## Input and Result Data

A request to a business application specifies the input data (parameters) to be processed by the business application. The business application then generates the result data. Both the input and result data can be either structured or opaque result data. Both of these are described in the following sections.

In addition to the per-request input data a business application can define additional parameters that apply to all requests in the job. These are batch job parameters.

# Java Example Application

The section provides an example of using the CPF to create a job containing a single request using structured input data to the MapTileByTileId business application and retrieving the structured result file.

The first step is to create a CpfClient instance to connect to the URL of the CPF web services using a consumer key and consumer secret.

String webServiceUrl = "**https://apps.gov.bc.ca/pub/cpf/ws/**";

String consumerKey = "...";

String consumerSecret = "...";

CpfClient client = new CpfClient(webServiceUrl, consumerKey, consumerSecret);

Then the getBusinessApplicationsNames method is called to check that the business application exists. The getBusinessApplicationVersions is called to get the version number of the most recent version.

String appName = "**MapTileByTileId**";

List<String> appNames = client.getBusinessApplicationNames();

if (appNames.contains(appName)) {

List<String> versions = client.getBusinessApplicationVersions(appName);

String version = versions.get(0);

:

} else {

throw new IllegalArgumentException("Business Application Not Found");

}

The job and request parameters for a single request can be specified in a single map.

Map<String, Object> parameters = new LinkedHashMap<String, Object>();

parameters.put("**mapGridName**", "**BCGS 1:20 000**");

parameters.put("**mapTileId**", "**92g025**");

The job can be created using the createJobWithStructuredSingleRequest method, which returns the URL to the job status page. There are other methods for creating jobs with opaque data and with multiple requests. These are described in the web service API section.

String jobUrl = client.createJobWithStructuredSingleRequest(

appName,

version,

parameters,

"application/json"

);

The getJobStructuredResults methods can be used to get the structured result file and return a reader to read each result as a Map. The second parameter is a timeout to wait for the results to be

Reader<Map<String, Object>> results = client.getJobStructuredResults(

jobUrl,

100000

);

try {

for (Map<String, Object> result : results) {

System.*out*.println(result.get("**mapTileBoundary**"));

}

} finally {

reader.close();

}

The batch job status page can be obtained using the getJobStatus method.

Map<String, Object> jobStatus = client.getJobStatus(jobUrl);

System.*out*.println(jobStatus.get("numCompletedRequests"));

The following shows the full code for the example application.

String webServiceUrl = "**https://apps.gov.bc.ca/pub/cpf/ws/**";

String consumerKey = "...";

String consumerSecret = "...";

CpfClient client = new CpfClient(webServiceUrl, consumerKey, consumerSecret);

String appName = "**MapTileByTileId**";

List<String> appNames = client.getBusinessApplicationNames();

if (appNames.contains(appName)) {

List<String> versions = client.getBusinessApplicationVersions(appName);

String version = versions.get(0);

Map<String, Object> parameters = new LinkedHashMap<String, Object>();

parameters.put("**mapGridName**", "**BCGS 1:20 000**");

parameters.put("**mapTileId**", "**92g025**");

String jobUrl = client.createJobWithStructuredSingleRequest(

appName,

version,

parameters,

"application/json"

);

Reader<Map<String, Object>> results = client.getJobStructuredResults(

jobUrl,

100000

);

try {

for (Map<String, Object> result : results) {

System.*out*.println(result.get("**mapTileBoundary**"));

}

} finally {

reader.close();

}

Map<String, Object> jobStatus = client.getJobStatus(jobUrl);

System.*out*.println(jobStatus.get("numCompletedRequests"));

} else {

throw new IllegalArgumentException("Business Application Not Found");

}

# Web Service API Reference

The CPF application can be accessed using a HTML interface via a web browser or as a HTTP web service from a custom application developed using JavaScript, Java or other programming language. The same AP

This section describes the details of each web service interface provided in the CPF API.

All of the examples assume that the CPF web services are accessed from the BC Government CPF server. If a different server is used consult the owner of that server to find the correct URL to access that CPF instance. All paths in the API are relative to this URL and must be prefixed with this base URL. The JavaScript and Java clients require this URL in the constructor and add the appropriate web service path as required.

## Constructing a Client Instance

### Web Service API

A specific CPF client is not required for direct access to the Web Service API.

If the OAuth secured web services are used developers will need to implement their own library to build the request and perform the required OAuth authentication.

### JavaScript

The CPF JavaScript client can be constructed by creating an instance of the CpfClient class from the cpf\_client.js library. The constructor takes a URL to the CPF web services

var url = '**http://apps.gov.bc.ca/pub/cpf/ws/secure**';

var client = new CpfClient(url);

**NOTE:** The CPF JavaScript client does not accept a consumer key and consumer secret as an argument. The end user will be prompted using either a HTML form or pop-up login for their login credentials. The CPF client ensures that the user is logged in before each web service request. This helps prevent issues when the login times out. The CPF JavaScript client is therefor run using the end-user's login credentials.

### JavaScript

The following JavaScript example will replace the contents of the unordered list with the id 'names' with this list of business application names.

client.getBusinessApplicationNames(function(names) {

var ul = $('#names');

ul.empty();

$(names).each(function() {

ul.append('<li>' + $(this) + '</li>');

});

});

## Get Business Applications List

The Business Applications List web service resource returns a list of the business applications deployed to the CPF server. This can be used by applications to discover the applications available on a CPF server.

### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps |
| Method | GET |
| Content Types | json, html, xml, uri-list |
| Response | Resource List |

**Additional Fields**

|  |  |
| --- | --- |
| Name | Description |
| businessApplicationTitle | The display name of the business application & version. |
| businessApplicationName | The name of the business application to use in web service requests. |
| businessApplicationVersion | The most recent version of the business application. |

### JavaScript

The JavaScript client provides the getBusinessApplicationNames method to retrieve the list of business application names and call the callback function with an array of the names.

The following JavaScript example will replace the contents of the unordered list with the id 'names' with the list of business application names.

client.getBusinessApplicationNames(function(names) {

var ul = $('#names');

ul.empty();

$(names).each(function() {

ul.append('<li>' + $(this) + '</li>');

});

});

## Get Business Application Versions List

If an application needs to discover the list of versions supported by a business application on a CPF server the following command can be used to get the list of versions for a business application.

### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/{businessApplicationName} |
| Method | GET |
| Content Types | json, html, xml, uri-list |
| Response | Resource List |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| businessApplicationName | The name of the business application |

**Additional Fields**

|  |  |
| --- | --- |
| Name | Description |
| businessApplicationTitle | The display name of the business application & version. |
| businessApplicationName | The name of the business application to use in web service requests. |
| businessApplicationVersion | The most recent version of the business application. |

### JavaScript

The following JavaScript example will replace the contents of the unordered list with the id versions with the list of business application versions.

client.getBusinessApplicationVersions(function(versions) {

var ul = $('#versions');

ul.empty();

$(versions).each(function() {

ul.append('<li>' + $(this) + '</li>');

});

});

## Get Business Application Version Resources List

This resource returns a list of the resources available for a business application. This can be used to discover the operations possible on the business application as not all users or applications have all the operations available. The following operations are supported and described later in this document.

|  |  |
| --- | --- |
| Name | Description |
| instant | Form/resource to submit a single instant request to the business application. |
| single | Form/resource to submit a batch job containing a single request to the business application. |
| multiple | Form/resource to submit a batch job containing multiple requests to the business application. |

### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/{businessApplicationName}/{businessApplicationVersion} |
| Method | GET |
| Content Types | json, html, xml, uri-list |
| Response | Resource List |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| businessApplicationName | The name of the business application. |
| businessApplicationVersion | The version of the business application. |

**Additional Fields**

|  |  |
| --- | --- |
| Name | Description |
| businessApplicationTitle | The display name of the business application & version. |
| businessApplicationName | The name of the business application to use in web service requests. |
| businessApplicationVersion | The most recent version of the business application. |

#### JavaScript

Not currently supported in the JavaScript API.

## Creating Batch Jobs

The main purpose of the CPF is to allow users to submit a job of requests for execution in the cloud or to perform instant requests if a business application supports them. When the job has been completed the user can then download the results of the requests in a job. The methods described in this section describe how to submit jobs to the CPF using the CPF web services and clients.

### HTML Forms

The CPF provides HTML forms for end-users to create batch jobs for a business application. This can be used instead of creating a custom client application to allow users to submit jobs. Developers can use the forms as an example for creating a custom form. Each of these forms uses the web services to create the batch jobs.

