How-to Guide for External Job Scheduling

SAP Integrated Business Planning 1702 and Higher

6-9-2020

Document Version 1.11 - Public

Document History

If you use a local PDF copy or a paper printout of this document, make sure that you have the latest version. You can find the latest version attached to **SAP Note 2503171**.

Version	Date	Change
0.1	March, 2017	Initial version
1.0	May, 2017	First published version
1.1	Sep 30, 2017	Formal changes (title page, document history, ...)
1.2	Dec 11, 2017	Added note on SSL
1.3	Jan, 2018	Added JobinfoGet
1.4	March, 2018	Small adjustments in curl example
1.5	May, 2018	Certificate
1.6	July, 2018	Added SAP Solution Manager integration
1.7	Dec, 2018	Removed limitation on starting job chains with External Scheduler (Background section) Added JobListGet Added POSTMAN Sample
1.8	Jan, 2019	Enhanced the documentation for the ABAP code sample
1.9	March, 2019	Enhanced the ABAP section with a Sample using Certificate based Authentication for the Inbound Communication User
1.10	August, 2019	Added TLS Protocol Hardening in IBP section and Enhanced the ABAP section with TLS 1.2 information
1.11	June, 2020	Added copyright disclaimer for POSTMAN screenshots

Contents

About this document

This document is intended for customers, consulants, and partners. All of the information in this document can already be found elsewhere, but has been combined here to ensure nothing is missed.

Prerequisite

External Job Scheduling service is available begining with 1702, but received some important improvements in 1705.

Background

External Job Scheduling works as follows: there is a tool in customer network or anywhere in the cloud which is used for triggering jobs in different customer systems (on premise systems, cloud systems). This external scheduling tool could be Cisco Tidal, Control-M, CA Autosys, IBM Tivoli, Redwood BPA, etc.

> *At this point in time – January 2018 – the above mentioned tools do not yet support CSRF and the kind of web service the IBP system provides for job scheduling. One exception is Redwood BPA which has beta support of this. Please contact us if you want to use Redwood BPA.*
>
> *To work around the issue of missing support, it's possible to use a simple mechanism: the scheduling tool does not directly call the IBP web service, but calls a local program (like ABAP program, java program, CURL script, etc.). This local program then relays the request further to the IBP web service and provides the result back to the the scheduler.*
>
> *See* below *for details.*

To initiate communication from any system (here the external scheduling tool or a customer-local program) to IBP, it is important that the IBP system knows the communication partner. As IBP is based on NW 7.6, we are using IAM and NW communication management, and need the following:

a) The definition of a communication system – this is the external scheduling tool's hostname in the https request
b) The definition of a communication user – this is an artificial user defined in the IBP system to allow the external scheduling tool to logon to the IBP system and do what is needed
c) The definition of a communication arrangement – this is the combination of communication system, communication user, and the specification of what these do in the IBP system.

This way, an external tool is restricted to get the correct access but not more.

To see the Fiori Apps used to create this content, you must be an administrator, similar to how you need to bea business user role to see and use the business role template SAP_BR_ADMINISTRATOR.

Attention: *Prior to version IBP 1808 HFC07, external scheduling only allows the scheduling of single-step, non-chain templates. If you start with a job chain and remove all steps except a single one, the job will invisibly remain a job chain in the underlying implementation. To create a non-chain job, start fresh with an SAP-delivered template.*
Starting with version IBP 1808 HFC07, this limitation has been removed and job chains can be scheduled normally by the external scheduler.

Technical Background

To allow access to the OData services from outside the IBP system, you have to create a communication arrangement based on the communication scenario SAP_COM_0064. The arrangement will require a communication system and a communication user, which you also have to create. You use the Communication Management apps in IBP for those tasks.

The job scheduling method is a modifying request, therefore it can only be called using the HTTP method POST. The other OData methods can be called using the HTTP method GET. In order to protect SAP systems against cross-site request forgery attacks, processing modified OData calls requires the presence of a CSRF token. These tokens are obtained after a sucessful authenthication supported by the communication scenario (SAP_COM_0064). In the next step, you must "Fetch" the required token using a GET request with the corresponding header parameters, then you have to send back the token with subsequent calls. It's your task to protect the token information on your client-side application! The same CSRF token must be used throughout the entire session. Calls are grouped into a session through HTTP session cookies, which should also be saved from the first GET request and sent back to IBP with each subsequent HTTP request. For more details on handling CSRF tokens and session cookies, refer to the SAP Netweaver Gateway Security Guide (https://help.sap.com/viewer/p/SAP_GATEWAY).

The individual OData method calls are detailed below. In each URL, the part <SERVICE_URL> is the base URL of the OData service. You find that URL in the communication arrangement you created earlier.

OData Call to Find the Job Template

The job scheduling OData method requires two parameters to identify the job template to schedule: JobTemplateName and JobText. The parameter JobText is what you entered as job template name when you saved the parameterized data integration job template. The parameter JobTemplateName, however,

is a generated character string for job templates you save in the Application Jobs app, and is not displayed in IBP apps. The JobTemplateSet OData method returns the detailed list of all job templates available through the External Scheduler Integration OData service. The details include the JobTemplateName required for scheduling a job. The '$filter' OData query option will allow finding the template you created based on its JobText attribute. If you named your template <JOBTEXT>, the corresponding OData query could have the following URL:

```
<SERVICE_URL>/JobTemplateSet?$filter=JobTemplateText eq '<JOBTEXT>'
```

The attribute list that the query returns may be restricted to the single attribute JobTemplateName using the '$select' OData query option.

As this call is a HTTP GET request, it can be used to retrieve the CSRF token required later, adding 'X-CSRF-Token: Fetch' to the HTTP request header. The response header field 'x-csrf-token' and its value must then be sent back in the header of subsequent HTTP POST requests. The call will also set the required session cookies. The JobTemplateName value of a job template does not change. If you choose to save and hardcode that identifier instead of searching for it every time you schedule a job template, you may also use any other HTTP GET request to retrieve the required CSRF token and the HTTP cookies.

OData Call to Schedule a Job

Jobs are scheduled using the 'JobSchedule' method of the External Scheduler Integration OData service. The method can only be called as an HTTP POST request and takes two parameters:

- JobTemplateName, which you retrieved from the result of the JobTemplateSet method call
- JobText, the name with which you saved the parameterized data integration job template

If the corresponding values of the parameters are <JOBTEMPLATENAME> and <JOBTEXT>, respectively, the OData query could have the following URL:

```
<SERVICE_URL>/JobSchedule?JobTemplateName='<JOBTEMPLATENAME>'&JobText='<JOBTEXT>'
```

The call will schedule the job template for an immediate run. When scheduling is successful, the call returns an OData entity JobScheduleStatus with the property ReturnCode having the value 0. The entity will also have two properties, which identify the scheduled job: JobName and JobRunCount. The values of those properties will have to be passed to subsequent job status queries.

OData Call to Check the Status of a Job

The 'JobStatusGet' method of the External Scheduler Integration OData service returns the status of a scheduled job. The method takes two parameters: JobName and JobRunCount, which are returned by the call to the JobSchedule method. If the corresponding values of the parameters are <JOBNAME> and <JOBRUNCOUNT>, respectively, the OData query could have the following URL:

```
<SERVICE_URL>/JobStatusGet?JobName='<JOBNAME>'&JobRunCount='<JOBRUNCOUNT>'
```

The call returns a JobScheduleStatus entity. The JobStatus property of that entity is a single character showing the status of the job. The characters have the following meaning:

- R – Running (In Process)
- Y – Ready
- S – Scheduled
- A – Aborted (Failed or Canceled)

- F – Finished

The ReturnCode is not used currently.

OData Call to Cancel / Unschedule a Job

The cancellation or un-scheduling of a job can be triggered by calling the 'JobCancel' method of the External Scheduler Integration OData service. The method takes two parameters similar to the JobStatusGet method:

```
<SERVICE_URL>/JobCancel?JobName='<JOBNAME>'&JobRunCount='<JOBRUNCOUNT>'
```

Please remember that already running jobs might seem to be cancelled only but in fact still have executing elements in the different layers of the IBP system. This holds true for example for jobs which trigger SAP CPI DS (fka SAP HCI) tasks or which do heavy calculations inside HANA.

OData Call to Get Extended Info for Jobs

This method is available with SAP IBP 1711 HFC 08 and SAP IBP 1802.

The retrieval of detailed infos for a scheduled, running, or finished job can be retrieved via the extended info retrieval method 'JobinfoGet' of the External Scheduler Integration OData service. The method takes two parameters similar to the JobStatusGet method:

```
<SERVICE_URL>/JobinfoGet?JobName='<JOBNAME>'&JobRunCount='<JOBRUNCOUNT>'
```

This method functions correctly for multi-step jobs (job chains).

Returned is a list of Jobinfo entities per step which contain the following properties:

Property	Description
JobName	See above
JobRunCount	See above
JobStatus	Same as in JobStatusGet method
JobSdlDateTime	UTC time stamp containing the scheduled start date and time
JobStartDateTime	UTC time stamp containing the effective start date and time
JobEndDateTime	UTC time stamp containing the end date and time
JobAppRC	Same as in JobStatusGet method, not used currently
StepCount	Running number of the job step
StepStatus	R – Running (In Process) Y – Ready S – Scheduled A – Aborted (Failed or Canceled) F – Finished
StepStartDateTime	UTC time stamp containing the effective start date and time of the step
StepAppRC	Not used currently

OData Call to Get List of Jobs

This method is available with SAP IBP 1808.

You can use the method 'JobListGet' to retrieve a list of all jobs over a span of time. To use the method, send an HTTP request to a URL structured as follows:

```
<SERVICE_URL>/JobListGet?JobName=''&CreationTimeFrom=datetime'2018-11-20T00:00:00'&CreationTimeTo=datetime'2018-11-22T00:00:00'&JobCreator=''&JobText=''&Report=''&TemplateName=''&Variant=''
```

You should modify the values of request parameters CreationTimeFrom and CreationTimeTo to match the start and end dates of your search. Dates are formatted as year-month-day T hour:minute:second. The time zone is UTC (also known as GMT or Z).

The response will be a list of jobs created in the specified time range, and each job will have a series of properties. Most of the properties are the same as what is returned by method **JobinfoGet** (as explained in that section, above), with the following additions:

Property	Description
JobCreaDateTime	UTC timestamp for when the job was created
JobCreator	The ID of the user that created the job
JobLogStatus	The status of the job's logs
JobSchedDateTime	UTC time stamp containing the scheduled start date and time
JobText	The user-defined name of the job
NrSteps	The number of steps in this job template
Report	The name of the associated ABAP program
StepApRC	Not used currently
TemplateName	The unique ID of the template used to create the job
Variant	The ID of the associated variant

Further Information

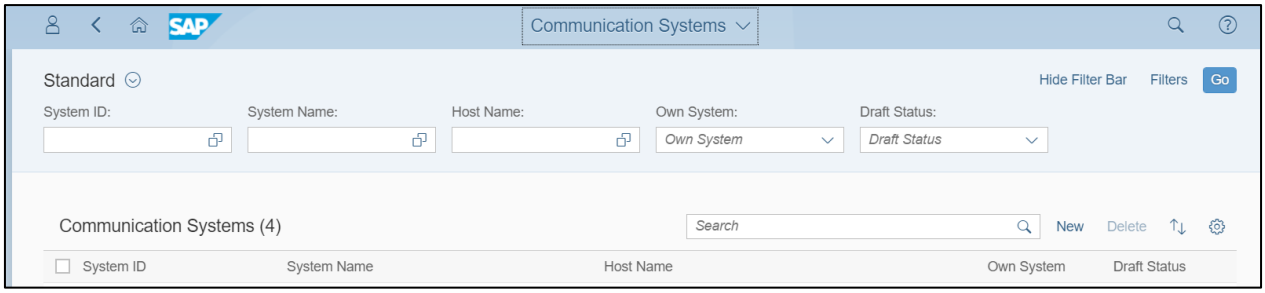
The SAP S/4HANA Cloud documentation has a detailed description of the Communication Management apps and the underlying process. Starting from the SAP Help Portal page for SAP S/4HANA Cloud (https://help.sap.com/viewer/p/SAP_S4HANA_CLOUD), choose the Product Assistance link of your preferred language and then navigate to Generic Information -> General Functions for the Key User -> Communication Management.

For more information on OData and network communication security, refer to the SAP Gateway Security Guide. The link is available from the SAP Help Portal: find the Security section on the SAP Help Portal page for SAP Gateway (https://help.sap.com/viewer/p/SAP_GATEWAY). The guide, among others, will provide more details on the topics of Cross-Site Request Forgery Protection, Handling Confidential Data in OData URLs, and Cross-Site Scripting (XSS) Protection.

Also useful is XSRF Protection for REST Services on help.sap.com

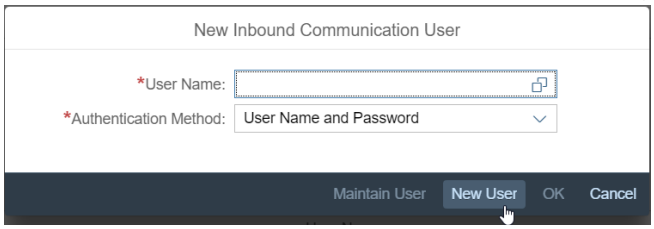
Preparation

Communication System & Communication User



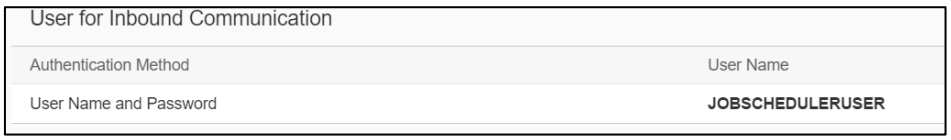
Open the *Communication Systems Fiori App* and click "New" and provide a meaningful system ID and description for your external scheduling tool. We here will use "JOBSCHEDULER" for demo purpose. Then we choose the correct hostname as it is used as the source of the request to the IBP system.

