Web Services Reference

Psoda Web Services (PWS) allows you to interface your back-office systems to a Psoda server, including the public Internet server.
PWS uses a Remote Method Invocation (RMI) model using XML over HTML.

Table of content

Security

For security PWS uses a Secure Sockets Layer (SSL) connection which encrypts all messages between the client application and the Psoda server. This makes it safe to transmit the messages across the internet without fear of the data contained in the messages being intercepted by third parties.
To prevent a third-party from injecting extra packets into the data stream Psoda uses a PHP session ID and also checks the source IP address for requests.
Finally each session is authorised and authenticated using a username and password. The permissions allocated to that user will determine the data and functionality that will be available via PWS.

Basic message formats

All requests are initiated by the client similar to HTML requests. For each request the server will send back a response message containing details of the success or failure of the request. One HTTP message may contain multiple requests, each with a unique identifier. The HTTP response will contain a response message corresponding to each request.

Request messages

More than one request can be included in a single HTTP message. All request messages have the following structure:

```
<request id="[request id]" lang="[language]">
    …
</request>
```

The `id` will be returned as part of the response for this request so it should be unique to allow requests and responses to be matched up. This can easily be done by using sequential numbers for each request. The request ID does not have to be a number though so it may also contain a pre- or postfix.

The `lang` attribute is optional and can be used to change the language used for human-readable messages. Subsequent responses will use the same language until the `lang` attribute is set for another request. The current options are:

lang	description
uk	British English
us	American English
ja	Japanese
cn	Simplified Chinese
af	Afrikaans

Not all messages have been translated into all of the languages. Where a message is not available in the selected language the British English version will be used instead.

Response messages

Response message can consist of one or more responses, depending on the number of requests in the original message. Each response will have the following structure:

```
<responses>
   <response id="[request id]">
      …
   </response>
   …
</responses>
```

The `[request id]` corresponds to the identifier used in the original request.

Responses can contain a number of different types:

Object

An object response returns the type and object identifier for the selected object. It has the following format:

```
<response id="[request id]">
    <object type="[type]" obj_id="[obj_id]" />
</response>
```

The `type` attribute indicates the type of the object returned and the `obj_id` indicates the unique object identifier.

Object details

This response is typically returned when the details of a particular object have been requested:

```
<response id="[request id]">
    <object_details obj_id="[obj_id]" name="[name]"
                type="[type]" type_string="[type string]"
                [name1]="[value1]" [name2]="[value2]">
          <attribute name="[name3]">
                [long value3 with multiple lines]
          </attribute>
    </object_details>
</response>
```

The `type` attribute indicates the type of the object returned and the `obj_id` indicates the unique object identifier. The rest of the entries list the attributes of the selected object.

Boolean

A Boolean response basically contains a true/false value:

```
<response id="[request id]">
    <boolean value="[true|false]"/>
</response>
```

String

A string response contains a string value in response to the request:

```
<response id="[request id]">
    <string value="[string value]"/>
</response>
```

Array

An array response returns a number of results in one response. This is the typical case when requesting a list of objects. The response has the following structure:

```
<response id="[request id]">
    <array>
           <entry key="[key_1]">
                  <string value="[value_1]"/>
           </entry>
           <entry key="[key_2]">
                  <boolean value="[true|false]"/>
           </entry>
           <entry key="[key_3]">
                  <object obj_id="[obj_id_3]"/>
           </entry>
           <entry key="[key_4]">
                  <array>
                     |
                  </array>
           </entry>

           <entry key="[key_x]" >
                  <string value="[value_x]"/>
           </entry>
    </array>
</response>
```

Error

An error response to any request will have the following structure:

```
<response id="[request id]">
    <error code="[error code]" message="[text message]" />
</response>
```

In the response message the `[request id]` corresponds to the identifier of the original request, `[error code]` is the Web Services error code and `[text message]` is a human readable message corresponding to the error. The `[text message]` will be presented in the current Web Services language.