#### Get Business Application Batch Job Single Request Form

This HTML form allows end-users to create a batch job containing a single request to a business application.

|  |  |
| --- | --- |
| Path | /ws/apps/{businessApplicationName}/{businessApplicationVersion}/single |
| Method | GET |
| Content Types | html |
| Response | Custom Form |

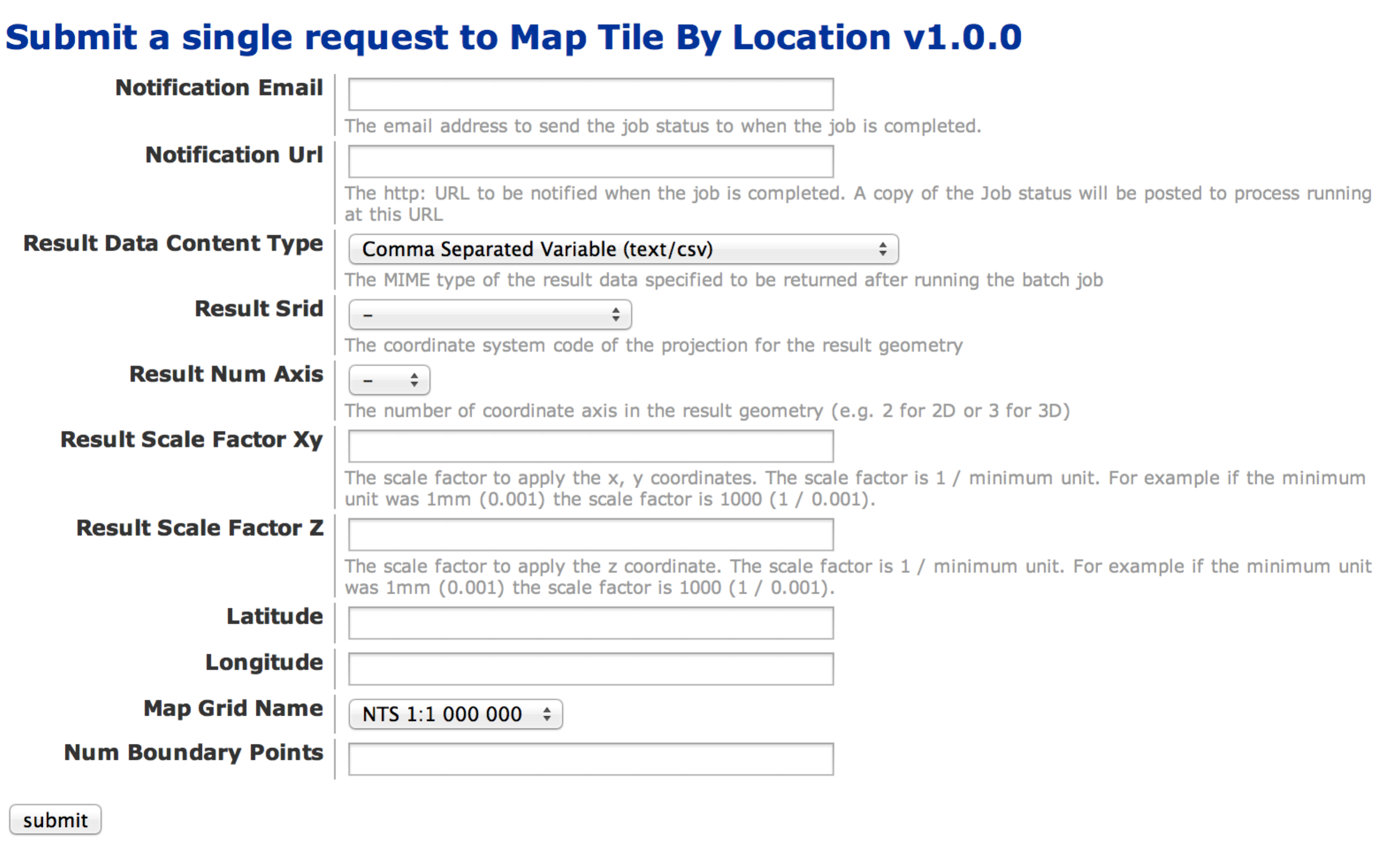
**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| businessApplicationName | The name of the business application. |
| businessApplicationVersion | The version of the business application. |

The form includes the following fields.

* Notification fields
* Content type for the result data for the job
* For geometry results the result srid, number of axis (2 for x, y, or 3 for x, y, z), XY scale factor and Z scale factor.
* One field for each job parameter
* One field for each request parameter for structured input data business applications.
* Input data content type and URL or file upload.

The following shows an example of the form for a structured input data business application with a geometry result.



#### Get Business Application Batch Job Multiple Requests Form

This HTML form allows end-users to create a batch job containing multiple requests to a business application.

**NOTE:** Currently it is not possible to create a multiple request batch job for an opaque input data business application.

|  |  |
| --- | --- |
| Path | /ws/apps/{businessApplicationName}/{businessApplicationVersion}/multiple |
| Method | GET |
| Content Types | html |
| Response | Custom Form |

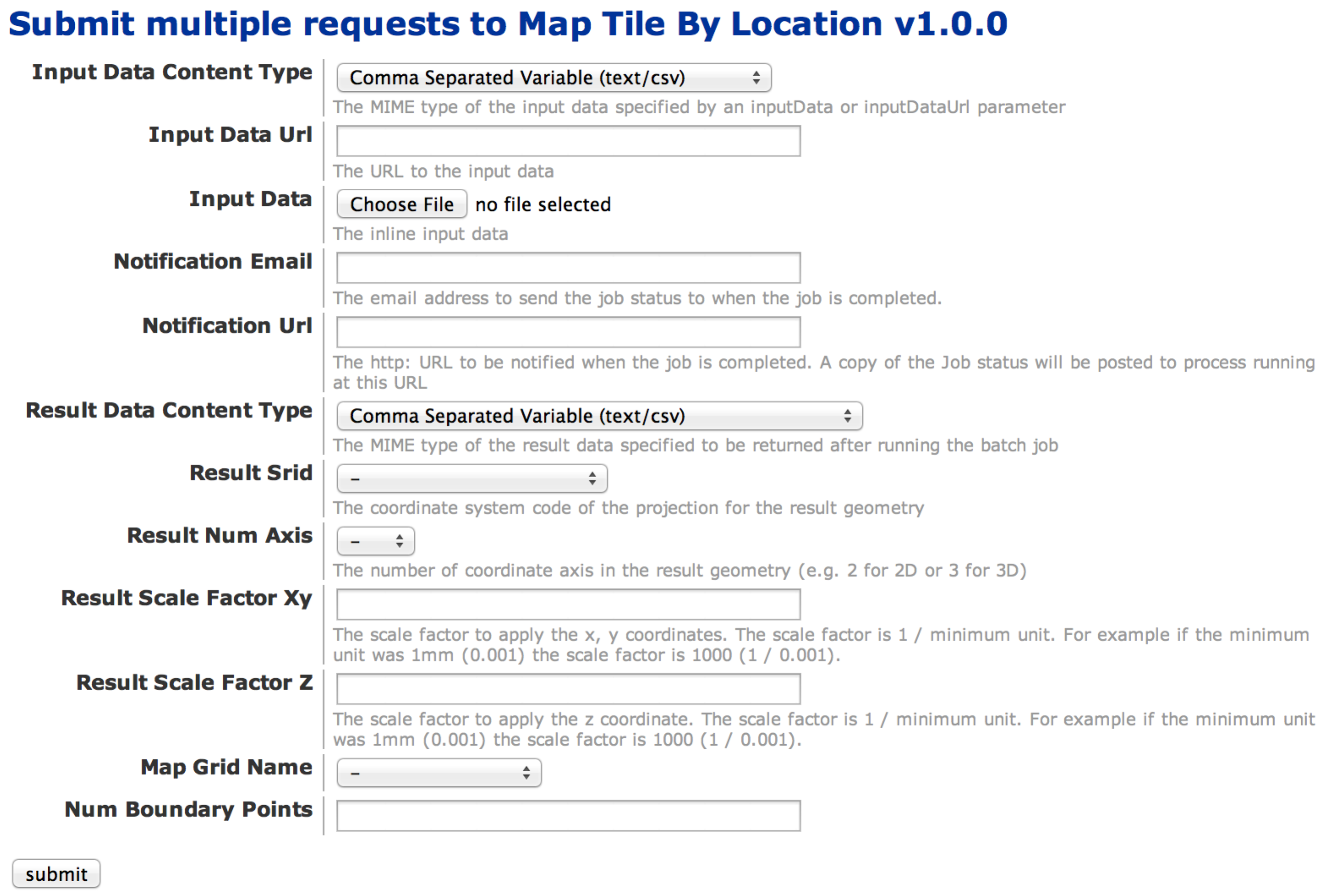
**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| businessApplicationName | The name of the business application. |
| businessApplicationVersion | The version of the business application. |

The form includes the following fields.