Next we add a new user for inbound communication (inbound from the IBP system's point of view).



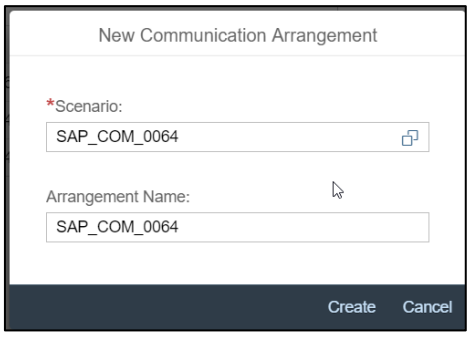
We are then forwarded to the *Communication User Fiori App* and can create a completely new IBP local user which is used to identify the inbound communication from the external scheduler. Here in the demo, we used JOBSCHEDULERUSER and let IBP generate a password.

In the communication system we specify "Username and Password" as authentication method.

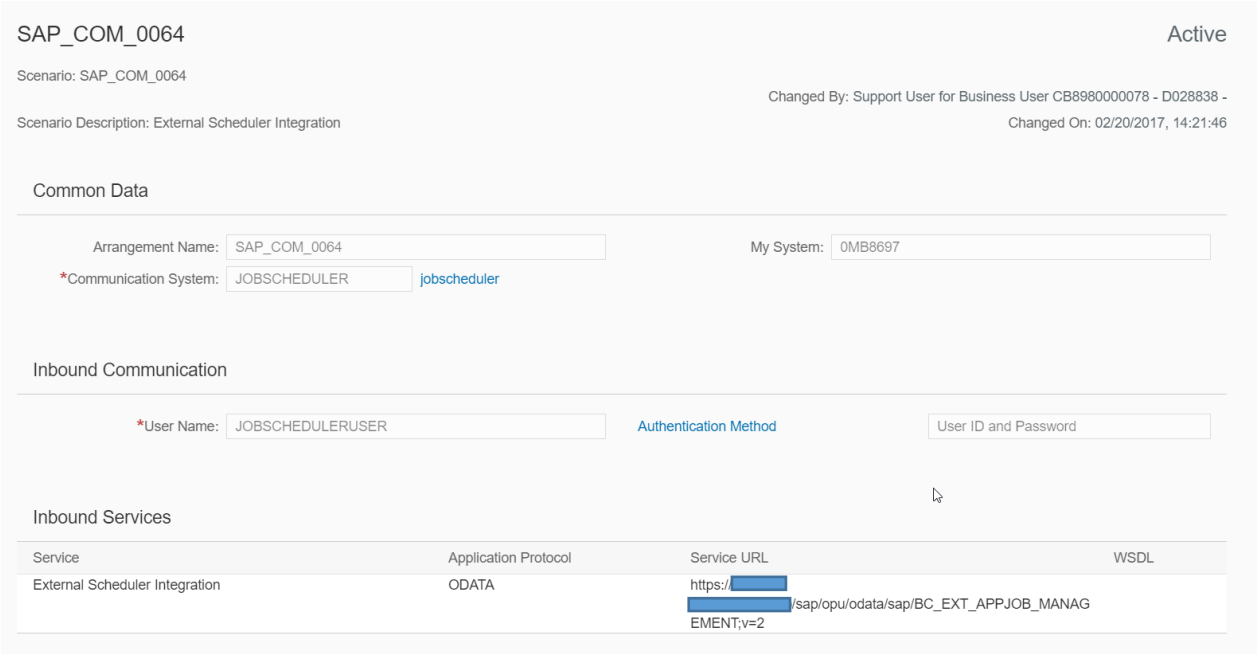


Communication Arrangement

In the *Communication Arrangement Fiori App* we choose "New" and create a new communication arrangement based on scenario SAP_COM_0064.



We select the communication system and communication user from the previous step.

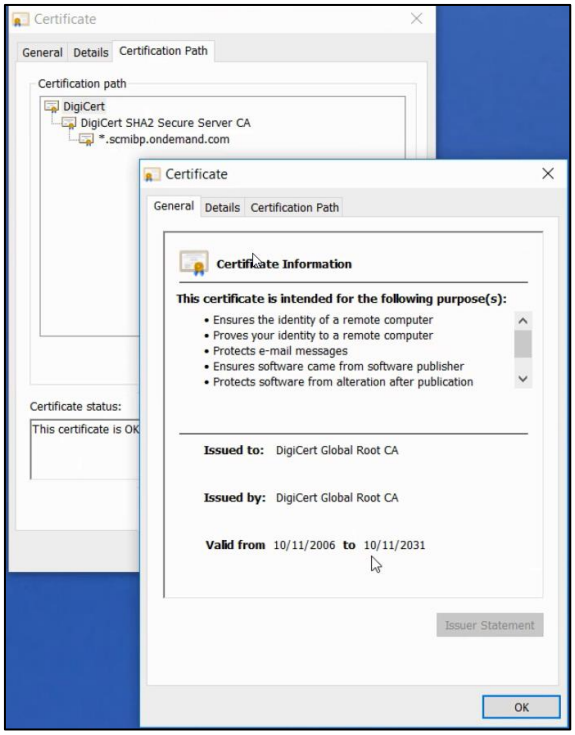


You can see in the lower right the URL which is to be used by the external scheduling tool as basis.

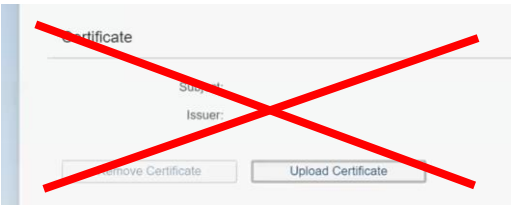
Root Certificate

The communication between your landscape and IBP Cloud System using https protocol secured with TLS. You have to ensure your client already included IBP System's Root CA into the respective trust store. In common operating systems and browsers this Root CA already included into default trust store, you need to explicitly include it only if you using a customized and/or empty initial trustlist. As root certificate, make sure you downloaded the "myXYZ-API.scmibp.ondemand.com" certificate and <u>not</u> the "myXYZ.scmibp.ondemand.com" certificate! If the "https:// myXYZ-API.scmibp.ondemand.com" redirects you to the identity authentication service logon, download the root certificate from the "https:// myXYZ-API.scmibp.ondemand.com?saml2=disabled".

IBP Cloud system server certificate looks as below, depending on the used implementation, you need to store the root certificate of it. The certificate looks like this:



Please note: The IBP system certificate and the used Root CA certificate are different than your communication user client-certificate! Above server certificate is presented in the beginning of the TLS handshake and must be trusted by your client implementation. Uploading this certificate to the communication user is **not** required!!

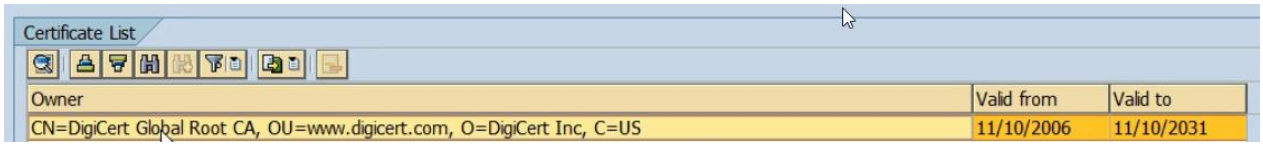


You need to maintain your communication user client-certificate if you decide to select certificate-based client authentication method.The communication user certificate purpose to map your communication user with existing client.

- When using **CURL** on a windows based machine, the certificate will not be installed manually, but it will be done automatically.
- When using **Java**, the certificate from your myXXXX-api will possibly need to be added to the java certificate store (jks-file).
- When using **ABAP** as caller, the root certficiate needs to be added via STRUST in sapsslc.pse (default) key store.

  Choose this: 

And after adding it will look like this:



TLS Protocol Hardening in IBP

After August 4th, 2019 IBP will no longer support protocol versions older than TLS 1.2 in your IBP non-production environments and production environments in order to align with the industry best practices for security and data integrity. Any connections to IBP that rely on older version than TLS 1.2 will fail.

You need to enable TLS 1.2 for your inbound connections to IBP, including the communication via external scheduler.

For more details, see Note 2723410 – TLS Protocol Hardening in IBP and Note 510007 - Setting up SSL on Application Server ABAP

First Tests

Retrieve Templates

To do a very first simple test to see whether this scheduling works at all, you can specify your local machine as communication system and another artificial user for your testing.

The <SERVICE_URL> you have obtained from the communication arrangement can be used from within your browser. Simply enter the following URL:

```
<SERVICE_URL>/JobTemplateSet
```

So it might look like:

```
https://theIBPsystem/sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=0002/JobTemplateSet
```

(of course replace “theIBPsystem” with the hostname of the IBP system)

If you use this as URL in your browser, you will be asked for a user/password – use the one you specified in the communication user. For the communication user, you usually need to hand over the user name you specified in the creation of the communication user.

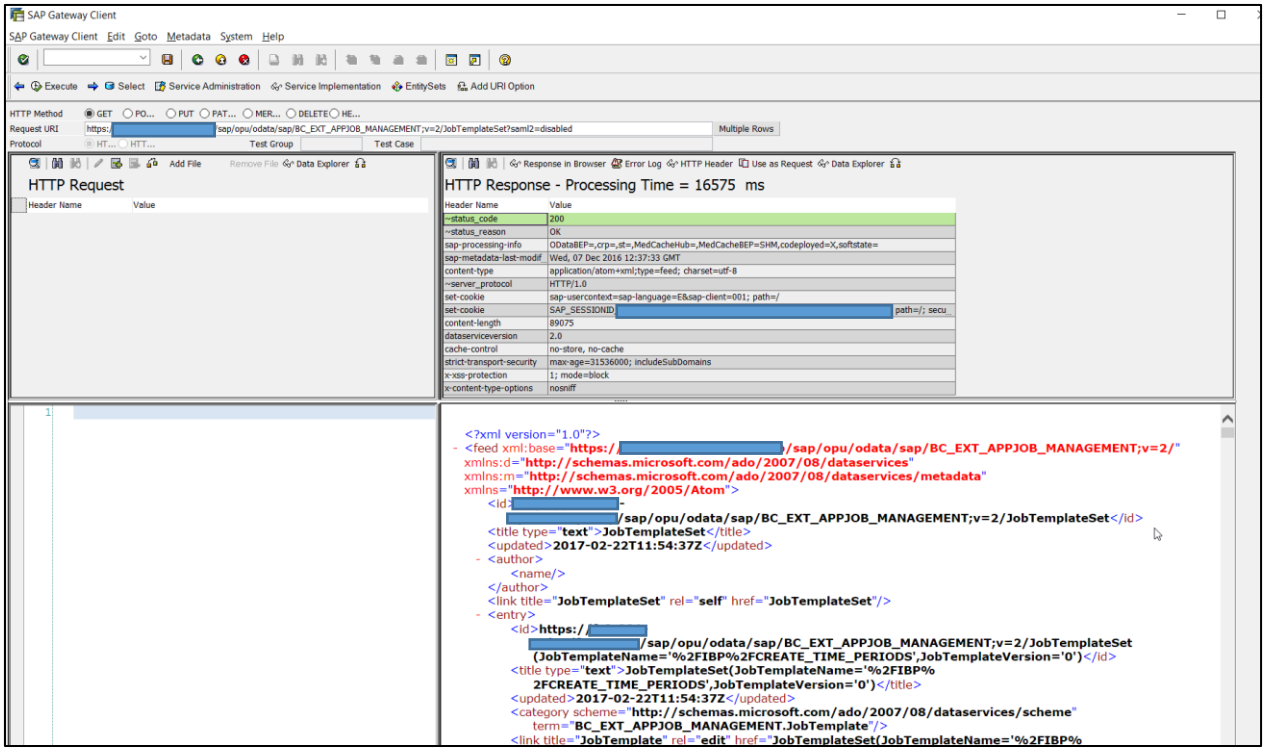
The response from the IBP system will be something like this:

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<feed xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
 xml:base="https:/[redacted]/sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2/">
 ▼<id>
    https://[redacted]/sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2/JobTemplateSet
  </id>
  <title type="text">JobTemplateSet</title>
  <updated>2017-02-20T16:13:50Z</updated>
 ▼<author>
    <name/>
  </author>
  <link href="JobTemplateSet" rel="self" title="JobTemplateSet"/>
 ▼<entry>
  ▼<id>
     https://[redacted]/sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2/JobTemplateSet(JobTemplateName='%2FIBP%2FCREATE_TIME_PERIODS',JobTemplateVersion='0')
   </id>
  ▼<title type="text">
     JobTemplateSet(JobTemplateName='%2FIBP%2FCREATE_TIME_PERIODS',JobTemplateVersion='0')
   </title>
   <updated>2017-02-20T16:13:50Z</updated>
   <category term="BC_EXT_APPJOB_MANAGEMENT.JobTemplate" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
   <link href="JobTemplateSet(JobTemplateName='%2FIBP%2FCREATE_TIME_PERIODS',JobTemplateVersion='0')" rel="edit" title="JobTemplate"/>
  ▼<content type="application/xml">
   ▼<m:properties xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
      <d:JobTemplateName>/IBP/CREATE_TIME_PERIODS</d:JobTemplateName>
      <d:JobTemplateVersion>0</d:JobTemplateVersion>
      <d:JobTemplateStepCount>1</d:JobTemplateStepCount>
      <d:JobPeriodicGranularity/>
      <d:JobReportName>/IBP/CREATE_TIME_PERIODS</d:JobReportName>
      <d:JobUserName/>
      <d:JobPeriodicValue>000</d:JobPeriodicValue>
      <d:JobTemplateText>Create Time Periods for Time Profiles</d:JobTemplateText>
      <d:CreationDateTime>2015-09-01T15:42:36.0944660</d:CreationDateTime>
      <d:CreationUserName>SAP</d:CreationUserName>
      <d:LastChangeDateTime>2015-12-17T18:42:23.2763080</d:LastChangeDateTime>
      <d:LastChangeUserName>SAP</d:LastChangeUserName>
      <d:SupportsTestModeInd>false</d:SupportsTestModeInd>
    </m:properties>
   </content>
  </entry>
 ▼<entry>
  ▼<id>
     https:/[redacted]/sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2/JobTemplateSet(JobTemplateName='%2FIBP%2FLOG_DELETION',JobTemplateVersion='0')
   </id>
  ▼<title type="text">
     JobTemplateSet(JobTemplateName='%2FIBP%2FLOG_DELETION',JobTemplateVersion='0')
   </title>
   <updated>2017-02-20T16:13:50Z</updated>
   <category term="BC_EXT_APPJOB_MANAGEMENT.JobTemplate" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
   <link href="JobTemplateSet(JobTemplateName='%2FIBP%2FLOG_DELETION',JobTemplateVersion='0')" rel="edit" title="JobTemplate"/>
  ▼<content type="application/xml">
   ▼<m:properties xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
      <d:JobTemplateName>/IBP/LOG_DELETION</d:JobTemplateName>
      <d:JobTemplateVersion>0</d:JobTemplateVersion>
      <d:JobTemplateStepCount>1</d:JobTemplateStepCount>
      <d:JobPeriodicGranularity/>
      <d:JobReportName>/IBP/R_LOG_DELETE</d:JobReportName>
      <d:JobUserName/>
```


You here can see all the job templates which you are able to schedule. Search the list and identify the templates you need to schedule later. Pick the entry inside the JobTemplateName tag for later use.

If you do not want to use your browser or have issues with that, try using SAPGUI transaction /IWFND/GW_CLIENT.



Schedule Job

To schedule a job, you need to send a POST request. To do this, there are several tools available. Here, we are using the SAP Gateway Client via SAPGUI transaction /IWFND/GW_CLIENT again.

As a URL, use
`<SERVICE_URL>/JobSchedule?JobTemplateName='<jobtemplatename>'&JobText='<title>'&JobUser='<jobuser>'`

The <SERVICE_URL> is the service's base URL, as copied from the communication arrangement.

The <jobtemplatename> is the name of the job template. You can find this in the JobTemplateSet call from above.

For <title>, provide the title for this job so it can easily be identified – maybe use the job template title (JobTemplateText property) and add "- scheduled via external scheduling tool".

For <jobuser>, provide the business user the job should run under. This can be used to specify, for example, the visibility filter for the IBP jobs. You can find the jobs in the Fiori App under this users name.

If you then POST this, you might get an error, such as what follows:

HTTP Method GET PO... PUT PAT... MER... DELETE HE...

Request URI https:/[redacted]sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2/JobSchedule?JobTemplateName='ZHDVKOF2BIAPONJPTD3U3FCNCR(Multiple Rows

Protocol HT... HTT... Test Group Test Case

Add File Remove File Data Explorer

HTTP Request

Header Name	Value
X-CSRF-Token	285ukmCo8zk8yiDfTHh4qA==

Response in Browser Error Log HTTP Header Use as Request Data Explorer

HTTP Response - Processing Time = 645 ms

Header Name	Value
~status_code	400
~status_reason	Bad Request
sap-processing-info	ODataBEP=,crp=,st=,MedCacheHub=,codeployed=X,softstate=
~server_protocol	HTTP/1.0
content-type	application/xml;charset=utf-8
content-length	1778
dataserviceversion	1.0
strict-transport-security	max-age=31536000; includeSubDomains
x-xss-protection	1; mode=block


```
<?xml version="1.0" encoding="UTF-8"?>
- <error xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
    <code>APJ_RT/032</code>
    <message xml:lang="en">Job scheduling failed</message>
  - <innererror>
    - <application>
        <component_id>BC-CCM-BTC</component_id>
        <service_namespace>/SAP/</service_namespace>
        <service_id>BC_EXT_APPJOB_MANAGEMENT</service_id>
        <service_version>0002</service_version>
      </application>
      <transactionid>545B03A9D9230230E0058AD11AF3AC59</transactionid>
      <timestamp/>
    - <Error_Resolution>
        <SAP_Transaction/>
        <SAP_Note>See SAP Note 1797736 for error analysis
            (https://service.sap.com/sap/support/notes/1797736)</SAP_Note>
      </Error_Resolution>
    - <errordetails>
      - <errordetail>
          <code>/IBP/CM_PL_OP/201</code>
          <message>Planning area DS6A isn't valid for you.</message>
          <propertyref/>
          <severity>error</severity>
          <target/>
        </errordetail>
      - <errordetail>
          <code>/IBP/CM_PL_OP/200</code>
          <message>You need authorization to use planning area DS6A.</message>
          <longtext_url>/sap/opu/odata/iwbep/message_text;o=/</longtext_url>
          <propertyref/>
          <severity>error</severity>
          <target/>
        </errordetail>
      - <errordetail>
          <code>/IBP/CM_PL_OP/000</code>
          <message>You need authorization to execute planning operator COPY, ID 200, area
```


XML Ln 1 Col 1 NUM

If the job scheduling worked, you will get something like the following:

Execute Select Service Administration Service Implementation EntitySets Add URI Option

HTTP Method GET PO... PUT PAT... MER... DELETE HE...

Request URI https:/[redacted]sap/opu/odata/SAP/BC_EXT_APPJOB_MANAGEMENT;v=2/JobSchedule?JobTemplateName='APJ_DEMO_REPORT'&JobText='T(Multiple Rows

Protocol HT... HTT... Test Group Test Case

Add File Remove File Data Explorer

HTTP Request

Header Name	Value
X-CSRF-Token	ePG1U0RrmgEgT1xvhdJVog==

Response in Browser Error Log HTTP Header Use as Request Data Explorer

HTTP Response - Processing Time = 219 ms

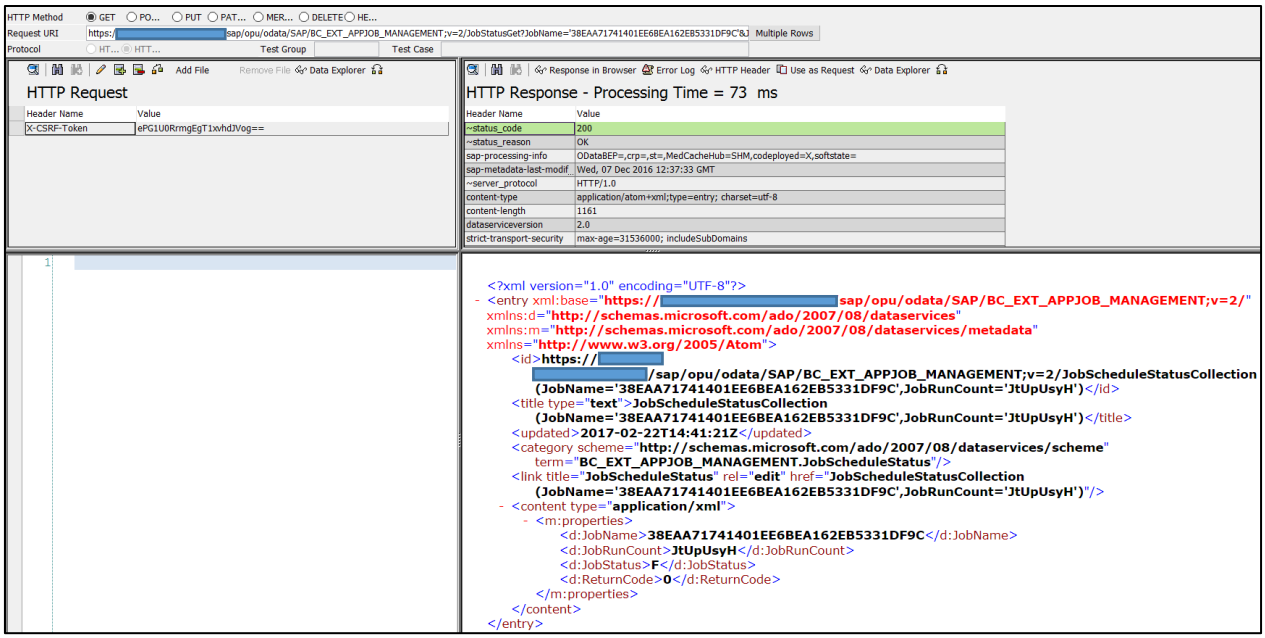
Header Name	Value
~status_code	200
~status_reason	OK
sap-processing-info	ODataBEP=,crp=,st=,MedCacheHub=Table,codeployed=X,softstate=
~server_protocol	HTTP/1.0
content-type	application/atom+xml;type=entry; charset=utf-8
content-length	1147
dataserviceversion	2.0
strict-transport-security	max-age=31536000; includeSubDomains
x-xss-protection	1; mode=block


```
<?xml version="1.0" encoding="UTF-8"?>
- <entry xml:base="https://[redacted]/sap/opu/odata/SAP/BC_EXT_APPJOB_MANAGEMENT;v=2/"
  xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
  xmlns="http://www.w3.org/2005/Atom">
    <id>https://[redacted]
        [redacted]/sap/opu/odata/SAP/BC_EXT_APPJOB_MANAGEMENT;v=2/JobScheduleStatusCollection
        (JobName='38EAA71741401EE6BEA162EB5331DF9C',JobRunCount='JtUpUsyH')</id>
    <title type="text">JobScheduleStatusCollection
        (JobName='38EAA71741401EE6BEA162EB5331DF9C',JobRunCount='JtUpUsyH')</title>
    <updated>2017-02-22T14:27:51Z</updated>
    <category scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"
        term="BC_EXT_APPJOB_MANAGEMENT.JobScheduleStatus"/>
    <link title="JobScheduleStatus" rel="edit" href="JobScheduleStatusCollection
        (JobName='38EAA71741401EE6BEA162EB5331DF9C',JobRunCount='JtUpUsyH')"/>
  - <content type="application/xml">
    - <m:properties>
        <d:JobName>38EAA71741401EE6BEA162EB5331DF9C</d:JobName>
        <d:JobRunCount>JtUpUsyH</d:JobRunCount>
        <d:JobStatus/>
        <d:ReturnCode>0</d:ReturnCode>
      </m:properties>
    </content>
  </entry>
```

Check Status of a Job

If you know the JobName and the JobRunCount this can be achieved by calling:

```
<SERVICE_URL>/JobStatusGet?JobName='<jobname>'&JobRunCount='<jobruncount>'
```



The letter **R** means Running (In Process) and **F** means Finished. The full list of status letters can be found in the section "OData Call to Check the Status of a Job".

The ReturnCode is not used.

Cancel a Scheduled or Running Job

This is a similar call as when retrieving the job status and can be done via:

```
<SERVICE_URL>/JobCancel?JobName='<jobname>'&JobRunCount='<jobruncount>'
```

This will either remove a job from the schedule or abort an already running job.

Connection Issues

We see issues when accessing the hostname mentioned in the communication arrangement from time to time.

Incorrect Server

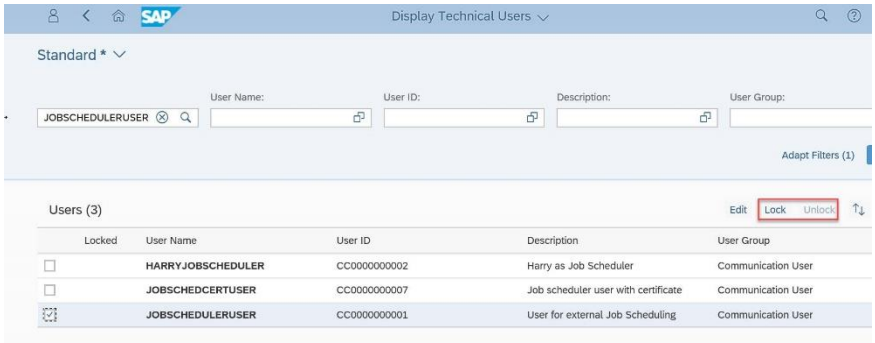
One source of issues in the past has been that the incorrect server was used. At the moment, the servername for the Fiori and Excel Add-In is "myNNNNN.scmibp.ondemand.com" and the one mentioned in the communication arrangement is "myNNNNN-**api**.scmibp.ondemand.com". The one with the "-api" is the correct one.

In case you have issues with authentication, you can add parameter "**saml2=disabled**". This helps for testing the services and to identify whether the "-api" servername is not configured correctly by SAP Cloud Operations or whether the issue is somewhere else. Usage of "saml2=disabled" only works on the "myNNNNN.scmibp.ondemand.com" site and not with "-api" and is not intended for productive tool usage.

Error 401- User Unauthorized

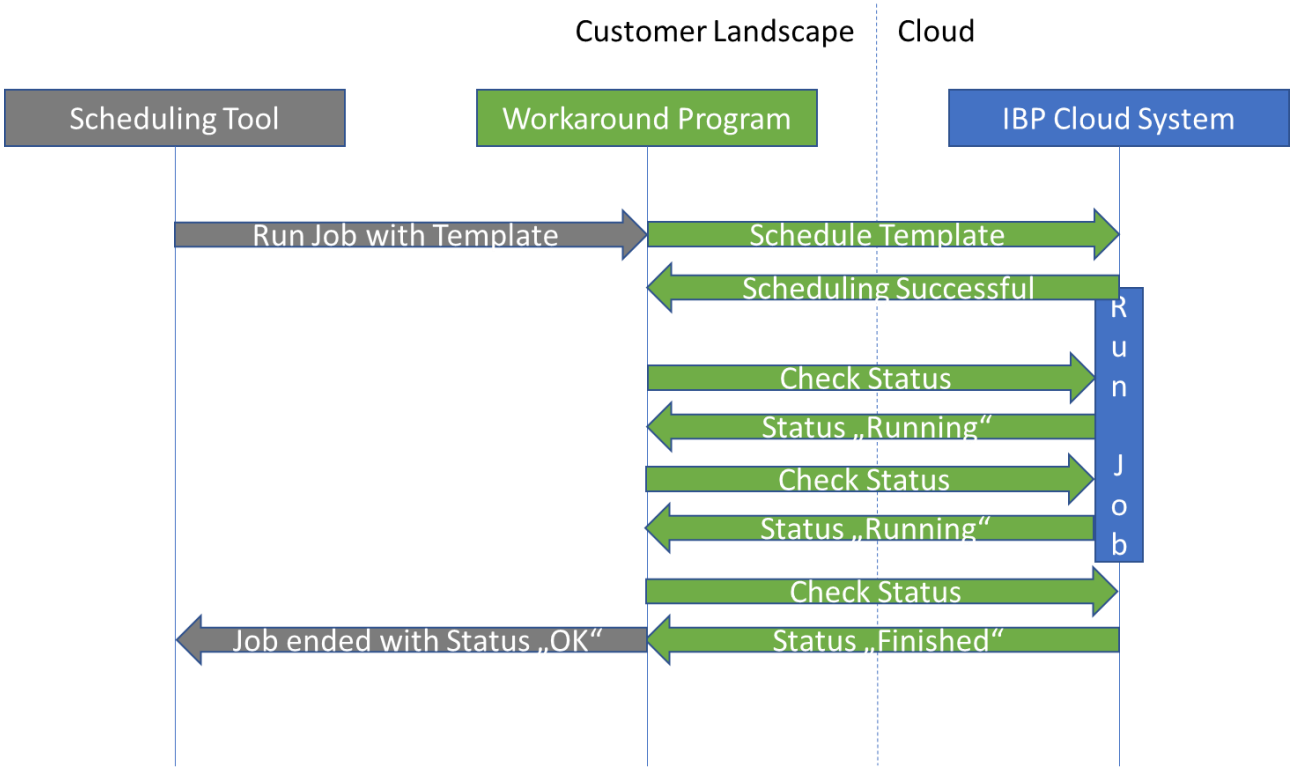
If you receive a *401- User Unauthorized* error when calling one of the Odata service operations and the calls to the service worked fine before, one of the reasons for this error could be that the Communication User has been locked due to several unsuccessfull login attempts.

The communication user can be unlocked using the Display Technical Users App, which is delieverd under the Identity and Access Management FIORI Group.



Workarounds for Scheduling Tools not supporting the API

The best way to get job scheduling done is to use the support of the API by the scheduling tool. But this may not be possible, so here is the proposed way to work around this issue.



1. Create a java program, CURL script, ABAP report, etc. inside your local landscape (“workaround program” in the above picture).
2. Implement the program/report/script in the following way:

    a. it triggers the scheduling of an application job template,
    b. then waits for the job to run,
    c. then waits for the job to finish or fail,
    d. finally it forwards the status back to the caller of the program/report/script
3. The scheduling tool is scheduling this local workaround program.
4. Best way would be if the scheduling tool could hand over the application job template ID, so you would only have a single workaround program and could schedule all available application job templates by simply parameterizing (via command line parameters, ABAP variants, etc.).
5. Ideally you implement a means when the workaround program is cancelled, then also the IBP application job is cancelled.

Code Samples

General Information on the Code Samples

Please remember that these code samples are provided as examples. They are using basic authentication, but a better approach would be to use client-certificates (which is available of course in our API URLs).

Important Elements

The first request has to be either a HEAD or a GET request, which provides authentication and requests for a CSRF token. Authentication is done via basic authentication. The CSRF token is retrieved and later added to the next requests. With the first request, a session cookie is returned in the response headers, which is to be added to all follow up POST requests.

The content of error pages and the response when all went well is XML-like formatted which helps in retrieving error codes, error messages, template names, etc.

Java

Yellow marked fields have to be replaced by correct values.

```
package sched;

import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;
import java.security.cert.Certificate;
import java.io.*;

import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLPeerUnverifiedException;
import java.util.Base64;
import java.util.Map;
import java.util.Set;
import java.util.List;

public class Sched {

    public static void main(String[] args) {

        System.out.println("Starting...");
        String vBaseURL      = "https://<hostname>/sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2";
        String vUserPassword = "<technical user>:<technical user pwd>";
        String vTemplateName = "<template>";
        String vTemplateText = "Scheduled via Java";
        String vExecuteUser  = "<job user id>";

        String vGetTemplates     = "/JobTemplateSet?";
        String vScheduleTemplate = "/JobSchedule?JobTemplateName='<templatename>'&JobText='<jobtext>'&JobUser='<jobuser>'";
        String vGetStatus        = "/JobStatusGet?JobName='<jobname>'&JobRunCount='<jobruncount>'";

        URL oURL;
        try {
            try {
                vScheduleTemplate = vScheduleTemplate.replace("<templatename>", URLEncoder.encode(vTemplateName,"UTF-8"));
                vScheduleTemplate = vScheduleTemplate.replace("<jobtext>",      URLEncoder.encode(vTemplateText,"UTF-8"));
                vScheduleTemplate = vScheduleTemplate.replace("<jobuser>",      URLEncoder.encode(vExecuteUser,"UTF-8"));
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }

            oURL = new URL(vBaseURL + vGetTemplates);

            javax.net.ssl.HttpsURLConnection oConnection = (HttpsURLConnection) oURL.openConnection();
            System.out.println(oURL.toString());

            oConnection.setRequestMethod("GET");
            oConnection.setDoOutput(true); // enable header values to be transferred

            String vUserPasswordEncoded = Base64.getEncoder().encodeToString(vUserPassword.getBytes());
            System.out.println("base64 = " + vUserPasswordEncoded);

            oConnection.setRequestProperty("Authorization", "Basic " + vUserPasswordEncoded);
            oConnection.setRequestProperty("X-Requested-With", "XMLHttpRequest");
            oConnection.setRequestProperty("Content-Type", "application/atom+xml");
            oConnection.setRequestProperty("DataServiceVersion", "2.0");
```

```
oConnection.setRequestProperty("X-CSRF-Token", "Fetch");

System.out.println("HTTPS GET request...");

System.out.println("Response: " + oConnection.getResponseCode() + ", " + oConnection.getResponseMessage());
BufferedReader oBufferedReader;

if (oConnection.getResponseCode() == 200) {
    oBufferedReader = new BufferedReader(new InputStreamReader(oConnection.getInputStream()));
} else {
    oBufferedReader = new BufferedReader(new InputStreamReader(oConnection.getErrorStream()));
}

// simply output the content to the console
String vLine = null;
String vTemplateList = null;
do {
    vLine = oBufferedReader.readLine();
    if (vLine != null) {
        vLine = vLine.replaceAll("[^\\x20-\\xFF]", "");
        if (vTemplateList == null)
            vTemplateList = vLine;
        else
            vTemplateList = vTemplateList + vLine;
        System.out.println(vLine);
    }
} while (vLine != null);
System.out.println();

if (oConnection.getResponseCode() == 200) {
    // show possible templates list
    System.out.println("Possible templates to schedule as job");
    int vPos = 0;
    do {
        int vTemplateNamePos = vTemplateList.indexOf("<d:JobTemplateName>", vPos);
        int vTemplateTextPos = vTemplateList.indexOf("<d:JobTemplateText>", vPos);
        if ((vTemplateNamePos >= 0) && (vTemplateTextPos >= 0)) {
            vTemplateNamePos += 19;
            vTemplateTextPos += 19;
            int vTemplateNamePosEnd = vTemplateList.indexOf("<", vTemplateNamePos);
            int vTemplateTextPosEnd = vTemplateList.indexOf("<", vTemplateTextPos);
            String vTemplateListName = vTemplateList.substring(vTemplateNamePos, vTemplateNamePosEnd);
            String vTemplateListText = vTemplateList.substring(vTemplateTextPos, vTemplateTextPosEnd);
            System.out.println("Template " + vTemplateListName + ": " + vTemplateListText);
        }
        vPos = Math.max(vTemplateNamePos, vTemplateTextPos) + 1;
    } while (vTemplateList.indexOf("<d:JobTemplateName>", vPos) >= 0);
}

String vCookie = "";

// simply output all parameters to the console
System.out.println("Response headers:");

Map<String, List<String>> oHeader = oConnection.getHeaderFields();
Set<Map.Entry<String, List<String>>> oHeaderEntrySet = oHeader.entrySet();
for (Map.Entry<String, List<String>> oHeaderEntry : oHeaderEntrySet) {
    String vHeaderName = oHeaderEntry.getKey();
    System.out.print(vHeaderName + " : ");
    List<String> oHeaderValues = oHeaderEntry.getValue();
    for (String vHeaderValue : oHeaderValues) {
        if (vHeaderValue != null)
            vHeaderValue = vHeaderValue.replaceAll("[^\\x20-\\xFF]", "");
        System.out.println(vHeaderValue);
    }

    if ((vHeaderName != null) && (vHeaderName.equalsIgnoreCase("Set-Cookie"))) {
        for (String vHeaderValue : oHeaderValues) {
            vCookie = vCookie + "; " + vHeaderValue;
        }
    }
    System.out.println();
}

System.out.println("Set-Cookie: " + vCookie);

String vCSRFToken = oConnection.getHeaderField("x-csrf-token");
System.out.println("CSRF Token: " + vCSRFToken);

// POST
oURL = new URL(vBaseURL + vScheduleTemplate);
System.out.println(oURL.toString());
javax.net.ssl.HttpsURLConnection oConnectionPOST = (HttpsURLConnection) oURL.openConnection();

oConnectionPOST.setRequestMethod("POST");
oConnectionPOST.setDoOutput(true); // enable header values to be transferred

oConnectionPOST.setRequestProperty("Cookie", vCookie);
```

```
oConnectionPOST.setRequestProperty("X-CSRF-Token", vCSRFToken); // 403 if missing

System.out.println("HTTPS POST request...");

System.out.println("Response: " + oConnectionPOST.getResponseCode() + ", " + oConnectionPOST.getResponseMessage());
if (oConnectionPOST.getResponseCode() == 200) {
    oBufferedReader = new BufferedReader(new InputStreamReader(oConnectionPOST.getInputStream()));
} else {
    oBufferedReader = new BufferedReader(new InputStreamReader(oConnectionPOST.getErrorStream()));
}
String vResponse = null;
vLine = null;
do {
    vLine = oBufferedReader.readLine();
    if (vLine != null) {
        vLine = vLine.replaceAll("[^\\x20-\\xFF]", "");
        if (vResponse == null)
            vResponse = vLine;
        else
            vResponse = vResponse + vLine;
        System.out.println(vLine);
    }
} while (vLine != null);

String vJobName     = null;
String vJobRunCount = null;

if (oConnectionPOST.getResponseCode() == 200) {
    // show scheduled job ID
    int vJobNamePos = vResponse.indexOf("<d:JobName>");
    int vJobRunCountPos = vResponse.indexOf("<d:JobRunCount>");
    if ((vJobNamePos >= 0) && (vJobRunCountPos >= 0)) {
        vJobNamePos += 11;
        vJobRunCountPos += 15;
        vJobName     = vResponse.substring(vJobNamePos,     vJobNamePos + 32);
        vJobRunCount = vResponse.substring(vJobRunCountPos, vJobRunCountPos + 8);
        System.out.println("Scheduled JobName '" + vJobName + "', JobRunCount '" + vJobRunCount + "'");
    }
} else {
    // show error messages
    int vPos = 0;
    do {
        int vCodePos = vResponse.indexOf("<code>", vPos);
        int vMessagePos = vResponse.indexOf("<message", vPos);
        if ((vCodePos >= 0) && (vMessagePos >= 0)) {
            vCodePos += 6;
            vMessagePos = vResponse.indexOf(">", vMessagePos + 8) + 1;
            int vCodePosEnd = vResponse.indexOf("<", vCodePos);
            int vMessagePosEnd = vResponse.indexOf("<", vMessagePos);
            if ((vCodePosEnd >= 0) && (vMessagePosEnd >= 0) &&
                (vMessagePos < vMessagePosEnd) && (vCodePos < vCodePosEnd)) {
                  String vErrCode = vResponse.substring(vCodePos, vCodePosEnd);
                  String vErrMessage = vResponse.substring(vMessagePos, vMessagePosEnd);
                  System.out.println("Error " + vErrCode + " - " + vErrMessage);
            }
        }
        vPos = Math.max(vMessagePos, vCodePos);
    } while (vResponse.indexOf("<code>", vPos) > 0);
}

