
cardinality_type	String	The join's cardinality type
common_rows	Integer	The number of common rows
common_values	Integer	The number of values in common
documented_cardinality	String	The documented cardinality
domain_quality	Double	Quality of the value intersection with respect to both sides
duplicate_matched_rows	Integer	Number of rows that matched that have values that were duplicated
duplicate_matched_values	Integer	Number of values that matched that have duplicates
join_quality	Double	Quality of the join with respect to both sides
lhs_column	Object	The name and id of the column on the left hand side of the join
lhs_domain_quality	Double	Quality of the value relationship with respect to the lhs
lhs_join_quality	Double	Quality of the join with respect to the lhs
lhs_matched_rows	Integer	The number of matched rows in the left hand side
lhs_null_count	Integer	The number of null values in the left hand side
lhs_table	Object	The name and id of the table on the left hand side of the join
lhs_unmatched_rows	Integer	The number of unmatched rows in the left hand side

Tag	Type	Description
lhs_unmatched_values	Integer	The number of unmatched values in the left hand side
rhs_column	Object	The name and id of the column on the right hand side of the join
rhs_domain_quality	Double	Quality of the value relationship with respect to the rhs
rhs_join_quality	Double	Quality of the join with respect to rhs
rhs_matched_rows	Integer	The number of matched rows in the right hand side
rhs_null_count	Integer	The number of null values in the right hand side
rhs_table	Object	The name and id of the table on the right hand side of the join
rhs_unmatched_rows	Integer	The number of unmatched rows in the right hand side
rhs_unmatched_values	Integer	The number of unmatched values in the right hand side
status	String	The status of the relationship
total_duplicate_rows	Integer	The total number of duplicate rows
total_duplicate_values	Integer	The total number of duplicate values
total_rows	Integer	The total number of rows
total_values	Integer	The total number of values
values_not_in_common	Double	The number of values not in common across both sides

Result Tags (Tables)

Tag	Type	Description
checksum	Integer	The checksum of the table
child_count	Integer	The number of children the table has
code_page	String	The character set name used when the table was loaded
column_count	Integer	The number of columns in the table
datasource	Integer	The id and name of the datasource for the table
dependency_count	Integer	The number of dependencies that have been discovered on the table
dependency_level	Integer	The number of columns on the lhs when last testing for dependencies
dependency_threshold	Integer	The quality threshold used when testing for dependencies
description	String	The description of the table
display_name	String	The display name of the table
display_steward	String	The display name of the user who stewards the table
display_steward_group	String	The display name of the group of the user who stewards the table
external_name	String	The external name of the table
high_validation_threshold	Integer	The threshold between an ok result and a passed result in validation
key_count	Integer	The number of discovered keys on the table

Tag	Type	Description
key_level	Integer	The number of columns on the lhs when last testing for keys
key_threshold	Integer	The quality threshold used when testing for keys
last_dependency_time	Timestamp	The last time a dependency was discovered on the table
last_dependency_user	String	The last user to discover a dependency on the table
last_key_time	Timestamp	The last time a key was discovered on the table
last_key_user	Object	The last user to discover a key on the table
last_validation_time	Timestamp	The last time the table was validated
last_validation_user	Object	The last user who validated the table
load_duration	Integer	The time, in milliseconds, that it took to load the table
low_validation_threshold	Integer	The threshold between a failed result and an ok result in validation
name	String	The name of the table
note_count	Integer	The number of notes attached to the table
relationship_count	Integer	The number of relationships that involve the table
row_count	Integer	The number of rows in the table
schema	String	The name of the containing schema
schema_id	Integer	The id of the containing schema
score	Double	The validation score
steward	String	The name of the user who stewards the table
steward_group	String	The name of the group for the user who stewards the table
storage_type	String	The storage method used for the table
subject_area	Object	The schema that contains the table
table_group	Object	The group of tables that the table belongs to
table_group_id	Integer	The id of the group of tables that the table belongs to
table_type	String	The type of the table
unique_formats	Integer	The number of unique formats in the table
unique_values	Integer	The number of unique values in the table
validation_status	String	The status of validation on the table (RED for fail, AMBER for ok and GREEN for pass)
version	Integer	The version of the table