* Input data content type and URL or file upload
* Notification fields
* Content type for the result data for the job
* For geometry results the result srid, number of axis (2 for x, y, or 3 for x, y, z), XY scale factor and Z scale factor.
* One field for each job parameter

The following shows an example of the form for a structured input data business application with a geometry result.



#### Get Business Application Instant Single Request Form

This HTML form allows end-users to submit a single instant request to a business application.

|  |  |
| --- | --- |
| Path | /ws/apps/{businessApplicationName}/{businessApplicationVersion}/instant |
| Method | GET |
| Content Types | html |
| Response | Custom Form |

**Parameters**

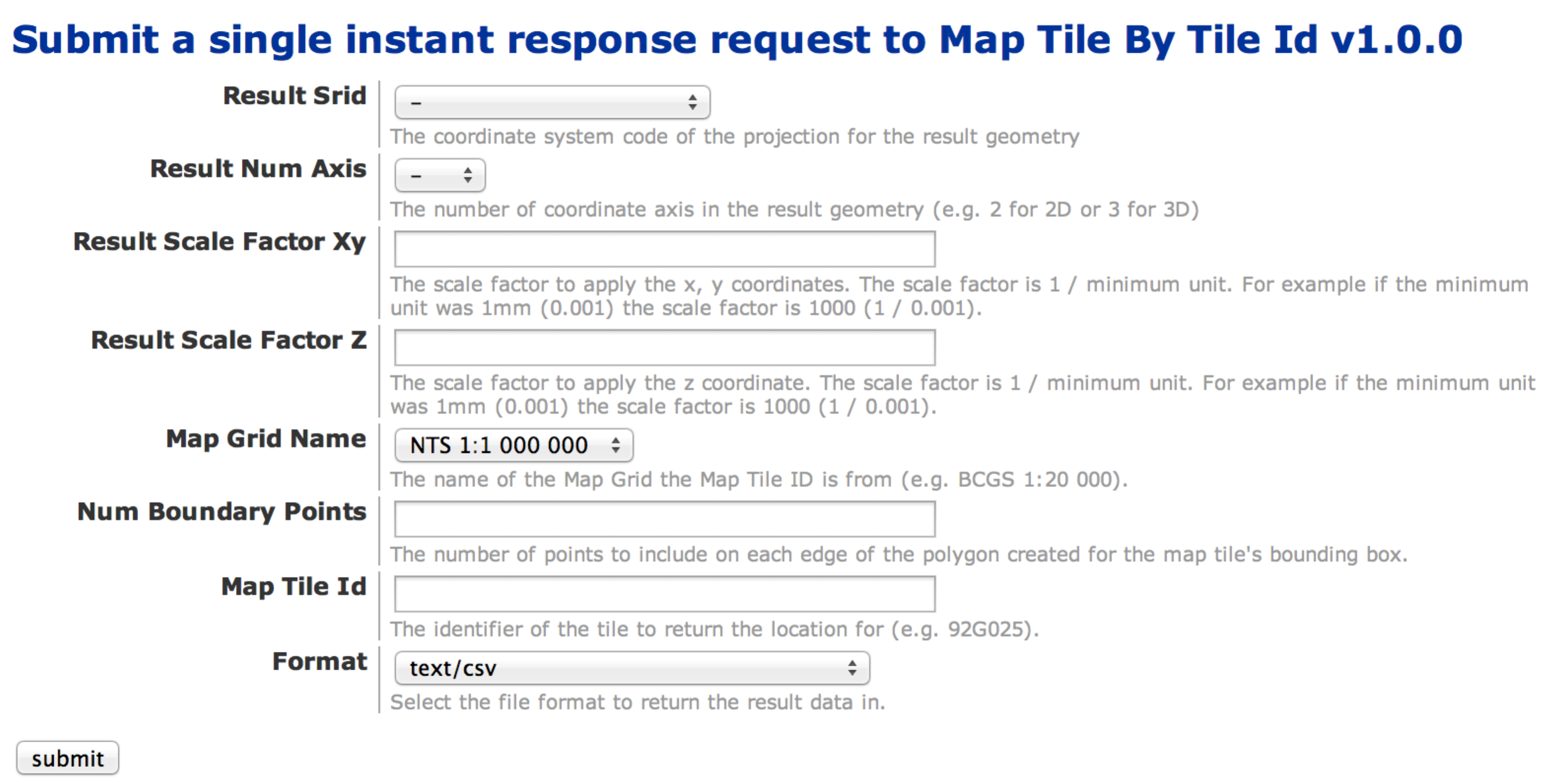
|  |  |
| --- | --- |
| Name | Description |
| businessApplicationName | The name of the business application. |
| businessApplicationVersion | The version of the business application. |

**NOTE:** Currently it is not possible to submit a single instant request for an opaque input data business application.

The form includes the following fields.

* For geometry results the result srid, number of axis (2 for x, y, or 3 for x, y, z), XY scale factor and Z scale factor.
* One field for each job parameter
* One field for each request parameter for structured input data business applications.
* Content type for the result data

The following shows an example of the form for a structured input data business application with a geometry result.



### Web Services

The following APIs are used to create new batch jobs or perform an instant request on a business application. They are HTTP POST API's and use the multipart/form-data encoding for passing parameters as it supports file uploads and larger file sizes. There is a limit to the size of a HTTP request using multipart/form-data. If a large file needs to be sent it is more efficient to create the file on a web server and use the option to pass a URL to the CPF APIs.

#### JavaScript Notes

The CPF JavaScript client requires that the developer create a HTML form that is populated with all the fields for a single or multiple CPF job for a single business application. The form can be a combination of hidden input fields, simple HTML fields or complex JavaScript fields. As there are a wide variety of requirements for forms the CPF JavaScript API does not include an API for constructing a form.

**NOTE:** Due to the cross-domain browser security mechanisms it is not possible to POST using JavaScript and get the contents of the document returned from the POST. This means it is not possible to get the Batch Job ID URL of the generated Job from JavaScript. The submit single and multiple web service API's return a HTML page that shows the Batch Job ID URL and the current status of the job. Future releases of the API will implement a mechanism to pass in a URL to a client application page that the user should be re-directed to when the job has been submitted.

A work around to this constraint would be to call the getJobIdUrls method before and after submitting a job. The before and after list can be compared to find the Batch Job ID URL for each new job added. This is not 100% reliable as the user could submit requests at the same time from different web browsers.

#### Post Business Application Batch Job Single Requests

Create a job for a structured or opaque input data business application containing a single request. The response is a HTTP 403 redirect to the Batch Job Info resource containing the ID and current status of the job. Clients can either save the redirect URL to follow later or follow the URL to get the full status and then read the URL from the status.

##### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/{businessApplicationName}/{businessApplicationVersion}/single |
| Method | POST |
| Request Body | multipart/form-data |
| Content Types | html, xml, json |
| Response | 403 Redirect to Get User Batch Job Info, the content type is passed on to the redirect. |

The parameters include separate parameters for each business application specific job parameter.

For structured input data the parameters include separate parameters for each business application specific request parameter.

For opaque input data the parameters include the inputDataContentType with the MIME-Type of the input data. The input data can either be specified as the inputData multipart/form-data file field or as a inputDataUrl that the CPF will download the file from. If the file is large use the inputDataUrl to point to the HTTP file server or HTTP resource in your application that the data can be downloaded from.