//
if ((vJobName != null) && (vJobRunCount != null)) {

    try {
        vGetStatus = vGetStatus.replace("<jobname>",     URLEncoder.encode(vJobName,"UTF-8"));
        vGetStatus = vGetStatus.replace("<jobruncount>", URLEncoder.encode(vJobRunCount,"UTF-8"));
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }

    oURL = new URL(vBaseURL + vGetStatus);

    javax.net.ssl.HttpsURLConnection oConnectionStatus = (HttpsURLConnection) oURL.openConnection();
    System.out.println(oURL.toString());

    oConnectionStatus.setRequestMethod("GET");
    oConnectionStatus.setDoOutput(true); // enable header values to be transferred
    oConnectionStatus.setRequestProperty("Cookie", vCookie);
    oConnectionStatus.setRequestProperty("X-CSRF-Token", vCSRFToken);

    System.out.println("HTTPS GET request...");

    System.out.println("Response: " + oConnectionStatus.getResponseCode() + ", " +
                       oConnectionStatus.getResponseMessage());

    if (oConnectionStatus.getResponseCode() == 200) {
        oBufferedReader = new BufferedReader(new InputStreamReader(oConnectionStatus.getInputStream()));
    } else {
```

```
                oBufferedReader = new BufferedReader(new InputStreamReader(oConnectionStatus.getErrorStream()));
            }

            // simply output the content to the console
            vLine = null;
            String vJobStatus = null;
            do {
                vLine = oBufferedReader.readLine();
                if (vLine != null) {
                    vLine = vLine.replaceAll("[^\\x20-\\xFF]", "");
                    if (vJobStatus == null)
                          vJobStatus = vLine;
                    else
                          vJobStatus = vTemplateList + vLine;
                    System.out.println(vLine);
                }
            } while (vLine != null);
            System.out.println();

            if (oConnectionStatus.getResponseCode() == 200) {
                int vJobStatusPos = vJobStatus.indexOf("<d:JobStatus>");
                if (vJobStatusPos >= 0) {
                    vJobStatusPos += 13;
                    int vJobStatusPosEnd = vJobStatus.indexOf("<", vJobStatusPos);
                    String vJobStatusCode = vJobStatus.substring(vJobStatusPos, vJobStatusPosEnd);
                    String vJobStatusText;
                    char vJobStatusCodeC = vJobStatusCode.charAt(0);
                    switch (vJobStatusCodeC) {
                    case 'F': vJobStatusText = "Finished"; break;
                    case 'R': vJobStatusText = "In Process"; break;
                    case 'Y': vJobStatusText = "Ready"; break;
                    case 'S': vJobStatusText = "Scheduled"; break;
                    case 'A': vJobStatusText = "Failed"; break;
                    case 'C': vJobStatusText = "Cancelled"; break;
                    default:  vJobStatusText = "unknown";
                    }
                    System.out.println("Job status is '" + vJobStatusCode + "': " + vJobStatusText);
                }
            }
        }
    } catch (java.net.MalformedURLException e) {
        e.printStackTrace();
    } catch (java.io.IOException e) {
        e.printStackTrace();
    }
  }
}
```

CURL

See https://curl.haxx.se/ for details on CURL itself.

This is sample code when used with a shell script. Remember to use "^" as escape character for "&" in command line when testing on windows with curl. Using this script is advisable.

When using Windows, simplest way to use the script is to install git bash shell from here: https://git-scm.com/downloads or via https://gitforwindows.org/, place the script in the executables folder, make sure a folder exists named "log", and run the script. The command line it would look for example like this: `"c:\Program Files\Git\bin\bash.exe" --login -i -- c:\ibp\schedule.sh <job template> <job template text> <job user id>`

You can run this script from the scheduling tool and the script will wait till the scheduled IBP job ends. At the moment there is no "timeout", but this can be done easily in the loop at the end of the script.

You might want to create a second script for cancelling a job, this would require to store the jobname and jobruncount to another file and use this from the cancelling script. The script would look similar, but with an adjusted POST request and without a loop at the end.

The CURL example here stores the CSRF token locally on the file system. Please make sure authorizations for users are set correctly and restrict access so nobody can misuse the CSRF token.

Yellow marked fields have to be replaced by correct values.

```
# External Scheduler Script for IBP

# Job template variables (passed as arguments at command line)
job_template_name=$1
job_text=$2
job_user=$3
# Max time to wait for job to run (3600 = 1 hour)
run_time_limit=$4

# If not specified, default max runtime is 1 hour
if [ -z "$run_time_limit" ];then
    run_time_limit=3600;
fi

now=$(date +"%F_%H_%M_%S")

# Username and password are that of a user with privileges to access IBP system api
username="<communication user>"
password="<communication user password>"

#Encode password for support of special characters
enc="$(echo -n "$username:$password" | base64)"
encauth="Authorization: Basic $enc"

# base service URL
hostname="<myhost>-api.scmibp.ondemand.com"
site_url="https://$hostname/sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2"

# metadata suffix (used to fetch csrf-token)
md_url="$site_url/\$metadata"

# data parameters passed to our target service (Job Scheduling)

data="/?JobTemplateName='$job_template_name'&JobText='$job_text'&JobUser='$job_user'"

# target service (in this case the scheduling service)
es_url="$site_url/JobSchedule$data"

# log file
log_filename="./log/$job_text - $now.txt"

echo "Scheduling Job URL will be: " $es_url | tee -a $log_filename

# service used to monitor status
stat_url="$site_url/JobStatusGet?"

echo "Status URL will be: " $stat_url | tee -a $log_filename

# session cookie store
cookie_path="./cookie.txt"

echo "Fetching CSRF token" | tee -a $log_filename

#fetch CSRF token
curl -k -c $cookie_path -b $cookie_path -H 'cache-control: no-cache' -H "x-csrf-token: fetch" -sI "$md_url" -H "$encauth" >
./header.txt

echo “Fetched CSRF token" | tee -a $log_filename

while read p; do
    [[ "$p" =~ ^x-csrf-token.*$ ]] && token=$(echo $p | cut -f2 -d:)
done <./header.txt

echo “CSRF token retrieved from header" | tee -a $log_filename

# Schedule Job
post_response=$(curl -i -k -s -c $cookie_path -b $cookie_path -H 'cache-control: no-cache' \
    -X POST --url "$es_url" -H "$encauth" -H  "x-csrf-token: $token" )

echo $post_response | tee -a $log_filename

# Parse response for name and run-count (used in retrieving status)
job_name=$(echo $post_response | sed "s/.*<d:JobName>//g" | sed "s/<\/d:JobName>.*//g")

job_run_count=$(echo $post_response | sed "s/.*<d:JobRunCount>//g" | sed "s/<\/d:JobRunCount>.*//g")

echo "Job name: " $job_name ", job run count: " $job_run_count | tee -a $log_filename

# Prepare request parameters for status GET
stat_f_url=$(echo $stat_url"JobName='"$job_name"'&JobRunCount='"$job_run_count"'")

#check for errors when triggering job in IBP
if echo "$job_name" | grep -q "error" || echo "$job_name" | grep -q "unavailable" || echo "$job_name" | grep -q "no valid server";
then
    echo "Error when triggering Job:" $job_text " (" $job_template_name ")" | tee -a $log_filename
    echo "HTML response from IBP: " | tee -a $log_filename
    echo $job_name | tee -a $log_filename
    exit 1
fi

if [ -z "$job_name" ];then
    echo "Error when triggering Job:" $job_text " (" $job_template_name " )" | tee -a $log_filename
    echo "Possible network or IBP server URL issue.  Check that IBP Job Management URL can be reached:" | tee -a $log_filename
    echo $md_url | tee -a $log_filename
    exit 1
fi

# Loop and print current status of job
n=0
while [ : ]
do
    echo "Job Statust Retrieval:" $job_text " (" $job_template_name ") " | tee -a $log_filename
    stat_get_response=$(curl -k -s -c $cookie_path -b $cookie_path --url "$stat_f_url")
    job_stat=$(echo $stat_get_response | sed "s/.*<d:JobStatus>//g" | sed "s/<\/d:JobStatus>.*//g")
    clear
    tput cup 5 5
    echo "Job Running:" $job_text " (" $job_template_name ") "
    tput cup 6 5
    echo "Status: " $job_stat ", running for: " $n " seconds"
```

```
    if [ "$job_stat" = "F" ]; then
        echo "Execution ID:" $job_name " with run count: " $job_run_count | tee -a $log_filename
        echo "finished with status: " $job_stat " (finished) after " $n " seconds " | tee -a $log_filename
        exit 0
    fi
    if [ "$job_stat" = "A" ]; then
        echo "Execution ID:" $job_name " with run count: " $job_run_count | tee -a $log_filename
        echo "finished with status: " $job_stat " (aborted/failed) after " $n " seconds " | tee -a $log_filename
        echo "Check IBP Application Job logs." | tee -a $log_filename
        exit 1
    fi
    n=$((n+1))
    sleep 60s
    if [ "$n" -gt "$run_time_limit" ]; then
        echo "Job:" $job_text " (" $job_template_name ") was cancelled because it ran for more than " $run_time_limit " seconds."
| tee -a $log_filename
        echo "Check IBP Application Job logs & update run time limit if job needs more time to execute." | tee -a $log_filename
        exit 1
    fi
done
tput cup 7 5
```

ABAP

Please see note [510007 - Setting up SSL on Application Server ABAP](#) for details on settings of SSL on the system where the ABAP program is running.

Make sure that the ssl/ciphersuites and client_ciphersuites profile parameters of the caller system, are TLS 1.2 enabled. For this, follow Section 7 *Recommended Configuration of Available TLS Protocol Versions (required for enabling TLSv1.2)* of the note [510007 - Setting up SSL on Application Server ABAP.](#)

510007 - Setting up SSL on Application Server ABAP

Show Changes

Version 144 from 27.05.2019 in English

Description | Software Components | References | Attributes | Languages

7. **Recommended Configuration of Available TLS Protocol Versions (required for enabling TLSv1.2)**

Over the course of year 2016, a growing number of TLS servers were reconfigured to abort/reject TLSv1.0 handshakes, or they are requring forward secrecy (PFS) cipher suites for access. **The currently recommended settings for TLSv1.2 interoperability are** (requiring at least CommonCryptoLib 8.4.38, recommending at least 8.4.49):

```
ssl/ciphersuites        = 135:PFS:HIGH::EC_P256:EC_HIGH

ssl/client_ciphersuites = 150:PFS:HIGH::EC_P256:EC_HIGH

icm/HTTPS/client_sni_enabled = TRUE

ssl/client_sni_enabled = TRUE

SETENV_16 = SECUDIR=$(DIR_INSTANCE)$(DIR_SEP)sec
SETENV_17 = SAPSSL_CLIENT_CIPHERSUITES=150:PFS:HIGH::EC_P256:EC_HIGH
SETENV_18 = SAPSSL_CLIENT_SNI_ENABLED=TRUE
```

Make sure that the IBP System's Root certificate is uploaded in the ABAP caller system via STRUST transaction, as described in the *Root Certificate* section.

Sample using User Name and Password Authentication for the Inbound Communication User

For the ABAP integration, a destination of type "G – HTTP Connection to external server" has to be created in your local system with transaction SM59.

Use the hostname to the IBP system as target in the technical settings. Leave everything else empty in this tab.

Administration | Technical Settings | Logon _Security | Special Options

Target System Settings

Targ | Serv

Path Prefix

HTTP Proxy Options

Global Configuration

Proxy Host

Proxy Service

Proxy User

Proxy PW Status: is initial

Administration | Technical Settings | Logon _Security | Special Options

Logon Procedure

Logon with User

Do Not Use a User

Basic Authentication

User

PW Status: is initial

Logon with Ticket

Do Not Send Logon Ticket

Send Logon Ticket Without Ref. to a Target System

Send Assertion Ticket for Dedicated Target System

System ID | Client

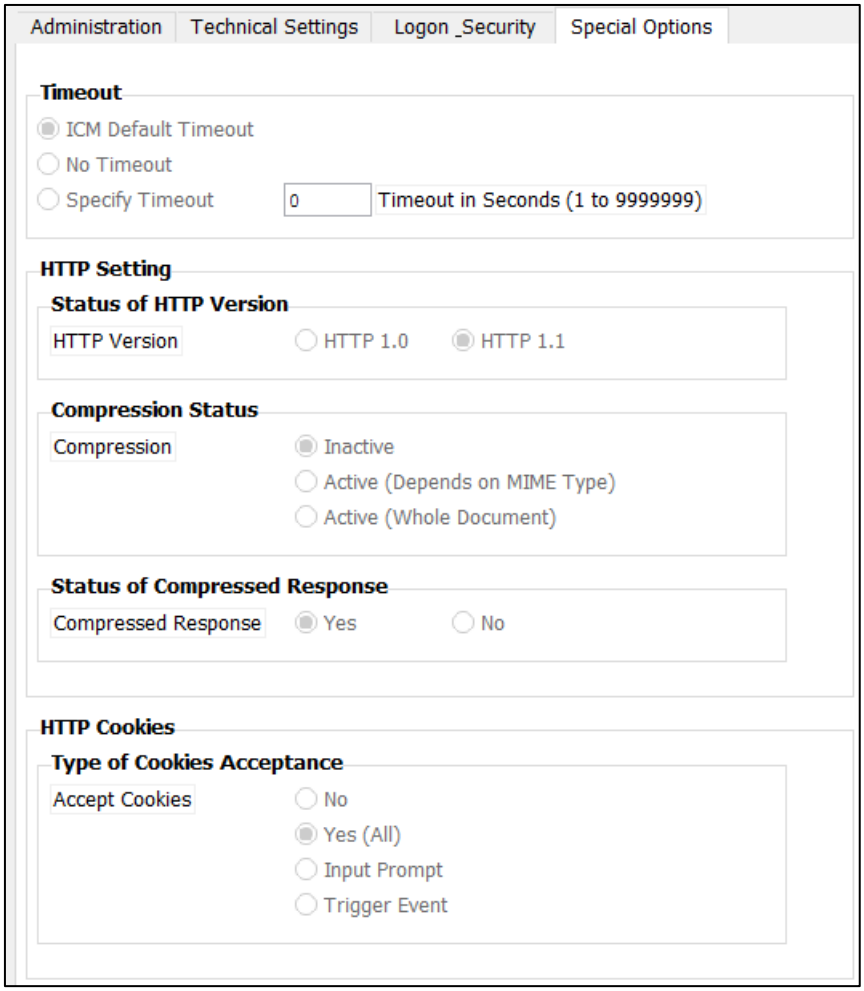
Security Options

Status of Secure Protocol

SSL: Inactive | Active

SSL Certificate: 001_SA 001 SAP Anonymous | Cert. List

Authorization for Destination



Remember that you need to adjust your settings depending on the exact system layout. Above example only works as this is the same network!