Not logged in

Any requests other than the Login request will fail with a NOT_LOGGED_IN error until the user has been successfully logged in, for example:

```
<response id="[request id]">
    <error code="NOT_LOGGED_IN" message="You have to log in before
you access that area." />
</response>
```

Access denied

If the user is logged in but is not allowed to access the requested area then an ACCESS_DENIED message will be returned, for example:

```
<response id="[request id]">
    <error code="ACCESS_DENIED" message="Sorry John, you do not
have the right privileges to access that area." />
</response>
```

Server error

Server-side errors will have an error code of `SERVER_ERROR`, for example:

```
<response id="[request id]">
    <error code="SERVER_ERROR" message="Could not connect to
database server." />
</response>
```

Specific request formats

Login request

No data access will be allowed until the Web Services has logged in. The login message has the following structure:

```
<request id="[request id]">
    <login username="[username]" password="[password]" />
</request>
```

A successful response to this request will have the following structure:

```
<response id="[request id]">
    <login_success user_obj_id="[obj_id]" />
</response>
```

The response message returns an `object` with a type of `user` and an object identifier of `[obj_id]`. This object corresponds to the logged in user based on the `username` and `password` passed in the request.

If the login failed then the response will contain an error indicator, for example:

```
<response id="[request id]">
    <error code="LOGIN_FAILED" message="No username/password match
in the database. [number] attempts in the last 5 minutes." />
</response>
```

A maximum of 3 login attempts are allowed from the same IP address within a 5 minute window. This prevents brute-force password cracking.

Logout request

Once the transactions have been completed the session can be closed by logging out:

```
<request id="[request id]">
    <logout/>
</request>
```

A successful response to this request will have the following structure:

```
<response id="[request id]">
    <logout_success/>
</response>
```

If the session is not logged in yet then the following error will be returned:

```
<response id="[request id]">
    <error code="NOT_LOGGED_IN" message="You have to log in before
you access that area." />
</response>
```

Object details request

This request is used to retrieve the details of the selected object. The request structure is:

```
<request id="[request id]">
    <get_object_details obj_id="[obj_id]"/>
```

```
</request>
```

If the user has the correct access to the selected object the response will have the following structure:

```
<response id="[request id]">
    <object_details obj_id="[obj_id]" name="[name]"
                type="[type]" type_string="[type string]"
                [name1]="[value_1]" [name2]="[value_2]">
          <attribute name="[name_3]">
                [long value3 with multiple lines]
          </attribute>
    </object_details>
</response>
```

The `type` attribute indicates the type of the object returned and the `obj_id` indicates the unique object identifier. The rest of the entries list the attributes of the selected object.

Object types

The following table lists the current object types supported in Psoda:

Type number	Type name
1	Organisation
2	Programme
3	Project
4	Product
5	Release
6	Release note
7	Baseline
8	Requirement
9	Test-case
10	User
11	Group
12	Relationship
13	ACL
14	Test step
15	Comment
16	Attachment
17	Risk
18	Notification
19	Test-run
20	Test-case result
21	Test-step result
22	Action
23	Change request
24	Issue
25	Report template
26	Report parameter
27	Defect
28	Workflow
29	Workflow state
30	Workflow transition
31	Feature
32	Milestone

Type number	Type name
33	Sub-project
34	Budget group
35	Budget item
36	Expense item
37	Task group
38	Task
39	Timesheet
40	Timesheet task
41	Benefit
42	Role
43	Holiday calendar
44	Leave application
45	Holiday
46	Dependency
47	Folder
48	Evaluation
49	Evaluation result
50	Vendor
51	Portfolio
52	Indicator
53	Indicator value
54	Material
55	Workflow action
56	Lesson
57	Whiteboard
58	Contract
59	Custom field
60	Exception
61	Assumptions
62	Decision

Lock object request

This request is used to lock the selected object before updating. The request structure is:

```
<request id="[request id]">
    <lock_object obj_id="[obj_id]"/>
</request>
```

If the user has the correct access to the selected object the response will have the following structure:

```
<response id="[request id]">
    <object_locked obj_id="[obj_id]"/>
</response>
```

The object will remain locked until it is unlocked using the `unlock_object` request below or up to 15 minutes after the last API interaction.