**Parameters**

| Name | Description |
| --- | --- |
| businessApplicationName | The name of the business application. |
| businessApplicationVersion | The version of the business application. |
| notificationEmail | The email address to send the job status to when the job is completed. |
| notificationUrl | The http: URL to be notified when the job is completed. A copy of the Job status will be posted to process running at this URL. |
| resultDataContentType | The MIME type of the result data specified to be returned after running the batch job. |
| resultSrid | The coordinate system code of the projection for the result geometry. |
| resultNumAxis | The number of coordinate axis in the result geometry (e.g. 2 for 2D or 3 for 3D). |
| resultScaleFactorXy | The scale factor to apply the x, y coordinates. The scale factor is 1 / minimum unit. For example if the minimum unit was 1mm (0.001) the scale factor is 1000 (1 / 0.001). |
| resultScaleFactorZ | The scale factor to apply the z coordinate. The scale factor is 1 / minimum unit. For example if the minimum unit was 1mm (0.001) the scale factor is 1000 (1 / 0.001). |
| *<jobFieldName>* | One parameter for each job parameter defined for the business application. |
| *<requestFieldName>* | One parameter for each request parameter defined for the structured input data business applications. |
| inputDataContentType | The content type of the input data. |
| inputData | The multi-part form data file containing the input data for a single opaque inout data request. **Either inputData or inputDataUrl can be specified, but not both.** |
| inputDataUrl | The URL to a http server containing the input data for a single opaque inout data request. **Either inputData or inputDataUrl can be specified, but not both.** |

##### JavaScript

The following example shows a simple form (singleForm) that submits a batch job with a single request to version 1.0.0 of the MapTileById business application. The form uses a hidden field for the mapGridName business application parameter and a text entry field for the mapTileId parameter.

JQuery is used to assign a function to be called when the submit button is clicked.

:

<head>

:

<script type="text/javascript">

$(document).ready(function() {

$('#**submit'**).click(function() {

client.submitSingle($('**#singleForm**'), 'MapTileById', '1.0.0');

});

});

</script>

</head>

<body>

<form id="multipleForm" method="post" enctype="**mulipart/form-data**">

<input type="hidden" name="**mapGridName**"

value="**BCGS 1:20 000**" />

<b>BCGS Map Tile Id: </b>

<input name="**mapTileId**"/> (e.g. 92g025)

<input type="button" id="**submit**" value="Submit" />

</form>

##### Java

The Java client provides convenience methods for submitting jobs that have structured or opaque input data. All of the methods return the Batch Job ID URL that can be used to download the results.

###### Create a Job with a Single Structured Request

The client provides the createJobWithStructuredSingleRequest method to create a job with a single request to a business application that supports structured input data.

The method accepts the business application name, version, request parameter map and result content type as arguments. If the job was created correctly it returns the job ID URL that can be used to check the status of the job and download the results. The request parameter map contains the name-value pairs for job or request parameters for the request.

The result content type for structured result data can be any of the supported values. However application/json is recommended if the results are going to be converted to a Java Map using the CPF Client.

String businessApplicationName = "**MapTileByTileId**";

String businessApplicationVersion = "**1.0.0**";

Map<String, Object> request = new HashMap<String, Object>();

request.put("**mapGridName**", "**BCGS 1:20 000**");

request.put("**mapTileId**", "**92j016**");

String resultContentType = "**application/json**";

String jobIdUrl = client.createJobWithStructuredSingleRequest(

businessApplicationName,

businessApplicationVersion,

request,

resultContentType

);

###### Create a Job with a Single Opaque URL Request

The client provides the createJobWithOpaqueUrlSingleRequest method to create a job with a single request to a business application that supports opaque input data. The input data must be accessible on a web or FTP server that can be specified as a URL. The URL must be on a network that is accessible to the CPF server (e.g. public internet).

The method accepts the business application name, version, job parameter map, URL to the input data, input data content type and result content type as arguments. If the job was created correctly it returns the job ID URL that can be used to check the status of the job and download the results. The job parameter map contains the global job name-value pair parameters that are applicable to all requests.

The result content type for structured result data can be any of the supported values. However application/json is recommended if the results are going to be converted to a Java Map using the CPF Client.

String businessApplicationName = "**digest**";

String businessApplicationVersion = "**1.0.0**";

Map<String, Object> jobParameters = new HashMap<String, Object>();

jobParameters.put("**algorithmName**", "**SHA**");

String inputDataUrl = "**http://www2.gov.bc.ca/common/images/bc\_logo.gif**"

String inputDataContentType = "**image/gif**"

String resultContentType = "**application/json**";

String jobIdUrl = client.createJobWithOpaqueUrlSingleRequest (

businessApplicationName,

businessApplicationVersion,

jobParameters,

inputDataUrl,

inputDataContentType,

resultContentType

);

###### Create a Job with a Single Opaque Resource Request

The client provides the createJobWithOpaqueResourceSingleRequest method to create a job with a single request to a business application that supports opaque input data. The input data is specified using a spring framework Resource[[1]](#footnote-1) object. The content of the location specified by the resource will be read by the client and sent as binary data to the server.

The method accepts the business application name, version, job parameter map, resource containing the input data, input data content type and result content type as arguments. If the job was created correctly it returns the job ID URL that can be used to check the status of the job and download the results. The job parameter map contains the global job name-value pair parameters that are applicable to all requests.

The result content type for structured result data can be any of the supported values. However application/json is recommended if the results are going to be converted to a Java Map using the CPF Client.

String businessApplicationName = "**digest**";

String businessApplicationVersion = "**1.0.0**";

Map<String, Object> jobParameters = new HashMap<String, Object>();

jobParameters.put("**algorithmName**", "**SHA**");

Resource inputDataResource = **new FileSystemResource("file.txt");**

String inputDataContentType = "**text/plain**"

String resultContentType = "**application/json**";

String jobIdUrl = client.createJobWithOpaqueUrlSingleRequest (

businessApplicationName,

businessApplicationVersion,

jobParameters,

inputDataResource,

inputDataContentType,

resultContentType

);

#### Post Business Application Batch Job Multiple Requests

Create a job for a structured or opaque input data business application containing multiple requests. The response is a HTTP 403 redirect to the Batch Job Info resource containing the ID and current status of the job. Clients can either save the redirect URL to follow later or follow the URL to get the full status and then read the URL from the status.

##### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/{businessApplicationName}/{businessApplicationVersion}/multiple |
| Method | POST |
| Request Body | multipart/form-data |
| Content Types | html, xml, json |
| Response | 403 Redirect to Get User Batch Job Info, the content type is passed on to the redirect. |

The parameters include separate parameters for each business application specific job parameter.

The inputDataContentType parameter specifies the MIME-Type of the input data, it must be before the input data in the form parameters.

The input data can either be specified as the inputData multipart/form-data file field or as an inputDataUrl that the CPF will download the file from. If the file is large use the inputDataUrl to point to the HTTP file server or HTTP resource in your application that the data can be downloaded from.

For structured input data business applications the input data must contain a structured file. Each record in the file is one request to be executed by the business application. The record contains fields for each of the business application specific request parameters,

For opaque input data business applications each request to be executed is specified using either an inputData or inputDataUrl. The inputData or inputDataUrl parameter can be repeated for each request to be processed. If each request uses a different content type the inputDataContentType can be specified for each request, otherwise it can be specified once and used for all the requests in that batch job.

**Parameters**

| Name | Description |
| --- | --- |
| businessApplicationName | The name of the business application. |
| businessApplicationVersion | The version of the business application. |
| notificationEmail | The email address to send the job status to when the job is completed. |
| notificationUrl | The http: URL to be notified when the job is completed. A copy of the Job status will be posted to process running at this URL. |
| resultDataContentType | The MIME type of the result data specified to be returned after running the batch job. |
| resultSrid | The coordinate system code of the projection for the result geometry. |
| resultNumAxis | The number of coordinate axis in the result geometry (e.g. 2 for 2D or 3 for 3D). |
| resultScaleFactorXy | The scale factor to apply the x, y coordinates. The scale factor is 1 / minimum unit. For example if the minimum unit was 1mm (0.001) the scale factor is 1000 (1 / 0.001). |
| resultScaleFactorZ | The scale factor to apply the z coordinate. The scale factor is 1 / minimum unit. For example if the minimum unit was 1mm (0.001) the scale factor is 1000 (1 / 0.001). |
| *<jobFieldName>* | One parameter for each job parameter defined for the business application. |
| inputDataContentType | The content type of the input data, repeat for each inputData and inputDataUrl. |
| inputData | The multi-part form data file containing the input data for a single opaque inout data request. |
| inputDataUrl | The URL to a http server containing the input data for a single opaque inout data request. |

##### JavaScript

The following example shows a simple form (multipleForm) that submits a structured input data batch job with a multiple requests to version 1.0.0 of the MapTileById business application. The form uses a hidden field for the inputDataContentType and a file select field for the inputData.

JQuery is used to assign a function to be called when the submit button is clicked.