```
REPORT z_ibp_schedule LINE-SIZE 1023.

CONSTANTS:
  cv_base_path        TYPE string VALUE '/sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2/',
  cv_template_set_url TYPE string VALUE 'JobTemplateSet?',
  cv_job_schedule_url TYPE string VALUE
    'JobSchedule?JobTemplateName=''<templatename>''&JobText=''<jobtext>''&JobUser=''<jobuser>''',
  cv_job_status_url   TYPE string VALUE
    'JobStatusGet?JobName=''<jobname>''&JobRunCount=''<jobruncount>''',
  cv_replace_template TYPE string VALUE '<templatename>',
  cv_replace_jobtext  TYPE string VALUE '<jobtext>',
  cv_replace_jobuser  TYPE string VALUE '<jobuser>',
  cv_replace_jobname  TYPE string VALUE '<jobname>',
  cv_replace_jobcount TYPE string VALUE '<jobruncount>',
  cv_find_jobname     TYPE string VALUE '<d:JobName>',
  cv_find_jobcount    TYPE string VALUE '<d:JobRunCount>',
  cv_find_status      TYPE string VALUE '<d:JobStatus>',
  cv_replace_hostname TYPE string VALUE '<hostname>',
  cv_find_jtname_s    TYPE string VALUE '<d:JobTemplateName>',
  cv_find_jtname_e    TYPE string VALUE '</d:JobTemplateName>',
  cv_find_jttext_s    TYPE string VALUE '<d:JobTemplateText>',
  cv_find_jttext_e    TYPE string VALUE '</d:JobTemplateText>',
  cv_req_method_head  TYPE string VALUE 'HEAD',
  cv_fd_name_reqwith  TYPE string VALUE 'X-Requested-With',
  cv_fd_value_reqwith TYPE string VALUE 'XMLHttpRequest',
  cv_fd_name_cnttype  TYPE string VALUE 'Content-Type',
```

```
  cv_fd_value_cnt_at  TYPE string VALUE 'application/atom+xml',
  cv_fd_name_dsv      TYPE string VALUE 'DataServiceVersion',
  cv_fd_value_dsv2    TYPE string VALUE '2.0' ,
  cv_fd_name_csrftk   TYPE string VALUE 'X-CSRF-Token',
  cv_fd_value_fetch   TYPE string VALUE 'Fetch',
  cv_fd_name_scokie   TYPE string VALUE 'Set-Cookie',
  cv_fd_name_cookie   TYPE string VALUE 'Cookie',
  cv_encod_utf8       TYPE abap_encod VALUE 'UTF-8',
  cv_maxruns          TYPE i VALUE 1000.

PARAMETERS:
  techuser TYPE string  DEFAULT '<technical user>' LOWER CASE,
  techupwd TYPE string  DEFAULT '<technical user password>' LOWER CASE,
  template TYPE string  DEFAULT '<template>' LOWER CASE,
  execuser TYPE string  DEFAULT '<executing business user>' LOWER CASE,
  jobtext  TYPE string  DEFAULT 'Scheduled via Sample Code in ABAP' LOWER CASE,
  hostdest TYPE rfcdest DEFAULT '<rfc destination>'.


DATA:
  lv_template_list_xml TYPE string,
  lv_cookie            TYPE string,
  lv_csrf              TYPE string,
  lv_template_set_url  TYPE string,
  lv_job_schedule_url  TYPE string,
  lv_job_status_url    TYPE string,
  lv_encoded_jobtext   TYPE string,
  lv_job_schedule_xml  TYPE string,
  lv_index_1           TYPE i,
  lv_index_2           TYPE i,
  lv_jobname           TYPE string,
  lv_jobcount          TYPE string,
  lv_jtname            TYPE string,
  lv_jttext            TYPE string,
  lr_client            TYPE REF TO if_http_client,
  lv_job_status_xml    TYPE string,
  lv_jobstat           TYPE string,
  lv_runs              TYPE i.

CONCATENATE cv_base_path cv_template_set_url INTO lv_template_set_url.

" Get the job template list
PERFORM get_request USING    lv_template_set_url
                             techuser
                             techupwd
                             hostdest
                     CHANGING lv_template_list_xml.

IF find( val = lv_template_list_xml sub = template ) >= 0.
  WRITE: / 'Template found, can be scheduled.'.
ELSE.
  WRITE: / 'Template not found, cannot be scheduled.'.
ENDIF.

" Optional:
" List all the job templates
" In case that the job template name to be scheduled is not known in advance,
" it can be extracted from the lv_template_list_ml
DO.
  lv_index_1 = find( val = lv_template_list_xml sub = cv_find_jtname_s ).
  lv_index_2 = find( val = lv_template_list_xml sub = cv_find_jtname_e ).
  IF lv_index_1 <= 0 OR lv_index_2 <= 0.
    EXIT. " do
  ELSE.
    lv_index_1 = lv_index_1 + strlen( cv_find_jtname_s ).
    lv_jtname = substring( val = lv_template_list_xml off = lv_index_1 len = ( lv_index_2 -
lv_index_1 ) ).

    lv_index_1 = find( val = lv_template_list_xml sub = cv_find_jttext_s ).
    lv_index_2 = find( val = lv_template_list_xml sub = cv_find_jttext_e ).
    IF lv_index_1 <= 0 OR lv_index_2 <= 0.
      EXIT. " do
```

```
    ELSE.
      lv_index_1 = lv_index_1 + strlen( cv_find_jtname_s ).
      lv_jttext = substring( val = lv_template_list_xml off = lv_index_1 len = ( lv_index_2 -
lv_index_1 ) ).
    ENDIF.

    WRITE:  / lv_jtname.
    WRITE AT 40: lv_jttext.
    lv_index_2 = lv_index_2 + strlen( cv_find_jttext_e ).
  ENDIF.
  lv_template_list_xml = substring( val = lv_template_list_xml off = lv_index_2 ).
ENDDO.

" Prepare the URL lv_job_schedule_url, to schedule the template
lv_encoded_jobtext = cl_http_utility=>escape_url( jobtext ).

CONCATENATE cv_base_path cv_job_schedule_url INTO lv_job_schedule_url.
REPLACE cv_replace_template IN lv_job_schedule_url WITH template.
REPLACE cv_replace_jobtext  IN lv_job_schedule_url WITH lv_encoded_jobtext.
REPLACE cv_replace_jobuser  IN lv_job_schedule_url WITH execuser.

" Mandatory:
" Do a head request to pick the csrf token and the cookie
" The csrf token and the cookie are mandatory for the subsequent POST request
PERFORM head_request USING lv_job_schedule_url
                           techuser
                           techupwd
                           hostdest
                     CHANGING lv_cookie
                              lv_csrf
                              lr_client.

"Do the real job template scheduling
PERFORM post_request USING    lv_job_schedule_url
                              techuser
                              techupwd
                              hostdest
                              lv_cookie
                              lv_csrf
                              lr_client
                     CHANGING lv_job_schedule_xml.

" Optional:
" Extract the job name and job count from the lv_job_schedule_ml
" They are the identifiers of the scheduled job
" and can be used to check the job status

IF find( val = lv_job_schedule_xml sub = cv_find_jobname )  >= 0 AND
   find( val = lv_job_schedule_xml sub = cv_find_jobcount ) >= 0.
  lv_index_1 = find( val = lv_job_schedule_xml sub = cv_find_jobname ).
  lv_index_1 = lv_index_1 + strlen( cv_find_jobname ).
  lv_index_2 = find( val = lv_job_schedule_xml sub = '<' off = lv_index_1 ) - lv_index_1.
  lv_jobname = substring( val = lv_job_schedule_xml off = lv_index_1 len = lv_index_2 ).

  lv_index_1  = find( val = lv_job_schedule_xml sub = cv_find_jobcount ).
  lv_index_1  = lv_index_1 + strlen( cv_find_jobcount ).
  lv_index_2  = find( val = lv_job_schedule_xml sub = '<' off = lv_index_1 ) - lv_index_1.
  lv_jobcount = substring( val = lv_job_schedule_xml off = lv_index_1 len = lv_index_2 ).
ENDIF.

IF lv_jobname IS NOT INITIAL AND lv_jobcount IS NOT INITIAL.
  CONCATENATE cv_base_path cv_job_status_url INTO lv_job_status_url.

  REPLACE cv_replace_jobname  IN lv_job_status_url WITH lv_jobname.
  REPLACE cv_replace_jobcount IN lv_job_status_url WITH lv_jobcount.

" Optional:
" Retrieve the status of the scheduled  job polling every 10 seconds
" until the job reaches a final status (finished, aborted)
" or up to cv_maxruns times
" The job status polling is needed if subsequent actions have to be performed
" after the execution of the job finished
```

```
  DO.
    PERFORM get_request USING    lv_job_status_url
                                 techuser
                                 techupwd
                                 hostdest
                       CHANGING lv_job_status_xml.

    IF find( val = lv_job_status_xml sub = cv_find_status )  >= 0.
      lv_index_1 = find( val = lv_job_status_xml sub = cv_find_status ).
      lv_index_1 = lv_index_1 + strlen( cv_find_status ).
      lv_index_2 = find( val = lv_job_status_xml sub = '<' off = lv_index_1 ) - lv_index_1.
      lv_jobstat = substring( val = lv_job_status_xml off = lv_index_1 len = lv_index_2 ).
    ENDIF.

    CASE lv_jobstat.
      WHEN 'F'. " Finished
        WRITE: / 'Finished'.
        EXIT.
      WHEN 'A'. " Aborted / Deleted
        WRITE: / 'Aborted'.
        EXIT.
      WHEN 'R'. " Active (Running)
        WRITE: / 'Running', sy-uzeit.
      WHEN 'P' OR 'S' OR 'Y'. " Scheduled, Released, Ready
        WRITE: / 'Not yet running', sy-uzeit.
      WHEN OTHERS.
        WRITE: / 'Unknown status. Ending report'.
        EXIT.
    ENDCASE.

    WAIT UP TO 10 SECONDS.

    ADD 1 TO lv_runs.

    IF lv_runs > cv_maxruns.
      EXIT.
    ENDIF.

  ENDDO.
ENDIF.

WRITE: / 'End'.

*.
FORM get_request USING    iv_url       TYPE string
                          iv_user      TYPE string
                          iv_password  TYPE string
                          iv_hostdest  TYPE rfcdest
                 CHANGING ev_response  TYPE string.

  DATA:
    lr_client               TYPE REF TO if_http_client,
    lv_success              TYPE boole_d,
    lv_error_code           TYPE sysubrc,
    lv_error_message        TYPE string,
    lv_error_message_class  TYPE arbgb,
    lv_error_message_number TYPE msgnr,
    lv_xstring_get          TYPE xstring.

  CLEAR:
    ev_response.


  WRITE: / 'Instantiating HTTP connection. URL: ', iv_url.

  cl_http_client=>create_by_destination( EXPORTING destination = iv_hostdest
                                         IMPORTING client      = lr_client  ).

  lr_client->request->set_method( if_http_request=>co_request_method_get ).

  cl_http_utility=>set_request_uri( request = lr_client->request
```

```
                                    uri     = iv_url ).


  lr_client->propertytype_accept_cookie = if_http_client=>co_enabled.
  lr_client->propertytype_logon_popup   = if_http_client=>co_disabled.
  lr_client->propertytype_apply_sproxy  = if_http_client=>co_enabled.
  lr_client->propertytype_redirect      = if_http_client=>co_enabled.

  lr_client->authenticate( username = iv_user
                           password = iv_password ).

