Web Services in Experian Pandora

This guide details advanced configuration, which may require Experian consultancy to configure and implement.

WARNING: Incorrect configuration or use can result in unpredictable results, application crashes, invalid data, web service credit usage and unsecured data. Experian will not accept responsibility for misuse.

Introduction

Experian Pandora has a rich suite of inbuilt data validation and transformation functions that can be used through the Experian Pandora user interface with tables of data. In the scenario where access to a function external to Experian Pandora is required, Experian Pandora provides two mechanisms for interoperability. One is the scripting engine described in the Scripting and REST guide, the second is the Call Web Service function available in Experian Pandora's function editor. In short, the Call Web Service function allows connections to most types of Web Services to be created through configuration alone, allowing users of Experian Pandora to access the Web Services directly as a Experian Pandora function. Additional functionality includes results caching to improve Web Service performance and reduce an organisation's costs; and exposing cached data for visualisation and searching outside of Experian Pandora. Any Web Service can be used that returns information in an organised and standardised structure. Experian Pandora can match JSON, XML, SOAP, plain text providing the response can be templated through configuration.

Enabling Web Services

In order to use Web Services in Experian Pandora there are several parts of configuration:

1. Host "file" – A json configuration file listing web services and parameters
2. Header, request and response templates – One set per web service
3. Experian Pandora configuration – Settings to enable functionality

The Host "file" is a json description of all Web Services configured for access by Experian Pandora - this essentially describes how Experian Pandora should present each Web Service to a user, how it should call them and if it should cache the results. This can be stored as a file, typically called 'host' (with no file extension – important), or provided by a url (e.g. http://www.example.com/host).

The json configuration is an object with two properties, the first records global properties and the second is an array of web services objects. The simplistic layout is below:

```
{
        "global_properties": {
        },
        "web_services": [
        ]
}
```

Below is a table of properties for each type of object:

Parent Object	Property	Datatype/ Value	Example	Description
<Root>.global_properties	connect_timeout	Integer	5000	Value in milliseconds each Web Service request should wait for connection before timing out.
<Root>.global_properties	response_timeout	Integer	20000	Value in milliseconds each Web Service request should wait for a response before timing out.
web_services[n]	name	String	EchoTest	ID of Web Service (should not contain spaces), not displayed to user.
web_services[n]	prompt	String		Plain text name of Web Service for user interface
web_services[n]	description	String		Description of Web Service for user interface
web_services[n]	method	GET or POST	GET	Which method to use to connect to the Web Service
web_services[n]	protocol	http or https	http	Which protocol to use to connect to the Web Service
web_services[n]	domain	String	scooterlabs.com	The domain name of the Web Service's URL (e.g. www.example.com)
web_services[n]	endpoint	String	echo.json	The endpoint of the Web Service (everything after the domain) e.g. example/endpoint/service. **Important note:** Parameters may be specified as part of the path in the form {{parameterID}}. In this case the specified parameter becomes part of the endpoint path and will not be included in the URL or POST body (see below).
web_services[n]	use_cache	Boolean	true	Whether the Web Service should store results in the cache (and retrieve from).
web_services[n]	cache_life	String	10d:0.2 or 10d	Required even if cache is not in use, specifies the length of time and variation of 'time to live' (ttl). Takes the form of a number, followed by a letter (m for minutes, h for hours or d for days). Optionally a colon and percentage expressed as a decimal can be appended. This will vary the ttl by a percentage, e.g. 10d:0.2 will mean all items placed in the cache will live somewhere between

				8 and 10 days.
web_services[n]	force_cache_clear	Boolean	false	Setting this to true will clear the Web Service's cache on the next restart of the Experian Pandora service.
web_services[n]	interactive	Boolean	true	Set this to true if the Web Service is expected to be fast enough to return results whilst a user is interactively using Experian Pandora. Setting this to false will only call the Web Service when a function is executed in a table export or similar.
web_services[n]	authorization	String	Basic OAuth https://example.com/token	**Optional (5.9.0 and later)** Basic and OAuth 2.0 (client credentials) are supported for service authorization. Use 'user' and 'secret' properties to set the credentials. If using OAuth 2 and credentials are required in the POST body (rather than in a header field) 'OAuth/Body' may be specified before the URL.
web_services[n]	user	String		**Optional (5.9.0 and later)** The user for authorization
web_services[n]	secret	String		**Optional (5.9.0 and later)** The secret (password) to use for authorization
web_services[n]	cache_filters	Array		An array of objects (details below). Cache filters allow different caching strategies to be applied based on the results from a Web Service. The last cache filter to match a result will apply, if none are met then the default cache life will be used instead.
web_services[n]	params	Array		An array of objects (details below). Params are the parameters a user needs to set to use the Web Service. Params for a GET request are added by name value pairs to the URL. Params for a POST request are parsed into the POST request body.
web_services[n]	static_params	Array		An array of objects (details below). Static Params are parameters that are needed to use a Web Service, but do not change and/or shouldn't be