:

<head>

:

<script type="text/javascript">

$(document).ready(function() {

$('#**submit'**).click(function() {

client.submitMultiple($('**#multipleForm**'), 'MapTileById', '1.0.0');

});

});

</script>

</head>

<body>

<form id="multipleForm" method="post" enctype="**mulipart/form-data**">

<input type="hidden" name="**inputDataContentType**" value="**text/csv**" />

<b>Select CSV file</b>

<input type="file" name="**inputData**" />

<input type="button" id="**submit**" value="Submit" />

</form>

The following example shows a simple form (multipleForm) that submits a opaque input data batch job with a mutliple requests to version 1.0.0 of the ToJpeg business application (for example only, not a real application). The form uses a hidden field for the inputDataContentType and two file select field for inputData. for the two GIF files, one for each of the two requests in the batch job.

:

<head>

:

<script type="text/javascript">

$(document).ready(function() {

$('#**submit'**).click(function() {

client.submitMultiple($('**#multipleForm**'), 'ToJpeg', '1.0.0');

});

});

</script>

</head>

<body>

<form id="multipleForm" method="post" enctype="**mulipart/form-data**">

<input type="hidden" name="**inputDataContentType**" value="**image/gif**" />

<b>Select GIF file 1</b>

<input type="file" name="**inputData**" />

<b>Select GIF file 2</b>

<input type="file" name="**inputData**" />

<input type="button" id="**submit**" value="Submit" />

</form>

##### Java

The Java client provides convenience methods for submitting jobs that have structured or opaque input data. All of the methods return the Batch Job ID URL that can be used to download the results.

###### Create a Job with Multiple Structured Requests

The client provides the createJobWithStructuredMultipleRequests method to create a job with a multiple requests to a business application that supports structured input data.

The method accepts the business application name, version, job parameter map, list of request parameter maps and result content type as arguments. If the job was created correctly it returns the job ID URL that can be used to check the status of the job and download the results. The job parameter map contains the global job name-value pair parameters that are applicable to all requests. Each request parameter map contains the name-value parameters for each request.

The result content type for structured result data can be any of the supported values. However application/json is recommended if the results are going to be converted to a Java Map using the CPF Client.

String businessApplicationName = "**MapTileByTileId**";

String businessApplicationVersion = "**1.0.0**";

Map<String, Object> jobParameters = new HashMap<String, Object>();

jobParameters.put("**mapGridName**", "**BCGS 1:20 000**");

Map<String, Object> request1 = new HashMap<String, Object>();

request1.put("**mapTileId**", "**92j016**");

Map<String, Object> request2 = new HashMap<String, Object>();

request2.put("**mapTileId**", "**92j017**");

List<Map<String,Object>> requests = Arrays.asList(

request1,

request2

);

String resultContentType = "**application/json**";

String jobIdUrl = client.createJobWithStructuredSingleRequest(

businessApplicationName,

businessApplicationVersion,

jobParameters,

requests,

resultContentType

);

###### Create a Job with Multiple Structured Requests from a Resource

The client provides the createJobWithStructuredResourceMultipleRequests method to create a job with a multiple requests to a business application that supports structured input data. The data must be in an existing structured data file (or other resource).

The method accepts the business application name, version, job parameter map, number of requests in the input data, input data resource, input data content type, and result content type as arguments. If the job was created correctly it returns the job ID URL that can be used to check the status of the job and download the results.

The result content type for structured result data can be any of the supported values. However application/json is recommended if the results are going to be converted to a Java Map using the CPF Client.

String businessApplicationName = "**MapTileByTileId**";

String businessApplicationVersion = "**1.0.0**";

Map<String, Object> jobParameters = new HashMap<String, Object>();

jobParameters.put("**mapGridName**", "**BCGS 1:20 000**");

int numRequests = **1**;

Resource inputData = new FileSystemResource**("/tmp/test.csv**");

String inputDataContentType = "**text/csv**";

String resultContentType = "**application/json**";

String jobIdUrl = client.createJobWithStructuredSingleRequest(

businessApplicationName,

businessApplicationVersion,

jobParameters,

numRequests,

inputData,

inputDataContentType,

resultContentType

);

###### Create a Job with Multiple Structured Requests from a URL

The client provides the createJobWithStructuredResourceMultipleRequests method to create a job with a multiple requests to a business application that supports structured input data. The data must be in an existing structured data file available via a publically accessible URL.

The method accepts the business application name, version, job parameter map, number of requests in the input data, input data resource, input data content type, and result content type as arguments. If the job was created correctly it returns the job ID URL that can be used to check the status of the job and download the results.

The result content type for structured result data can be any of the supported values. However application/json is recommended if the results are going to be converted to a Java Map using the CPF Client.

String businessApplicationName = "**MapTileByTileId**";

String businessApplicationVersion = "**1.0.0**";

Map<String, Object> jobParameters = new HashMap<String, Object>();

jobParameters.put("**mapGridName**", "**BCGS 1:20 000**");

int numRequests = **1**;

String inputDataUrl = "**http://mysrver.com/test.csv**");

String inputDataContentType = "**text/csv**";

String resultContentType = "**application/json**";

String jobIdUrl = client.createJobWithStructuredSingleRequest(

businessApplicationName,

businessApplicationVersion,

jobParameters,

numRequests,

inputDataUrl,

inputDataContentType,

resultContentType

);

###### Create a Job with a Multiple Opaque URL Requests

The client provides the createJobWithOpaqueUrlMultipleRequests method to create a job with multiple requests to a business application that supports opaque input data. The input data must be accessible on a web or FTP server that can be specified as a URL. The URL must be on a network that is accessible to the CPF server (e.g. public internet).

The method accepts the business application name, version, job parameter map, list of URLs to the input data, input data content type and result content type as arguments. If the job was created correctly it returns the job ID URL that can be used to check the status of the job and download the results. The job parameter map contains the global job name-value pair parameters that are applicable to all requests.

The result content type for structured result data can be any of the supported values. However application/json is recommended if the results are going to be converted to a Java Map using the CPF Client.

String businessApplicationName = "**digest**";

String businessApplicationVersion = "**1.0.0**";

Map<String, Object> jobParameters = new HashMap<String, Object>();

jobParameters.put("**algorithmName**", "**SHA**");

List<String> inputDataUrls = Arrays.asList(

"**http://maps.google.ca/images/experiments/nav\_logo78.png**",

"**http://www.blogger.com/img/logo40.png**"

);

String inputDataContentType = "**image/png**"

String resultContentType = "**application/json**";

String jobIdUrl = client.createJobWithOpaqueUrlMultipleRequests(

businessApplicationName,

businessApplicationVersion,

jobParameters,

inputDataUrls,

inputDataContentType,

resultContentType

);

###### Create a Job with a Multiple Opaque Resource Requests

The client provides the createJobWithOpaqueResourceMultipleRequests method to create a job with multiple requests to a business application that supports opaque input data. The input data is specified using list of spring framework Resource[[2]](#footnote-2) objects. The content of the location specified by the resource will be read by the client and sent as binary data to the server.

The method accepts the business application name, version, job parameter map, resource containing the input data, input data content type and result content type as arguments. If the job was created correctly it returns the job ID URL that can be used to check the status of the job and download the results. The job parameter map contains the global job name-value pair parameters that are applicable to all requests.

The result content type for structured result data can be any of the supported values. However application/json is recommended if the results are going to be converted to a Java Map using the CPF Client.

String businessApplicationName = "**digest**";

String businessApplicationVersion = "**1.0.0**";

Map<String, Object> jobParameters = new HashMap<String, Object>();

jobParameters.put("**algorithmName**", "**SHA**");

List<Resource> inputDataResources = Arrays.asList(

**new FileSystemResource("file1.txt"),**

**new FileSystemResource("file2.txt")**

);

String inputDataContentType = "**text/plain**"

String resultContentType = "**application/json**";

String jobIdUrl = client.createJobWithOpaqueResourceMultipleRequests(

businessApplicationName,

businessApplicationVersion,

jobParameters,

inputDataResources,

inputDataContentType,

resultContentType

);

#### Post Business Application Instant Single Request

Execute a single instant request to a structured or opaque input data business application. The response is either a Object Detail for structured result data or the file for a opaque result data business application

##### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/{businessApplicationName}/{businessApplicationVersion}/instant |
| Method | POST |
| Request Body | multipart/form-data |
| Content Types | html, xml, json |
| Response | Object Detail or download file |

The parameters include separate parameters for each business application specific job parameter.