  WRITE: / 'Sending GET request...'.

  lr_client->send( EXCEPTIONS http_communication_failure = 1
                              http_invalid_state         = 2
                              http_processing_failed     = 3
                              http_invalid_timeout       = 4
                              OTHERS                     = 5 ).
  CASE sy-subrc.
    WHEN 0. WRITE: 'Success'.                      lv_success = 'X'.
    WHEN 1. WRITE: 'HTTP communication failure'.   lv_success = space.
    WHEN 2. WRITE: 'HTTP invalid state'.           lv_success = space.
    WHEN 3. WRITE: 'HTTP processing failed'.       lv_success = space.
    WHEN 4. WRITE: 'HTTP invalid timeout'.         lv_success = space.
    WHEN OTHERS. WRITE: 'HTTP Error'.              lv_success = space.
  ENDCASE.
  IF lv_success IS INITIAL.

    lr_client->get_last_error( IMPORTING code           = lv_error_code
                                         message        = lv_error_message
                                         message_class  = lv_error_message_class
                                         message_number = lv_error_message_number ).

    WRITE: / 'Returncode:', lv_error_code, ', message:', lv_error_message.
  ELSE.
    WRITE: / 'Retrieving response...'.

    lr_client->receive( EXCEPTIONS http_communication_failure = 1
                                   http_invalid_state         = 2
                                   http_processing_failed     = 3
                                   OTHERS                     = 4 ).
    CASE sy-subrc.
      WHEN 0. WRITE: 'Success'.                      lv_success = 'X'.
      WHEN 1. WRITE: 'HTTP communication failure'.   lv_success = space.
      WHEN 2. WRITE: 'HTTP invalid state'.           lv_success = space.
      WHEN 3. WRITE: 'HTTP processing failed'.       lv_success = space.
      WHEN OTHERS. WRITE: 'HTTP Error'.              lv_success = space.
    ENDCASE.

    IF lv_success IS INITIAL.
      lr_client->get_last_error( IMPORTING code           = lv_error_code
                                           message        = lv_error_message
                                           message_class  = lv_error_message_class
                                           message_number = lv_error_message_number ).

      WRITE: / 'Returncode:', lv_error_code, ', message:', lv_error_message.
    ELSE.
      lv_xstring_get = lr_client->response->get_data( ).

      DATA(lr_conv) = cl_abap_conv_in_ce=>create( encoding = cv_encod_utf8 ).

      lr_conv->convert( EXPORTING input = lv_xstring_get
                        IMPORTING data  = ev_response  ).

      WRITE: / ev_response.
    ENDIF.
  ENDIF.

  lr_client->close( ).

ENDFORM.
```

```
*.
FORM head_request USING   iv_url       TYPE string
                          iv_user      TYPE string
                          iv_password  TYPE string
                          iv_hostdest  TYPE rfcdest
                  CHANGING ev_cookie TYPE string
                           ev_csrf   TYPE string
                           er_client TYPE REF TO if_http_client.

  DATA:
    lv_success              TYPE boole_d,
    lv_error_code           TYPE sysubrc,
    lv_error_message        TYPE string,
    lv_error_message_class  TYPE arbgb,
    lv_error_message_number TYPE msgnr,
    lv_xstring_get          TYPE xstring.


  CLEAR:
    ev_cookie,
    ev_csrf.


  WRITE: / 'Instantiating HTTP connection. URL: ', iv_url.

  cl_http_client=>create_by_destination( EXPORTING destination = iv_hostdest
                                         IMPORTING client      = er_client  ).

  er_client->request->set_method( cv_req_method_head ).
  cl_http_utility=>set_request_uri( request = lr_client->request
                                    uri     = iv_url ).

  er_client->propertytype_accept_cookie = if_http_client=>co_enabled.
  er_client->propertytype_logon_popup   = if_http_client=>co_disabled.
  er_client->propertytype_apply_sproxy  = if_http_client=>co_enabled.
  er_client->propertytype_redirect      = if_http_client=>co_enabled.

  er_client->authenticate( username = iv_user
                           password = iv_password ).

  er_client->request->set_header_field( name  = cv_fd_name_reqwith
                                        value = cv_fd_value_reqwith   ).

  er_client->request->set_header_field( name  = cv_fd_name_cnttype
                                        value = cv_fd_value_cnt_at   ).

  er_client->request->set_header_field( name  = cv_fd_name_dsv
                                        value = cv_fd_value_dsv2   ).

  er_client->request->set_header_field( name  = cv_fd_name_csrftk
                                        value = cv_fd_value_fetch       ).

  WRITE: / 'Sending HEAD request...'.

  er_client->send( EXCEPTIONS http_communication_failure = 1
                              http_invalid_state         = 2
                              http_processing_failed     = 3
                              http_invalid_timeout       = 4
                              OTHERS                     = 5 ).
  CASE sy-subrc.
    WHEN 0. WRITE: 'Success'.                     lv_success = abap_true.
    WHEN 1. WRITE: 'HTTP communication failure'.  lv_success = space.
    WHEN 2. WRITE: 'HTTP invalid state'.          lv_success = space.
    WHEN 3. WRITE: 'HTTP processing failed'.      lv_success = space.
    WHEN 4. WRITE: 'HTTP invalid timeout'.        lv_success = space.
    WHEN OTHERS. WRITE: 'HTTP Error'.             lv_success = space.
  ENDCASE.
  IF lv_success IS INITIAL.

    er_client->get_last_error( IMPORTING code           = lv_error_code
                                         message        = lv_error_message
                                         message_class  = lv_error_message_class
```

```
                                        message_number = lv_error_message_number ).

    WRITE: / 'Returncode:', lv_error_code, ', message:', lv_error_message.
  ELSE.
    WRITE: / 'Retrieving response...'.

    er_client->receive( EXCEPTIONS http_communication_failure = 1
                                   http_invalid_state         = 2
                                   http_processing_failed     = 3
                                   OTHERS                     = 4 ).
    CASE sy-subrc.
      WHEN 0. WRITE: 'Success'.                    lv_success = abap_true.
      WHEN 1. WRITE: 'HTTP communication failure'. lv_success = space.
      WHEN 2. WRITE: 'HTTP invalid state'.         lv_success = space.
      WHEN 3. WRITE: 'HTTP processing failed'.     lv_success = space.
      WHEN OTHERS. WRITE: 'HTTP Error'.            lv_success = space.
    ENDCASE.

    IF lv_success IS INITIAL.
      er_client->get_last_error( IMPORTING code           = lv_error_code
                                           message        = lv_error_message
                                           message_class  = lv_error_message_class
                                           message_number = lv_error_message_number ).

      WRITE: / 'Returncode:', lv_error_code, ', message:', lv_error_message.
    ELSE.

      ev_cookie = er_client->response->get_header_field( name = cv_fd_name_scokie ).

      ev_csrf   = er_client->response->get_header_field( name =  cv_fd_name_csrftk ).


      WRITE: / 'Cookie', ev_cookie.
      WRITE: / 'CSRF Token', ev_csrf.

    ENDIF.
  ENDIF.

ENDFORM.


*.
FORM post_request USING    iv_url       TYPE string
                           iv_user      TYPE string
                           iv_password  TYPE string
                           iv_hostdest  TYPE rfcdest
                           iv_cookie    TYPE string
                           iv_csrf      TYPE string
                           ir_client    TYPE REF TO if_http_client
                  CHANGING ev_response  TYPE string.

  DATA:
    lr_client               TYPE REF TO if_http_client,
    lv_success              TYPE boole_d,
    lv_error_code           TYPE sysubrc,
    lv_error_message        TYPE string,
    lv_error_message_class  TYPE arbgb,
    lv_error_message_number TYPE msgnr,
    lv_xstring_post         TYPE xstring.

  CLEAR:
    ev_response.

  WRITE: / 'Instantiating HTTP connection. URL: ', iv_url.

  ir_client->request->set_method( if_http_request=>co_request_method_post ).
  cl_http_utility=>set_request_uri( request = ir_client->request
                                    uri     = iv_url ).

  ir_client->propertytype_accept_cookie = if_http_client=>co_enabled.
  ir_client->propertytype_logon_popup   = if_http_client=>co_disabled.
  ir_client->propertytype_apply_sproxy  = if_http_client=>co_enabled.
```

```
  ir_client->propertytype_redirect      = if_http_client=>co_enabled.

  ir_client->authenticate( username = iv_user
                           password = iv_password ).

  ir_client->request->set_header_field( name  = cv_fd_name_cookie
                                        value = iv_cookie  ).

  ir_client->request->set_header_field( name  = cv_fd_name_csrftk
                                        value = iv_csrf        ).
  WRITE: / 'Sending POST request...'.

  ir_client->send( EXCEPTIONS http_communication_failure = 1
                              http_invalid_state         = 2
                              http_processing_failed     = 3
                              http_invalid_timeout       = 4
                              OTHERS                     = 5 ).
  CASE sy-subrc.
    WHEN 0. WRITE: 'Success'.                     lv_success = abap_true.
    WHEN 1. WRITE: 'HTTP communication failure'.  lv_success = space.
    WHEN 2. WRITE: 'HTTP invalid state'.          lv_success = space.
    WHEN 3. WRITE: 'HTTP processing failed'.      lv_success = space.
    WHEN 4. WRITE: 'HTTP invalid timeout'.        lv_success = space.
    WHEN OTHERS. WRITE: 'HTTP Error'.             lv_success = space.
  ENDCASE.
  IF lv_success IS INITIAL.

    ir_client->get_last_error( IMPORTING code           = lv_error_code
                                         message        = lv_error_message
                                         message_class  = lv_error_message_class
                                         message_number = lv_error_message_number ).

    WRITE: / 'Returncode:', lv_error_code, ', message:', lv_error_message.
  ELSE.
    WRITE: / 'Retrieving response...'.

    ir_client->receive( EXCEPTIONS http_communication_failure = 1
                                   http_invalid_state         = 2
                                   http_processing_failed     = 3
                                   OTHERS                     = 4 ).
    CASE sy-subrc.
      WHEN 0. WRITE: 'Success'.                     lv_success = abap_true.
      WHEN 1. WRITE: 'HTTP communication failure'. lv_success = space.
      WHEN 2. WRITE: 'HTTP invalid state'.          lv_success = space.
      WHEN 3. WRITE: 'HTTP processing failed'.      lv_success = space.
      WHEN OTHERS. WRITE: 'HTTP Error'.             lv_success = space.
    ENDCASE.

    IF lv_success IS INITIAL.
      ir_client->get_last_error( IMPORTING code           = lv_error_code
                                           message        = lv_error_message
                                           message_class  = lv_error_message_class
                                           message_number = lv_error_message_number ).

      WRITE: / 'Returncode:', lv_error_code, ', message:', lv_error_message.
    ELSE.

      lv_xstring_post = ir_client->response->get_data( ).

      DATA(lr_conv) = cl_abap_conv_in_ce=>create( encoding = cv_encod_utf8 ).

      lr_conv->convert( EXPORTING input = lv_xstring_post
                        IMPORTING data  = ev_response  ).

      WRITE: / ev_response.

    ENDIF.
  ENDIF.

ENDFORM.
```

ABAP code sample details:

GET Request

GET_REQUEST – FORM

```
FORM get_request USING    iv_url       TYPE string
                          iv_user      TYPE string
                          iv_password  TYPE string
                          iv_hostdest  TYPE rfcdest
                 CHANGING ev_response  TYPE string.
```

This form can be used to perform GET HTTP requests.

Input:

- **iv_url** – the URL of the GET request.
  Example: iv_url = /sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2/JobTemplateSet?
  For the GET request to fetch the list of IBP job templates
- **iv_user** – the IBP Communication User defined in the Preparation section, example: JOBSCHEDULERUSER
- **iv_password** – the password of the Communication User
- **iv_hostdest** – the destination of type "G – HTTP Connection to external server" (for the IBP system) defined in the system where the ABAP report is executed

Output:

- **ev_response** – the XML response of the GET request in string format

  Example: For the GET request to retrieve the list of job templates, the ev_response contains for each job template, following properties


```
<m:properties xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">

<d:JobTemplateName>ZHDVKOF2BIAPONJPTEHJC26PCR4</d:JobTemplateName>
<d:JobTemplateVersion>0</d:JobTemplateVersion>
<d:JobTemplateStepCount>1</d:JobTemplateStepCount>
<d:JobPeriodicGranularity/>
<d:JobReportName>/IBP/RHCI_DI</d:JobReportName>
<d:JobUserName/>
<d:JobPeriodicValue>000</d:JobPeriodicValue>
<d:JobTemplateText>EH_HCI</d:JobTemplateText>
<d:CreationDateTime>2016-10-21T14:17:12.7827710</d:CreationDateTime>
<d:CreationUserName>CB8980000030</d:CreationUserName>
<d:LastChangeDateTime>2017-09-18T13:36:24.3154350</d:LastChangeDateTime>
<d:LastChangeUserName>CB8980000030</d:LastChangeUserName>
<d:SupportsTestModeInd>false</d:SupportsTestModeInd>

</m:properties>
```


Following requests, described in the document, can be performed using the GET_REQUEST Form:

- OData Call to List and Find the Job Template

- OData Call to Check the Status of a Job
- OData Call to Get Extended Info for Jobs
- OData Call to Get List of Jobs

POST Request

The OData Call to Schedule a Job and the OData Call to Cancel a Job are POST requests.

HEAD_REQUEST Form

Every POST request needs to pass a CSRF Token and a cookie. The call to the HEAD_REQUEST fetches the csrf-token and the cookie which will be used in subsequent POST request.