				displayed to a user. Handling of static params are the same as params above.
web_services[n]	object_mapping	Object		Cache configuration in the form of an Elastic Search index mapping. (Beyond the scope of this guide).
web_services[n]	return_params	Array		An array of objects (details below). Return Params are the parameters that can be returned from a Web Service.
cache_filters[n]	param	String		The parameter ID/name to test. Should match the name of the return parameter.
cache_filters[n]	pattern	String		A regex that is used to test the return parameter.
cache_filters[n]	cache_life	String	2d:0.2 or 2d	The cache life to apply if the parameter's value matches the regex pattern. (Same format as cache life above)
params[n]	name	String		The ID of the parameter (should not contain spaces), not displayed to user.
params[n]	prompt	String		Plain text name of parameter for user interface
params[n]	description	String		Description of parameter for user interface
params[n]	validator	String	.{4,100}	A regex that can be used to test each parameter value. Used to perform basic validation on parameters and therefore minimise unnecessary function calls.
static_params[n]	param	String		The ID of the static parameter (should not contain spaces), not displayed to user.
static_params[n]	value	String		The value of the static parameter
return_params[n]	name	String		The ID of the return parameter (should not contain spaces), not displayed to the user).
return_params[n]	prompt	String		Plain text name of return parameter for user interface
return_params[n]	description	String		Description of return parameter for user interface

An example of a host file with one Web Service is listed below:

```
{
	"global_properties": {
		"connect_timeout": 5000,
		"response_timeout": 20000
	},
	"web_services":
	[
		{
			"name": "EchoTest",
			"prompt": "Echo Test Web Service",
			"description": "Echo test web service for demonstrating functionality",
			"method": "GET",
			"protocol": "http",
			"domain": "scooterlabs.com",
			"endpoint": "echo.json",
			"use_cache": true,
			"cache_life": "10d:0.2",
			"force_cache_clear": false,
			"interactive": true,
			"cache_filters": [
				{
					"param": "param1",
					"pattern": "test",
					"cache_life": "2d:0.2"
				}
			],
			"params": [
				{
					"name": "param1",
					"prompt": "Parameter 1",
					"description": "Parameter 1, which is between 4 and 100 characters long",
					"validator": ".{4,100}"
				},
				{
					"name": "param2",
					"prompt": "Parameter 2",
					"description": "Parameter 2, which is an alpha string between 2 and 3
characters long",
					"validator": "[A-Za-z]{2,3}"
				},
				{
					"name": "param3",
					"prompt": "Parameter 3",
					"description": "Parameter 3, which is an integer number between 0 and 4
digits long",
					"validator": "[0-9]{0,4}"
				}
			],
			"static_params": [
				{
					"param": "aStaticParameter",
					"value": "Static"
				}
			],
			"object_mapping": {
				"properties" : {
					"retParam1": {"type" : "string", "index" : "not_analyzed"},
					"retParam2": {"type" : "string", "index" : "not_analyzed"},
					"retParam3": {"type" : "string", "index" : "not_analyzed"},
					"yourIP": {"type" : "string", "index" : "not_analyzed"}
				}
			},
			"return_params": [
			 {
				"name": "retParam1",
				"prompt": "Return Parameter 1",
				"description": "The echoed parameter 1"
			 },
			 {
				"name": "retParam2",
				"prompt": "Return Parameter 2",
				"description": "The echoed parameter 2"
			 },
```

```
                                {
                                        "name": "retParam3",
                                        "prompt": "Return Parameter 3",
                                        "description": "The echoed parameter 3"
                                },
                                {
                                        "name": "yourIP",
                                        "prompt": "Your IP",
                                        "description": "The IP captured by the echo test, which should be yours."
                                }
                               ]
                        }
                ]
}
```

Web Service Stubs

In addition to the host file, each Web Service needs three configuration files:

1. Headers – contains header information for the Web Service
2. Request – contains the request body template for the Web Service
3. Response – contains the response body template for the Web Service

These files need to be in the same "location" as the Host file and need to be named as such (with no file extension):

1. host_<lowercase_name_of_service>_headers - e.g. host_echotest_headers
2. host_<lowercase_name_of_service>_request - e.g. host_echotest_request
3. host_<lowercase_name_of_service>_response - e.g. host_echotest_response

Note: If on a case-sensitive file system, the entire file name must be lowercase.