If the object is already locked by another user then the error response will have this structure:

```
<response id="[request id]">
    <error code="ALREADY_LOCKED" obj_id="[obj_id]"
          message="[object type] [object name] is already locked
for editing by [lock holder first name] [lock holder last name]"
</response>
```

Unlock object request

This request is used to unlock the selected object after updating. The request structure is:

```
<request id="[request id]">
    <unlock_object obj_id="[obj_id]"/>
</request>
```

If the request was successful the response will have the following structure:

```
<response id="[request id]">
    <object_unlocked obj_id="[obj_id]"/>
</response>
```

If the object was not locked by this API user account then the error response will have this structure:

```
<response id="[request id]">
    <error code="NOT_LOCKED" obj_id="[obj_id]"
          message="[object type] [object name] is not currently
locked by you"/>
</response>
```

Update object details request

This request is used to update the details of the selected object. Before you can update an object's details you have to lock it using the `lock_object` request above.

The request structure is:

```
<request id="[request id]">
    <update_object_details obj_id="[obj_id]"
          [name1]="[value_1]" [name2]="[value_2]">
          <attribute name="name_3">
                [long value3 with multiple lines]
          </attribute>
    </update_object_details>
</request>
```

A successful response will contain all the attributes that were updated:

```
<response id="[request id]">
    <object_updated obj_id="[obj_id]">
          <attribute name="[name_1]" value="[value_1]"/>
          <attribute name="[name_2]">
                [long value with multiple lines]
```

```
            </attribute>
            <attribute name="[name 3]" error="Value out of range"/>
            .
            .
            .
            <attribute name="[name_n]" value="[value_n]"/>
        </object_updated>
        .
        .
        .
</response>
```

If other objects are updated as a consequence of this update request then there will be one `<object_updated>` section for each of those other objects.

If the object was not locked by this API user account then the error response will have this structure:

```
<response id="[request id]">
    <error code="NOT_LOCKED" obj_id="[obj_id]"
          message="[object type] [object name] is not currently
locked by you"/>
</response>
```

Delete object request

This request is used to delete the selected object. Before you can delete an object you have to lock it using the `lock_object` request above.

The request structure is:

```
<request id="[request id]">
    <delete_object obj_id="[obj_id]"/>
</request>
```

A successful response will contain all the attributes that were updated:

```
<response id="[request id]">
    <object_deleted obj_id="[obj_id]"/>
</response>
```

If the object was not locked by this API user account then the error response will have this structure:

```
<response id="[request id]">
    <error code="NOT_LOCKED" obj_id="[obj_id]"
          message="[object type] [object name] is not currently
locked by you"/>
</response>
```

Create object request

This request is used to create a new object.

The request structure is:

```
<request id="[request id]">
    <create_object parent_obj_id="[parent_obj_id]" type="[type]"
          [name1]="[value_1]" [name2]="[value_2]">
          <attribute name="name_3">
                [long value3 with multiple lines]
          </attribute>
```

```
    </create_object>
</request>
```

A successful response will contain the obj_id for the newly created object:

```
<response id="[request id]">
    <object_created obj_id="[obj_id]"/>
    .
    .
</response>
```

If the object could not be created then an error response is returned:

```
<response id="[request id]">
    <error code="NOT_CREATED"
          message="[error message]"/>
</response>
```

Method call request

This request calls a specific method on the selected object:

```
<request id="[request id]">
    <call obj_id="[obj_id]" method="[method name]">
          <parameter name="[name_1]" value="[value_1]"/>
          <parameter name="[name_2]" value="[value_2]"/>

          <parameter name="[name_t]" value="[value_t]"/>
    </call>
</request>
```

The required parameters and the response message will depend on the specific method being called. Please refer to the method reference later on in this reference guide.