Result Tags (Table Version Groups)

Tag	Type	Description
auto_revalidate	Boolean	Whether or not to revalidate the table automatically when a new version is loaded
last_reloaded	Integer	The last time this table was reloaded
reload_method	String	The method to use to automatically reload the table. If manual then reloading is not done automatically.
reload_period	String	How often to reload the table (if method is not manual)
source	String	The name of the source file for the table group
versions	Object List	The number of versions contained in this group and the id and name of each table contained in this group.
versions_to_archive	Integer	The number of versions to archive before automatically deleting the oldest archived version
versions_to_keep	Integer	The number of versions to keep before automatically archiving the oldest version
working_days_only	Boolean	Whether automatic reloading should only be done on working days

Result Tags (Users)

Tag	Type	Description
current_project	Integer	The user's current project
email	String	The user's email address
is_admin	Boolean	Whether the user is an administrator
is_disabled	Boolean	Whether the user is allowed to access Experian Pandora
last_logged_in	String	A String representation of the last time the user was logged in, which will show "Never" if they have never logged in
last_logged_in_time	Timestamp	The last time the user was logged in
phone1	String	The user's primary phone number
phone2	String	The user's secondary phone number
role	Integer	The user's role
status	String	The status of the user

Result Tags (Views)

Tag	Type	Description
column_count	Integer	The number of columns in the view
query	String	The query that makes up the view
row_count	Integer	The number of rows in the view

object/object_type/object_name/mapping

This resource provides the mapping hierarchy when the format is json or xml and a mapping document when the format is txt, text, html or pdf. Only relevant for Table and View object types.

Supported Formats

- json, xml, txt, text, html, pdf

Result Tags (Mapping Hierarchy)

Tag	Type	Description
columns	ObjectList	Information about the columns
datasource_type	DataSourceType	Information about the datasource used to import the source or target, including the name, display name, description, index and icon type.
external_name	String	The external name of the source or target table
forced_split	Boolean	Whether this part of the hierarchy was subject to a forced split
internal_name	String	The internal name of the source or target table
is_lookup_table	Boolean	Whether the source or target is a lookup table
name	String	The name of the source or target table
object_id	Integer	The repository id for the source or target
object_type	ObjectType	The object type of the source or target
row_count	Integer	The number of rows in the source or target table
schema_name	String	The name of the Schema that contains this table
view_type	ViewType	Information about the view type of the source or target, including the name, display name and description

The value of **column** is a list of objects which could have the following tags:

Tag	Type	Description
datatype	Datatype	The documented datatype of the column
default_if_null	String	What the default is if there is a null value
description	String	The description of the column
english	String	English transformation if transformed
expression	String	The expression that describes the column
external_name	String	The external name of the column
group_index	Integer	The position the column takes in the grouping of the table, if it is involved in grouping
id	Integer	The repository object id for the column
index	Integer	The index of the column
is_lookup_column	Boolean	Whether the column is a lookup column
length	Integer	The maximum value length in the column
name	String	The name of the column
nullable	Boolean	Whether this column is allowed to have null values
original_column	Column	The details for the original column
precision	Integer	The largest number of figures in any numeric values in this column including both sides of the decimal point
scale	Integer	The largest number of figures in any numeric values in this column after the decimal point

Tag	Type	Description
<code>sort_index</code>	Integer	The position the column takes in the sorting of the table, if it is sorted
<code>sort_order</code>	SortOrder	The order in which the column was sorted, if it was
<code>source</code>	Boolean	Whether this column is a source
<code>source_id</code>	Integer	The id of this column's source
<code>sources</code>	StringList	The sources of this column
<code>sql</code>	String	SQL transformation if transformed
<code>type</code>	MappingType	The name, display name and description of the mapping type of this column
<code>view_name</code>	String	The name of the intermediary database view
<code>visible</code>	Boolean	Whether or not this column is visible in the UI

/object/object_type/object_name/children/child_object_type

This resource provides a list of children for the specified object, of the specified type.