For structured input data the parameters include separate parameters for each business application specific request parameter.

For opaque input data the parameters include the inputDataContentType with the MIME-Type of the input data. The input data can either be specified as the inputData multipart/form-data file field or as a inputDataUrl that the CPF will download the file from. If the file is large use the inputDataUrl to point to the HTTP file server or HTTP resource in your application that the data can be downloaded from.

**Parameters**

| Name | Description |
| --- | --- |
| businessApplicationName | The name of the business application. |
| businessApplicationVersion | The version of the business application. |
| notificationEmail | The email address to send the job status to when the job is completed. |
| notificationUrl | The http: URL to be notified when the job is completed. A copy of the Job status will be posted to process running at this URL. |
| resultDataContentType | The MIME type of the result data specified to be returned after running the batch job. |
| resultSrid | The coordinate system code of the projection for the result geometry. |
| resultNumAxis | The number of coordinate axis in the result geometry (e.g. 2 for 2D or 3 for 3D). |
| resultScaleFactorXy | The scale factor to apply the x, y coordinates. The scale factor is 1 / minimum unit. For example if the minimum unit was 1mm (0.001) the scale factor is 1000 (1 / 0.001). |
| resultScaleFactorZ | The scale factor to apply the z coordinate. The scale factor is 1 / minimum unit. For example if the minimum unit was 1mm (0.001) the scale factor is 1000 (1 / 0.001). |
| *<jobFieldName>* | One parameter for each job parameter defined for the business application. |
| *<requestFieldName>* | One parameter for each request parameter defined for the structured input data business applications. |
| inputDataContentType | The content type of the input data. |
| inputData | The multi-part form data file containing the input data for a single opaque inout data request. **Either inputData or inputDataUrl can be specified, but not both.** |
| inputDataUrl | The URL to a http server containing the input data for a single opaque inout data request. **Either inputData or inputDataUrl can be specified, but not both.** |

#### JavaScript

Not currently supported in the JavaScript API.

#### Java

Not currently supported in the Java API.

## Get Users Resources

This resource returns a list of the resources available for a user. This can be used to discover the operations possible on the business application as not all users or applications have all the operations available. The following operations are supported and described later in this document.

|  |  |
| --- | --- |
| Name | Description |
| jobs | Resource showing the list of the user's jobs. |
| apps | Resource showing the list of business applications, from there a list of the user's jobs for that application can be obtained. |
| accountInformation | Resource showing the user's account information. |

### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/users/{consumerKey} |
| Method | GET |
| Content Types | json, html, xml, uri-list |
| Response | Resource List |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| consumerKey | The user's consumer key. |

#### JavaScript

Not currently supported in the JavaScript API.

#### Java

Not currently supported in the Java API.

## Get User Account Information

Get the user's consumer key and consumer secret as a HTML page for the user's reference. The user can use the reset credentials button to get a new consumer key generated.

### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/users/{consumerKey}/account |
| Method | GET |
| Content Types | html |
| Response | Object View |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| consumerKey | The user's consumer key. |

#### JavaScript

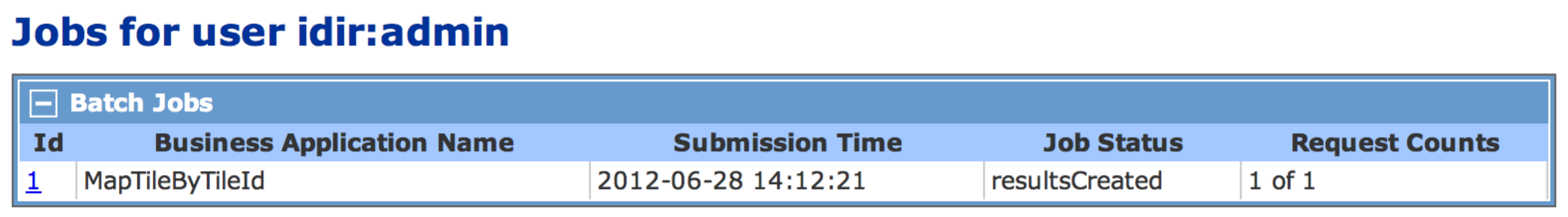
Not currently supported in the JavaScript API.

#### Java

Not currently supported in the Java API.

## Get User Batch Jobs List

Get the list of batch jobs for a user. The HTML page includes a custom table showing the status of the batch jobs.



### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/users/{consumerKey}/jobs |
| Method | GET |
| Content Types | json, html, xml, uri-list |
| Response | Resource List |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| consumerKey | The user's consumer key. |

**Additional Fields**

|  |  |
| --- | --- |
| Name | Description |
| batchJobId | The identifier of the batch job. |
| jobStatus | The current status of the batch job. |
| userId | The consumer key of the user. |
| creationTimestamp | The creation timestamp. |

### JavaScript

The JavaScript client returns the URLs to the Batch Job Info page for each job. The job status can be downloaded from there.

The following JavaScript example will replace the contents of the unordered list with the id jobUrls with the list of links to the batch job status pages.

client.getJobIdUrls(function(jobUrls) {

var ul = $('#jobUrls');

ul.empty();

$( jobUrls).each(function() {

ul.append('<li><a href="' + $(this) + '">' + $(this) + '</a></li>');

});

});

### Java

The Java client returns the URLs to the Batch Job Info page for each job. The job status can be downloaded from there.

List<String> jobUrls = client.getUserJobIdUrls();

## Get User Business Applications List

The User Business Applications List web service resource returns a list of the business applications deployed to the CPF server. This can be used to navigate to the user's jobs for each application.

### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/users/{consumerKey}/apps |
| Method | GET |
| Content Types | json, html, xml, uri-list |
| Response | Resource List |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| consumerKey | The user's consumer key. |

#### JavaScript

Not currently supported in the JavaScript API.

#### Java

Not currently supported in the Java API.

## Get User Business Application Resources List

The User Business Applications Resources List web service resource returns a list of the resources available for a user for the business applications deployed to the CPF server. This can be used to navigate to the user's jobs for each application.

|  |  |
| --- | --- |
| Name | Description |
| jobs | Resource showing the list of the user's jobs for the business application. |

### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/users/{consumerKey}/apps/{businessApplicationName} |
| Method | GET |
| Content Types | json, html, xml, uri-list |
| Response | Resource List |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| consumerKey | The user's consumer key. |
| businessApplicationName | The name of the business application. |

#### JavaScript

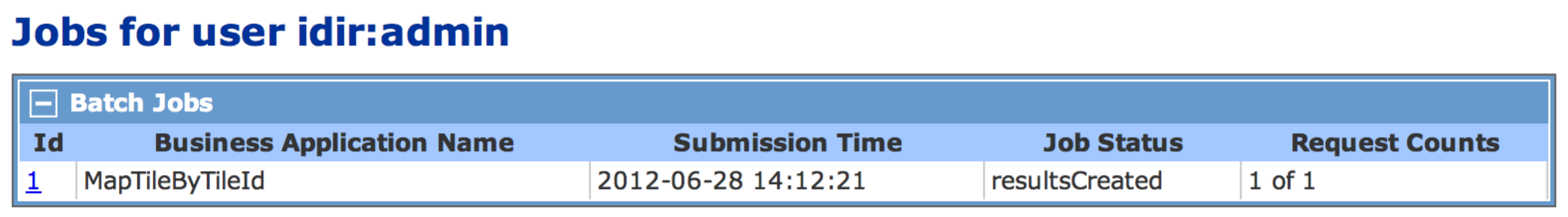
Not currently supported in the JavaScript API.

#### Java

Not currently supported in the Java API.

## Get User Business Application Batch Job List

Get the list of the user's batch jobs for a business application. The HTML page includes a custom table showing the status of the batch jobs.



### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/users/{consumerKey}/apps/{businessApplicationName}/jobs |
| Method | GET |
| Content Types | json, html, xml, uri-list |
| Response | Resource List |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| consumerKey | The user's consumer key. |
| businessApplicationName | The name of the business application. |

**Additional Fields**

|  |  |
| --- | --- |
| Name | Description |
| batchJobId | The identifier of the batch job. |
| jobStatus | The current status of the batch job. |
| userId | The consumer key of the user. |
| creationTimestamp | The creation timestamp. |

### JavaScript

The JavaScript client returns the URLs to the Batch Job Info page for each job. The job status can be downloaded from there.