File upload requests

The following requests can be used to upload an attachment to Psoda.

To start the file upload send the following request:

```
<request id="[request id]">
    <start_file_upload obj_id="[destination obj_id]"
          filename="[file name]"
          mimetype="[mime type]"
          filesize="[file size in bytes]"
          blob="[1st portion of file base64 encoded]">
    </start_file_upload>
</request>
```

The API user will need access to create new attachments on the destination object. The response will look like this:

```
<response id="[request id]">
    <file_upload_started file_id="[file_id]"
          bytes_written="[bytes written]"/>
</response>
```

If the file is less than 8KBytes then it can be uploaded in this single request. Bigger files has to be split into multiple portions and subsequent portions can be uploaded using the following request:

```
<request id="[request id]">
    <continue_file_upload file_id="[file_id]"
```

```
          blob="[next portion of file base64 encoded]">
    </continue_file_upload>
</request>
```

The response will be:

```
<response id="[request id]">
    <file_upload_continued file_id="[file_id]"
          bytes_written="[bytes written for the last request]"/>
          total_bytes_written="[total bytes written]"/>
</response>
```

Once the complete file has been written to disk, i.e. total bytes written = file size, then the attachment is created and the final response will be:

```
<response id="[request id]">
    <file_upload_completed attachment_obj_id="[obj_id]"
          bytes_written="[bytes written for the last request]"/>
          total_bytes_written="[total bytes written]"/>
</response>
```

In this response the attachment_obj_id is the object ID for the newly created attachment.

An example session

The following sequence shows how an example session may progress.

Step 1 – Login

```
<request id="1">
    <login username="joebloggs" password="forgetmenot" />
</request>
```

The login was successful:

```
<response id="1">
    <object type="user" obj_id="4352" />
</response>
```

Step 2 - Request the user's details:

```
<request id="2">
    <object_details obj_id="4352"/>
</request>
```

The user's details are returned:

```
<response id="2">
    <object_details type="user" obj_id="4352">
          <attribute name="username" value="joebloggs"/>
          <attribute name="firstname" value="Joe"/>
          <attribute name="lastname" value="Bloggs"/>
          |
          <attribute name="logged_in" value="true"/>
    </object_details>
</response>
```

Step 3 – Get the user's organisation:

The user object is a child of the organisation object, so we are requesting the parent of the user object. The `getParent()` method does not have any parameters.

```
<request id="3">
```

```
    <call obj_id="4352" method="getParent"/>
</request>
```

The response contains the organisation's object identifier, 53:

```
<response id="3">
    <object type="organisation" obj_id="53" />
</response>
```

Step 4 – Get the organisation's details:

Use the organisation's object identifier, 53, to request all of the details of the organisation:

```
<request id="4">
    <object_details obj_id="53"/>
</request>
```

The organisation's details are returned:

```
<response id="4">
    <object_details type="organisation" obj_id="53">
          <attribute name="name" value="DemoOrg"/>
          <attribute name="address">
                15 Long Avenue
                Newtown
                Metropolis
          </attribute>
          <attribute name="country" value="USA"/>
          |
          <attribute name="fax" value="+1 234 567 1235"/>
    </object_details>
</response>
```

Step 5 –Update the organisation details:

```
<request id="5">
    <update_object_details obj_id="53">
          <attribute name="name" value="Acme"/>
          <attribute name="address">
                16 Long Avenue
                Newtown
                Metropolis
          </attribute>
    </update_object_details>
</request>
```

This user does not have permission to change the organisation's details:

```
<response id="5">
    <error code="ACCESS_DENIED" message="Sorry Joe, you do not have
the right privileges to access that area." />
</response>
```