Supported Formats

- json, xml, txt, text, html, pdf

Result Tags

Result Tags are the same as in the `objects/object_type` resources

/object/object_type/object_name

This resource provides details of a single named object for the specified object.

Supported Formats

- json, xml, txt, text, html, pdf

Result Tags

Result Tags are the same as in the `objects/object_type` resources

/object/object_type/object_name/list

Same as `/object/object_type/object_name`

/object/object_type/object_name/node

This returns the node information for the object in the same way as if it was located using the `/explorer` resource, but is independent of that structure, and hence does not have a **path** element to the result. See `/explorer` for result tags. This is useful for building user interfaces with navigation entirely independent of the explorer structure provided.

Supported Formats

- json, xml, txt, text, html, pdf

Result Tags

/object/object_type/object_name/quality

Returns the quality report for an object. Only relevant for Table and Table_Version_Group object types.

Supported Formats

- txt, text, html, pdf

/object/object_type/object_name/query/query_type

This resource returns the result of running the query of the specified type on the specified object. The result is an array of row objects, each of which contains an array of value objects.

Supported Formats

- json, xml, txt, text, html, pdf

Result Tags (Row)

Tag	Type	Description
id	Integer	The identifier for the row (row number)
value	ObjectList	A list of objects, one for each value in each column position of the row

Result Tags (Value)

Tag	Type	Description
datatype	String	The datatype of the current value
value	Object	The actual value
length	Integer	The length of the value
obfuscated	Boolean	Whether or not the value has been obfuscated
foreground_color	Color	The colour that the value should be rendered in
icon_type	IconType	The name of the icon that is associated with this value
index	Integer	The column index for the value

/object/object_type/object_name/report/report_type

Supported Formats

- text, txt, html, pdf

This resource returns the specified report. The possible values for report_type are:

- Mapping - Same as **/object/object_type/object_name/mapping**
- Quality - Same as **/object/object_type/object_name/quality**
- Relationship - Produces a relationship report, only valid for **relationship** objects
- Dashboard - Same as **/object/object_type/object_name/dashboard**
- Allowed - Returns a list of allowed reports for the object
-

/object/object_type/object_name/dashboard

Returns the quality dashboard for an object. Only relevant for Table, Table_Version_Group and Column object types.

Supported Formats

- txt, text, html, pdf

/object/object_type/object_name/query/allowed

Returns the query types allowed for this specific object.

Supported Formats

- json, xml, txt, text, html, pdf

/object/object_type/object_name/script/script_name

Returns the result of running the specified script, which should be in the **scripts** folder on the server.

Supported Formats

- json, xml, txt, text, html, pdf

/scheduler

This resource provides information about jobs running in the scheduler

Supported Formats

- json, xml, txt, html, pdf

Supported Parameters

Parameter	Type	Usage
name	String	The regular expression pattern to use when retrieving scheduler information. Defaults to all

Result Tags

Tag	Type	Description
count	Integer	The number of jobs in the scheduler

Result Tags (Job)

Tag	Type	Description
user	String	The type of job
name	String	The name of the job
group	String	The name of the job group
priority	JobPriority	The priority of the job. See /type/job_priority for a list of priorities
attempts	Integer	The number of times the job has tried to run
max_attempts	Integer	The maximum allowed number of times the job can run before permanent failure
start_time	Timestamp	The time the job started
scheduled_time	Timestamp	The time the job is scheduled to start
end_time	Timestamp	The time the job ended
progress	Integer	The progress from 0 to 100 for the job
state	JobState	The state of the job, see /type/job_state for a list of states
job_type	JobType	The type of job. See /type/job for a list of types