If the Host file has been served as a file from the file system, Experian Pandora will attempt to access these files (one set per Web Service) from the same directory. If the Host file has been served from a url Experian Pandora will attempt to also access these files via http (e.g. http://www.example.com/host would be followed by http://www.example.com/host_echotest_headers).

Headers

This file should contain a json object with a set of key value pairs for each header that needs to be added to the request (e.g. an authentication token), or an empty json object if none are required (or the Web Service is a GET based service).

Example 1 (no headers):

```
{
}
```

Example 2 (e.g. Experian Email Validate):

```
{
        "Auth-Token":  "MYPASSWORD"
}
```

Request

For a GET based service, this file should be empty but should still exist. For a POST based service, this should be the request body to be sent to the Web Service. For any dynamic parameters, Experian Pandora will insert either Params or StaticParams where it matches their ID in curly brackets. For example, a parameter appearing in the configuration with the name “param1”, if found in the template as {{param1}} will be replaced with the actual value of the parameter as configured by the user in Experian Pandora.

Example 1 (e.g. Experian Mobile Validate V3):

```
{
        "Number": "{{number}}"
}
```

Example 2 (notice the String parameter has quotation marks, the Integer parameter does not) (e.g. Experian Email Validate):

```
{
        "Email": "{{email}}",
        "Timeout": {{timeOut}}
}
```

Response

Similar to the request file, the response file contains a template that describes the response Experian Pandora can expect from a Web Service. Again, where a return parameter is expected, if its ID/name is placed in double curly brackets, Experian Pandora will attempt to extract its value from the response. In addition, there are two further tokens that can be used in the template to deal with varying responses from a Web Service. Any area of the response that is surrounded by $OPT OPT$ tags will be treated as optional (may or may not occur), and these can be nested. Any area of the response that has the token IGN will be treated as an area that should be ignored (for example, system generated information or time/date stamps – information that can't be matched). The response template must match the final character of a response (i.e. IGN cannot be placed as the last part of a template). Experian Pandora will attempt to match the Web Service response against this template. If it successfully matches, then it will return results. If it doesn't, then the results will not be available and therefore this is the key configuration aspect to test.

Example 1 (e.g. EchoTest):

```
{
  $IGN$
  "request": {
    "aStaticParameter": $IGN$,
    "param1": {{retParam1}},
    "param2": {{retParam2}},
    "param3": {{retParam3}}
  },
  $OPT"client_ip": {{yourIP}},OPT$
  $IGN$
}
```

Example 2 (e.g. Experian Email Validate):

```
{
  "Email": {{emailSubmitted}},
  "Certainty": {{certainty}},
  "Message": {{message}}$OPT,
  "Corrections": {{corrections}}OPT$
}
```

Example 3 (e.g. Experian Mobile Validate V3):

```
{
  "ResultCode": {{resultCode}},
  $OPT"AdditionalPhoneInfo": {
          "ValidatedPhoneNumber": {{validatedPhoneNumber}},
          "CountryName": {{countryName}},
          "CountryCode": {{countryCode}},
          "OperatorName": {{operatorName}},
          $OPT"PortedOperatorName": {{portedOperatorName}},
          "PortedCountryName": {{portedCountryName}},
          "PortedCountryCode": {{portedCountryCode}},OPT$
          "IsRoaming": {{isRoaming}},
          $OPT"RoamingNetworkName": {{roamingNetworkName}},
          $OPT"RoamingNetworkPrefix": {{roamingNetworkPrefix}},OPT$
          "RoamingCountryCode": {{roamingCountryCode}},OPT$
          "MCCMNC": {{MCCMNC}}
  },OPT$
  "Number": {{number}},
  "PhoneType": {{phoneType}},
  "Certainty": {{certainty}}$IGN$
}
```