The following JavaScript example will replace the contents of the unordered list with the id jobUrls with the list of links to the batch job status pages.

client.getJobIdUrls(function(jobUrls) {

var ul = $('#jobUrls');

ul.empty();

$( jobUrls).each(function() {

ul.append('<li><a href="' + $(this) + '">' + $(this) + '</a></li>');

});

});

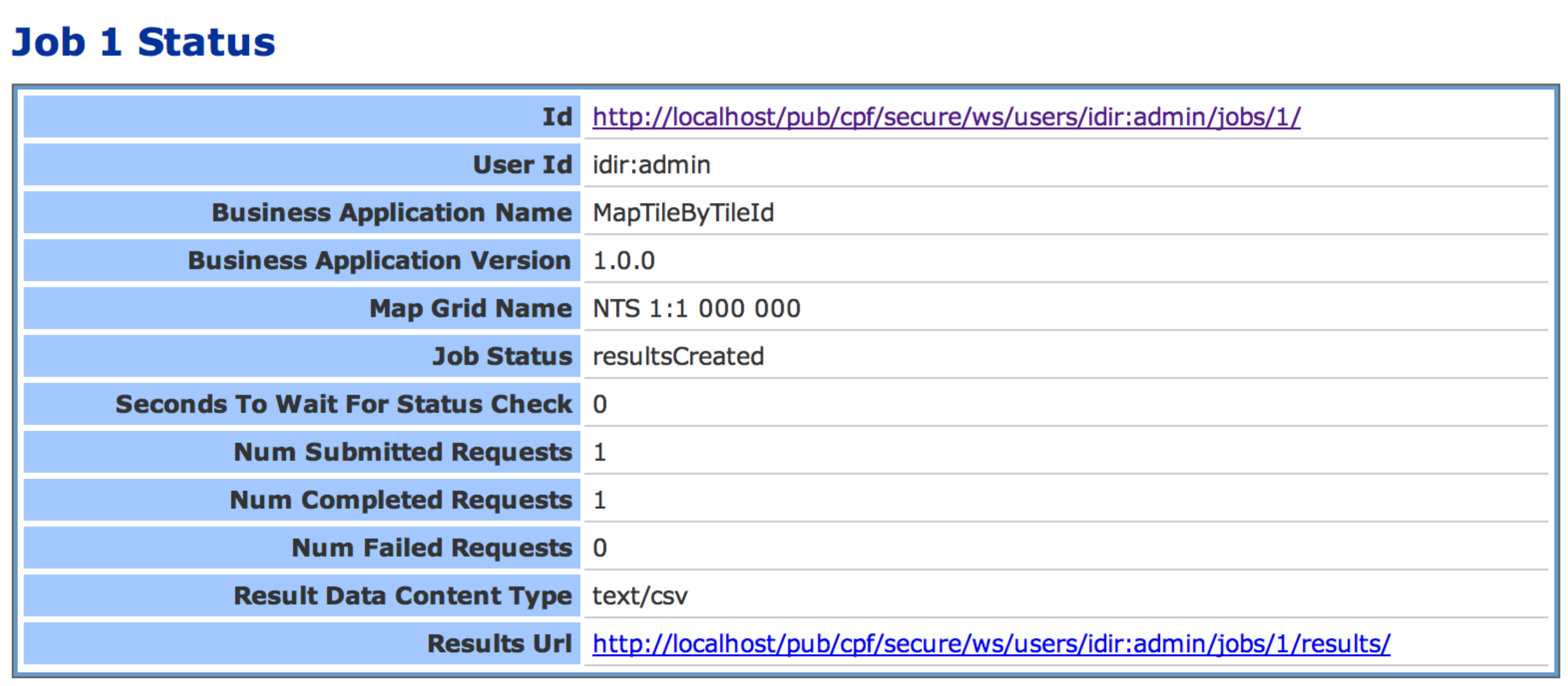
### Java

The Java client returns the URLs to the Batch Job Info page for each job. The job status can be downloaded from there.

List<String> jobUrls = client.getUserJobIdUrls();

## Get User Batch Job Info

Get current status of a batch job for a user. The HTML page includes a custom view showing the status of a batch job.



### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/users/{consumerKey}/jobs/{jobId} |
| Method | GET |
| Content Types | json, html, xml, uri-list |
| Response | Object Detail |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| consumerKey | The user's consumer key. |
| jobId | The identifier of the batch job. |

**Fields**

| Name | Description |
| --- | --- |
| id | The identifier of the batch job. |
| userId | The consumer key of the user. |
| businessApplicationName | The name of the business application. |
| businessApplicationVersion | The version of the business application. |
| *<jobFieldName>* | One field for each job parameter defined for the business application. |
| jobStatus | The current status of the batch job. |
| secondsToWaitForStatusCheck | The number of seconds to wait before checking the status again. |
| numSubmittedRequests | The number of requests submitted. |
| numCompletedRequests | The number of requests successfully completed. |
| numFailedRequests | The number of failed requests. |
| resultDataContentType | The result data content type. |
| resultsUrl | The URL to the Batch Job Results List resource if the job is completed. |

### JavaScript

The following JavaScript example will display an alert with the id of a job.

var jobStatusUrl = '**http://apps.gov.bc.ca/pub/cpf/ws/users/<userId>/jobs/1**';

client.getJobStatus(jobStatusUrl , function(jobStatus) {

alert(jobStatus['id']);

});

### Java

The Java client returns a Map containing each of the field values.

String jobStatusUrl = "**http://apps.gov.bc.ca/pub/cpf/ws/users/<userId>/jobs/1**";

Map<String client.getJobStatus(

jobStatusUrl

);

## Get User Batch Job Results List

This resource returns the list of result files for a batch job.

For structured result data there will be one structuredResultData file (if there was at least one successful request) and one errorResultData file (if there was at least one failed request).

For opaque result data there will be one or more opaqueResultData files for each successful request and one errorResultData file (if there was at least one failed request).

### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/users/{consumerKey}/jobs/{jobId}/results |
| Method | GET |
| Content Types | json, html, xml, uri-list |
| Response | Resource List |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| consumerKey | The user's consumer key. |
| jobId | The identifier of the batch job. |

**Additional Fields**

| Name | Description |
| --- | --- |
| batchJobId | The identifier of the batch job. |
| batchJobResultId | The identifier of the batch job result file. |
| userId | The consumer key of the user. |
| businessApplicationName | The name of the business application. |
| batchJobResultType | The type of result file structuredResultData, opaqueResultData, errorResultData. |
| batcJobResultContentType | The MIME-Type of the data in the result file. |

### JavaScript

The JavaScript API returns a list of JSON objects (one per file) containing the result fields above.

The following JavaScript example will replace the contents of the unordered list with the id resultUrls with the list of links to the files to download.

var jobStatusUrl = '**http://apps.gov.bc.ca/pub/cpf/ws/users/<userId>/jobs/1**';

client.getJobResultFileList(jobStatusUrl, function(results) {

var ul = $('#resultUrls');

ul.empty();

$(results).each(function() {

var url = $(this)['resourceUri'];

ul.append('<li><a href="' + url + '">' + url + '</a></li>');

});

});

### Java

The Java client returns a List of Map for each result file containing each of the field values.

String jobStatusUrl = "**http://apps.gov.bc.ca/pub/cpf/ws/users/<userId>/jobs/1**";

List<Map<String, Object>> results = client.getJobResultFileList(

jobStatusUrl,

5000

);

for (Map<String, Object> resultFile : results) {

String resultFileUrl = resultFile.get("resourceUri");

}

## Get User Batch Job Result File

This resource returns the contents of a Batch Job Result File.

### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/users/{consumerKey}/jobs/{jobId}/results/{resultId} |
| Method | GET |
| Content Types | Will return the result data content in the content type of the file. |
| Response | Resource List |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| consumerKey | The user's consumer key. |
| jobId | The identifier of the batch job. |
| resultId | The identifier of the batch job result file. |

### Error Result File

If there were any errors processing any of the requests in a CPF job an error file will be created. The error file can be downloaded and processed using the getJobErrorResults method.

#### JavaScript

The following JavaScript example will replace the contents of the table with the id errors with the list of errors.

var jobStatusUrl = '**http://apps.gov.bc.ca/pub/cpf/ws/users/<userId>/jobs/1**';

client.getJobErrorResults(jobStatusUrl, function(errors) {

var tbody = $('#errors tbody');

tbody.empty();

$(results).each(function() {

var error = $(this);

tbody.append('<tr><td>'

+ error['sequenceNumber'] + '</td><td>'

+ error['errorCode'] + '</td><td>'

+ error['errorMessage'] + '</td></tr>');

});

});

#### Java

The method takes the jobIdUrl and a wait time as arguments. The wait time is the number of milliseconds to wait for the job to complete before returning the errors. If the job is not completed an IllegalStateException will be thrown.