Experian Pandora Configuration

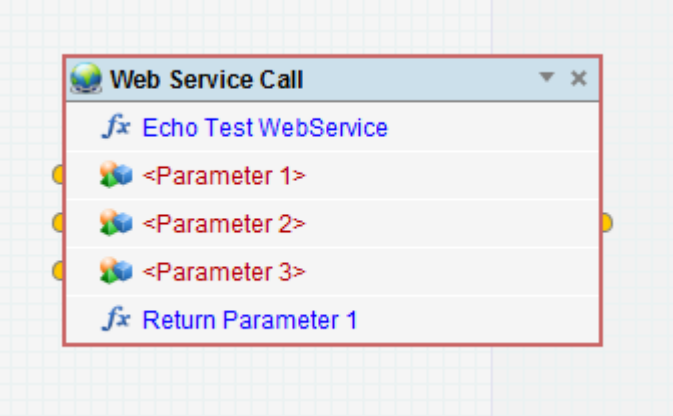
After configuring a host file and stubs for a Web Service, there are five Experian Pandora settings to configure. To configure these, open the Experian Pandora client and navigate to:

Settings > Server Settings > All Server Settings > Communication.

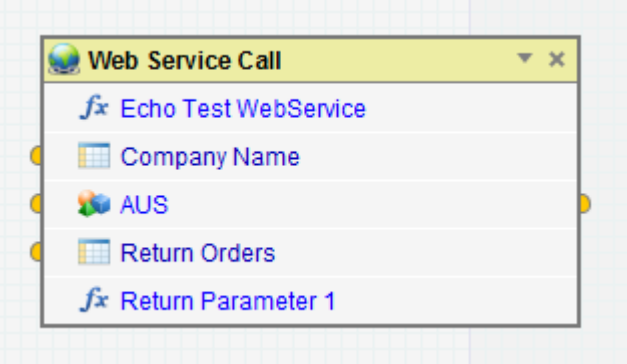
1. **Web Service Configuration Endpoint** – this is the url (e.g. http://www.example.com/host) or file path (e.g. file://C:\testdir\host) to the Host file. If this field is blank, then Experian Pandora won't have any enabled web services.
2. **Web Service Liveness** – this is a percentage setting that specifies what should happen if a Web Service starts failing during a data job (e.g. export). Each time a Web Service call fails this number will drop for that Web Service. If a job is currently running and the Web Service starts to fail once it reaches this cut-off point the job will be aborted. This is to avoid unexpected data corruption. The default is 100 (percent), so by default no failures are accepted. Each failure will decrease the 'health' by an increasing percentage amount (i.e. multiple failures are worse that single failures).
3. **Web Service Proxy** - allows the user to override any proxy settings configured in the host file and direct all Web Service calls through a specified proxy.
4. **Web Services Cache** – this setting enables Experian Pandora's embedded Elastic Search to cache Web Service results. This is highly recommended for increasing speed, however should only be used in a secure data environment as it will store Web Service results in a non-secure index.
5. **Web Services Cache External Access** – this setting enables http connectivity for the embedded Elastic Search cache. This will allow other Elastic Search Nodes to discover the cache, allow the cache to be searched externally by web service calls and allows Kibana (visualisation) to connect to the cache. This does however, reduce the security of data in the cache so should be carefully considered before enabling. This setting will not have any affect if the cache is not enabled.

Using Web Services in Experian Pandora

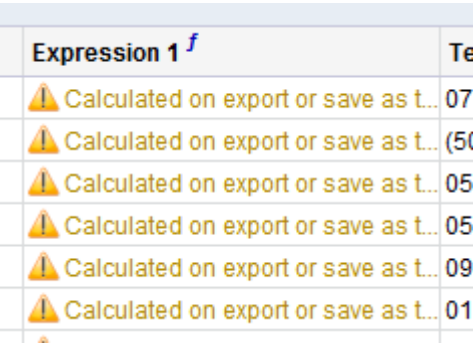
To use a pre-configured Web Service in Experian Pandora, simply insert a new custom function as part of a new column or transformation, select 'Web Service Call' from the list, populate the parameters, select the return parameter and finish the function:



Becomes:



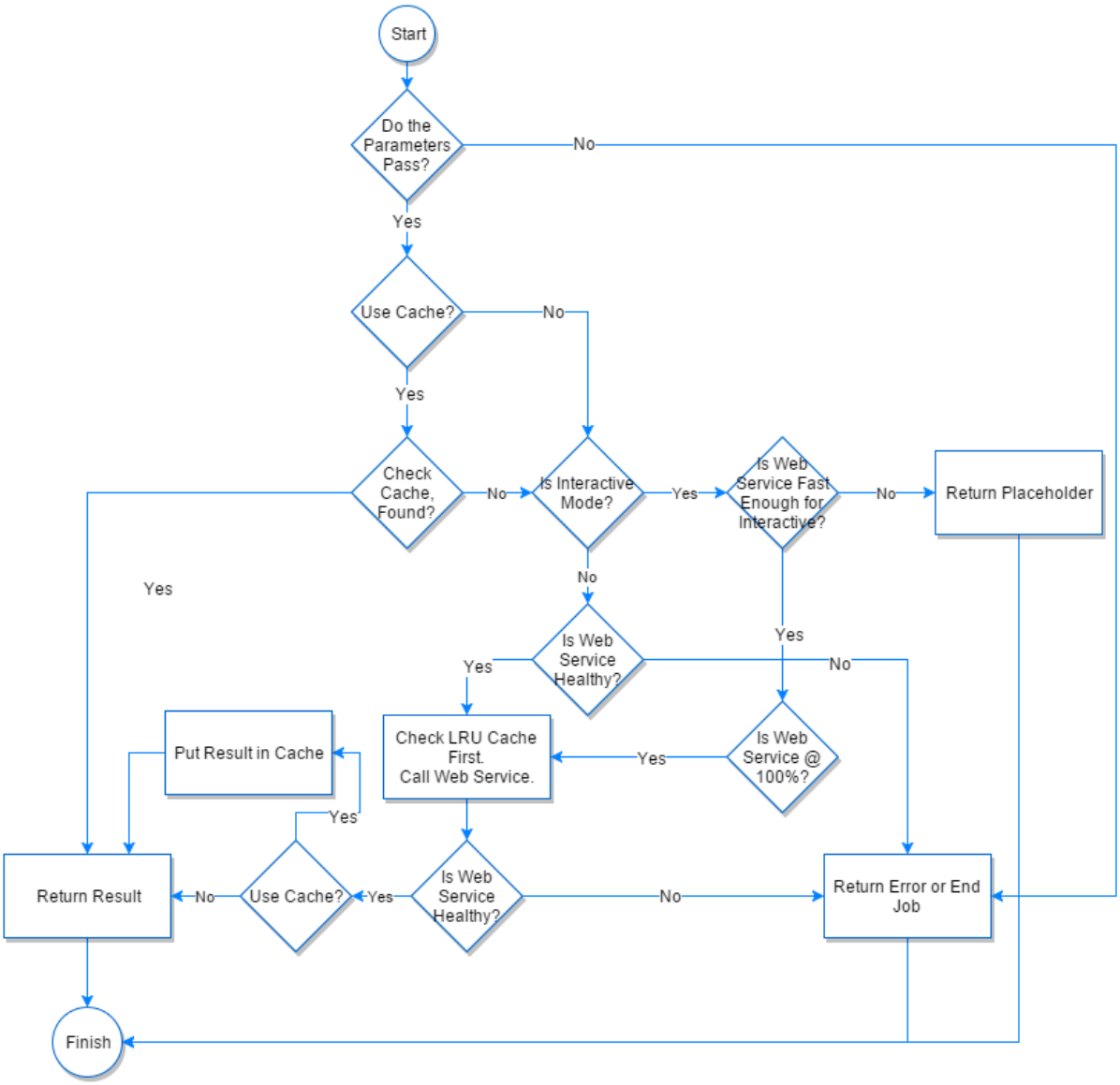
A Web Services function that hasn't been set to interactive will return a warning message indicating that it will only be calculated on export:



A Web Service that isn't responding will result in an error message instead.

Web Service Logical Flow

The following diagram details the logical decision flow for Web Services in order to minimise the number of external calls made and handle Web Service errors.



Visualising the Cache

If the cache has been exposed by http externally, Kibana can be used to visualise the contents of the cache. With Experian Pandora running, start Kibana and Kibana will auto-discover the cache. Kibana documentation will describe further how to start visualisations.