Reader<Map<String, Object>> errors = client.getJobErrorResults(

jobIdUrl,

5000

);

for (Map<String, Object> error : errors) {

System.out.println(error.get("sequenceNumber"));

System.out.println(error.get("errorCode"));

System.out.println(error.get("errorMessage"));

}

### Structured Result File

If the business application returned structured result data the single structured result file can be downloaded and processed using the getJobStructuredResults method.

**NOTE:** These methods assume that the structured results are in JSON format. Any other formats cannot be processed by these methods.

#### JavaScript

The following JavaScript example will replace the contents of the table with the id results with the list of results.

var jobStatusUrl = '**http://apps.gov.bc.ca/pub/cpf/ws/users/<userId>/jobs/1**';

client.getJobErrorResults(jobStatusUrl, function(results) {

var tbody = $('#results tbody');

tbody.empty();

$(results).each(function() {

var result = $(this);

var row = '<tr>';

result.each(function(key, value)) {

row += '<td>' + value + '</td>';

});

row += '</tr>';

tbody.append(row);

});

});

#### Java

The method takes the jobIdUrl and a wait time as arguments. The wait time is the number of milliseconds to wait for the job to complete before returning the errors. If the job is not completed an IllegalStateException will be thrown.

The map for each result contains the sequence number to tie it back to the original request and the values for each of the structured result attributes. If the business application returns multiple records for a single request there will be a resultNumber attribute in the map. The resultNumber starts an 1 and increments up to the number of result records returned for the request.

If there was no structured result file an IllegalStateException will be thrown.

Reader<Map<String, Object>> results = client.getJobStructuredResults(

jobIdUrl,

5000

);

for (Map<String, Object> result : results) {

Number sequenceNuumber = result.get("sequenceNumber");

String **mapTileName** = result.get("**mapTileName** ");

}

### Process Result File

A result file can be downloaded and processed as a binary stream.

#### JavaScript

Not currently supported in the JavaScript API.

#### Java

A result file can be downloaded using the processResultFile method. To ensure that the connection to the server is closed correctly the method does not return the input stream to download the file from. Instead it uses a callback mechanism. The method is passed an ObjectProcessor instance that will be used to process the input stream. The method gets an input stream for the URL, then invokes the process method on the callback instance with the input stream. When the process method returns or throws an exception the input stream and connection is closed.

The following example shows how to print the contents of all the files for a job.

import com.revolsys.util.ObjectProcessor

:

List<Map<String, Object>> results = client.getJobResultFileList(

jobIdUrl,

5000

);

for (Map<String, Object> resultFile : results) {

String url = (String)resultFile.get("resourceUri");

client.**processResultFile**(url, **new ObjectProcessor<InputStream>() {**

**public void process(InputStream in) {**

DataInputStream din = new DataInputStream(in);

try {

for (String line = din.readLine(); line != null; line=din.readLine()) {

System.out.println(line);

}

} catch (IOException e) {

throw new RuntimeException("Unable to process results", e);

}

}

});

}

## Post User Batch Job Delete

When the results of a job have been processed and they are no longer needed the client should delete the job from server. If the client does not close the job, it will be automatically closed after a few days.

### Web Service API

|  |  |
| --- | --- |
| Path | /ws/apps/users/{consumerKey}/jobs/{jobId} |
| Method | DELETE |
| Content Types | N/A |
| Response | N/A |

**Parameters**

|  |  |
| --- | --- |
| Name | Description |
| consumerKey | The user's consumer key. |
| jobId | The identifier of the batch job. |

### JavaScript

Not currently supported in the JavaScript API.

### Java

The following example shows how to delete all jobs for the user.

List<String> jobIdUrls = client.getUserJobIdUrls();

for (String jobIdUrl : jobIdUrls) {

client.closeJob(jobIdUrl);

}

1. Requirements
   1. JavaScript Requirements
      1. JQuery

The CPF JavaScript client uses the JQuery JavaScript library. Developers should have an understanding of JQuery as the examples in this document use JQuery to display the results from the CPF client.

The following should be included in the head section of the HTML document.

<script

type="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"

></script>

* + 1. CPF JavaScript API

The CPF JavaScript API (cpf\_client.js) can be referenced from the CPF web services web application. The path is always /js/cpf\_client.js below the application root. It is recommended to reference the script from the CPF server so that the most recent version is accessed.

The following should be included in the head section of the HTML document. Replace **https://apps.gov.bc.ca/pub/cpf**/ with the location for the server you are connecting to.

<script

type="text/javascript"

src="**https://apps.gov.bc.ca/pub/cpf**/js/cpf\_client.js"

></script>

* 1. Java Requirements
     1. Java

The developer's machine and deployment server must have Java 1.6+ installed.

* + 1. Subversion

CPF use the Apache Subversion source code control system to track changes to the source code and tag versions for each release. It is recommended that clients use a source code control system such as subversion for their applications.

Subversion command line tools can be downloaded from the following site.

<http://subversion.apache.org/> - **Command line tools**

<http://www.eclipse.org/subversive/> - **Eclipse Team Provider**

<http://tortoisesvn.tigris.org/> - **Windows SVN GUI**

A guide on using Subversion is available on the following site.

<http://svnbook.red-bean.com/>

The CPF source code can be accessed using the following Subversion repository within the BC Government network or VPN. Plugin developers do not need the source code to develop a plugin but it is available for reference purposes.

<http://poplar.idir.bcgov/svn/cpf/api-source/trunk/> - **Subversion repository for download**

<https://apps.gov.bc.ca/int/wsvn/CITZ.cpf/api-source/> - **Subversion Web interface**

Developers will need to request via their Business Analyst that a subversion repository and user account be created for their project.

* + 1. Maven

The CPF is developed using the Apache Maven build tool. CPF client must be developed using Maven as it manages the dependencies between the CPF components and external libraries. Developers must have an understanding of developing applications using Maven.

Maven 3.0.x can be downloaded from the following site.

<http://maven.apache.org/> - **Command line tools**

<http://eclipse.org/m2e/> - **Eclipse Maven**

A guide on using Maven is available on the following site.

<http://www.sonatype.com/books/mvnref-book/reference/>

The CPF libraries are deployed to the CITZ BC Government Maven repository inside the BC Government network/VPN. Follow instructions Appendix A.1.1 Maven settings.xml to use this repository to download the CPF libraries. **NOTE: This step must be completed before continuing with these instructions.**

The following example Maven settings.xml adds two new profiles that include references to the Sonatype maven repository and the CITZ BC Government Maven Repository. Copy the following file to ~/.m2/settings.xml (or merge if the file already exists) to enable Maven to download the libraries from these locations.

**NOTE: To use the CITZ BC Government Maven Repository a VPN connection is required.**

<?xml version="1.0" encoding="UTF-8"?>

<settings

xmlns="http://maven.apache.org/SETTINGS/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

http://maven.apache.org/SETTINGS/1.0.0

http://maven.apache.org/xsd/settings-1.0.0.xsd

"

>

<profiles>

<profile>

<id>citz-artifactory</id>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<repositories>

<repository>

<snapshots>

<enabled>false</enabled>

</snapshots>

<id>apps.gov.bc.ca-releases</id>

<name>apps.gov.bc.ca-releases</name>

<url>http://apps.gov.bc.ca/gov/artifactory/repo1</url>

</repository>

<repository>

<snapshots>

<enabled>true</enabled>

</snapshots>

<id>apps.gov.bc.ca</id>

<name>apps.gov.bc.ca-snapshots</name>

<url>http://apps.gov.bc.ca/gov/artifactory/repo2</url>

</repository>

</repositories>

</profile>

</profiles>

</settings>

This step assumes that the developer has already created a Maven Java project for their application with a pom.xml file for the maven configuration.

Add the following to the dependencies section of the pom.xml file. This will include the CPF client API and all required dependencies.

<dependency>

<groupId>ca.bc.gov.open.cpf</groupId>

<artifactId>cpf-api-client</artifactId>

<version>**3.0.0-SNAPSHOT**</version>

</dependency>

1. http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/core/io/Resource.html [↑](#footnote-ref-1)
2. http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/core/io/Resource.html [↑](#footnote-ref-2)