```
FORM head_request USING   iv_url       TYPE string
                          iv_user      TYPE string
                          iv_password  TYPE string
                          iv_hostdest  TYPE rfcdest
                  CHANGING ev_cookie TYPE string
                           ev_csrf   TYPE string
                           er_client TYPE REF TO if_http_client.
```

Input:

- **iv_url** – the URL of the POST request
  Example: iv_url = /sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2/JobSchedule?JobTemplateName='Z7ILD5Y5ARYPNRMFW7LYMT557IA'&JobText='Scheduled%20via%20Sample%20Code%20in%20ABAP'&JobUser='CB8980000030'
- **iv_user** – the IBP Communication User defined in the Preparation section, example: JOBSCHEDULERUSER
- **iv_password** – the password of the Communication User
- **iv_hostdest** – the destination of type "G – HTTP Connection to external server" (for the IBP system) defined in the system where the ABAP report is executed

Output:

- **ev_cookie** – the cookie returned by the HEAD request to be used in the subsequent POST request
- **ev_csrf** – csrf-token returned by the HEAD request to be used in the subsequent POST request
- **er_client** – the HTTP client to be used in the subsequent POST request

POST_REQUEST Form

```
FORM post_request USING    iv_url       TYPE string
                           iv_user      TYPE string
                           iv_password  TYPE string
                           iv_hostdest  TYPE rfcdest
                           iv_cookie    TYPE string
                           iv_csrf      TYPE string
                           ir_client    TYPE REF TO if_http_client
                  CHANGING ev_response  TYPE string.
```

Input:

- **iv_url** – the URL of the POST request
  Example: To schedule a Job Template iv_url = /sap/opu/odata/sap/BC_EXT_APPJOB_MANAGEMENT;v=2/JobSchedule?JobTemplateName='Z7ILD5Y5ARYPNRMFW7LYMT557IA'&JobText='Scheduled%20via%20Sample%20Code%20in%20ABAP'&JobUser='CB8980000030'
- **iv_user** – the IBP Communication User defined in the Preparation section, example: JOBSCHEDULERUSER
- **iv_password** – the password of the Communication User
- **iv_hostdest** – the destination of type "G – HTTP Connection to external server" (for the IBP system) defined in the system where the ABAP report is executed
- **iv_cookie** – the cookie returned by the HEAD request to be used in the subsequent POST request
- **iv_csrf** – csrf-token returned by the HEAD request to be used in the subsequent POST request
- **ir_client** – the HTTP client returned by the HEAD request

Output:

- **ev_response** – the XML response of the POST request in string format

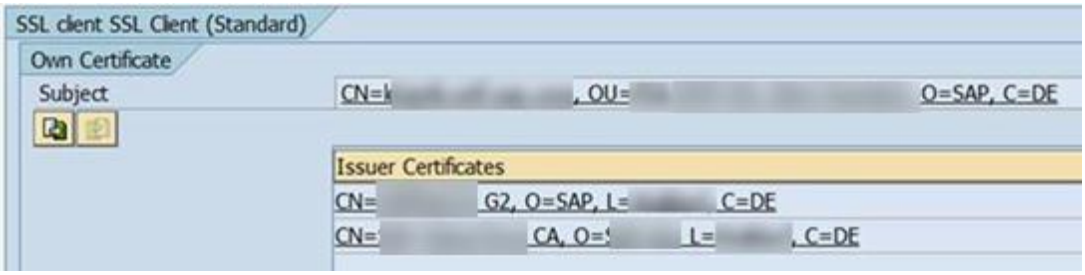
  Example: For the POST request to schedule a job, the ev_response contains the unique identifiers of the scheduled job: Job Name and Run Count. These identifiers can be used in subsequent GET requests to check the status of the job


```
<m:properties>
<d:JobName>FA163EE3A08E1ED9879ED50BD092B7F8</d:JobName>
<d:JobRunCount>xJAyMLTy</d:JobRunCount>
<d:JobStatus/>
<d:ReturnCode>0</d:ReturnCode>
</m:properties>
```


Sample using Certificate based Authentication for the Inbound Communication User

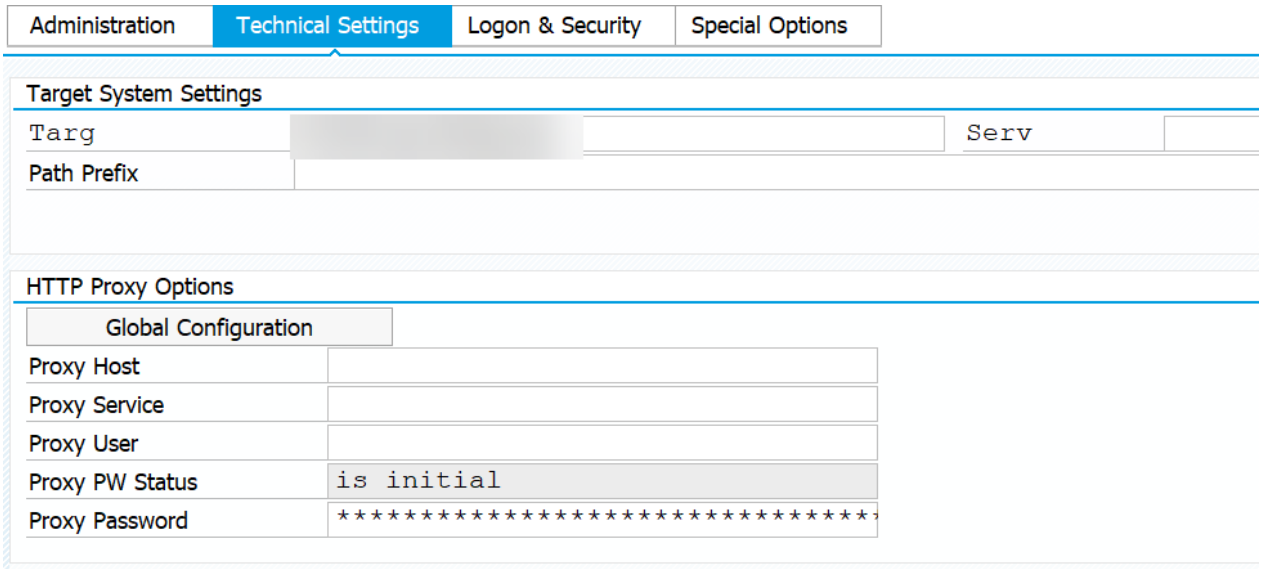
You need to perform the following steps to set up the certificate-based communication between the ABAP caller system and the IBP system:

1. In the ABAP caller system, using transaction STRUST, export the SSL client standard certificate (Note: the certificate does not include any private keys!). This is the certificate which is presented to the receiving party (IBP system).
   The certificate has to be signed by an appropriate certification authority (CA). CAs accepted for inbound integration with SAP IBP Cloud landscape are listed in Note 2607432.



2. Create a a destination of type “G – HTTP Connection to external server” has to be created in your local ABAP system with transaction SM59.

   Use the hostname to the IBP system (myXYZ-api.scmibp.ondemand.com) as target in the technical settings. Leave everything else empty in this tab.



   On the Logon&Security tab select *Do not Use a User* and under Security Options Set SSL - *Active* and the SSL Certificate – *DFAULT SSL Client (Standard).*

   With this configuration, the RFC destination will present the “Own Certificate” from the trust store (the content is managed by STRUST).

RFC Destination

Connection Type G HTTP Connection to External Server Des

Description

Description 1

Description 2

Description 3

Administration | Technical Settings | Logon & Security | Special Options

Logon Procedure

Logon with User

- (•) Do Not Use a User
- ( ) Basic Authentication

User

PW Status is initial

Logon with Ticket

- (•) Do Not Send Logon Ticket
- ( ) Send Logon Ticket Without Ref. to a Target System
- ( ) Send Assertion Ticket for Dedicated Target System

System ID Client

Anmeldung mit MQTT

Benutzer

PW Status is initial

Passwort **********

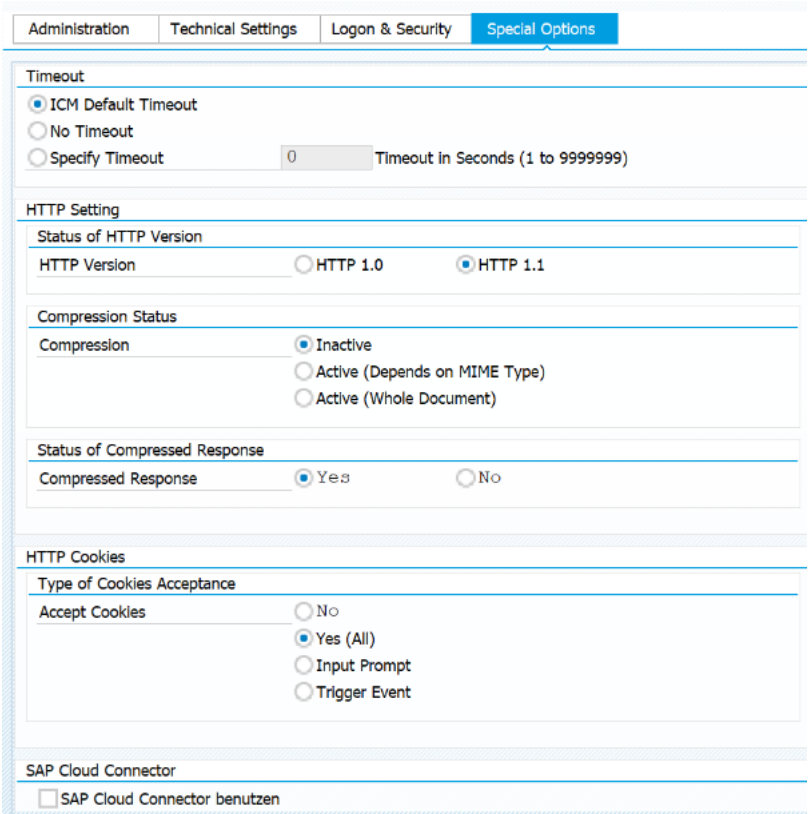
Security Options

Status of Secure Protocol

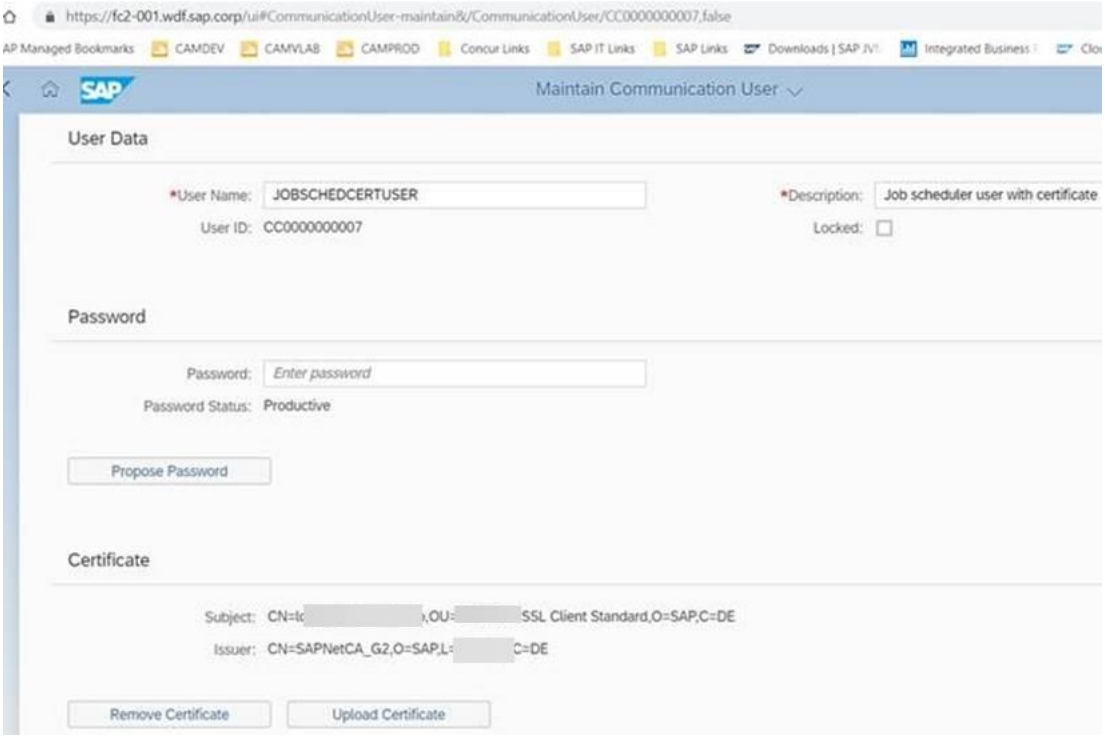
SSL ( ) Inactive (•) Active

SSL Certificate DFAULT SSL Client (Standard) Cert. List

Authorization for Destination



3. In the IBP system create a communication system and inbound communication user which uses the Certificate based authentication. For this user upload the previously exported (in Step 1) client certificate and enable certificate-based authentication.



Also create a communication arrangement for scenario SAP_COM_0064 to use the new communication system and user.

4. Execute the same ABAP report from the Sample using User Name and Password Authentication for the Inbound Communication User, using the new RFC destination and without entering a username/password.

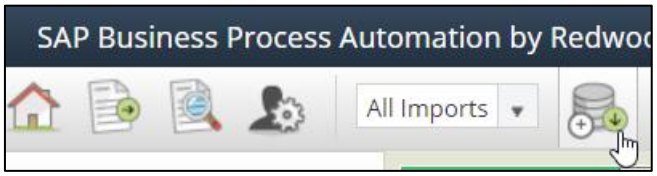
TECHUSER	
TECHUPWD	
TEMPLATE	Z
EXECUSER	CB
JOBTEXT	Scheduled via Sample Code in ABAP
HOSTDEST	

Usage of Redwood BPA

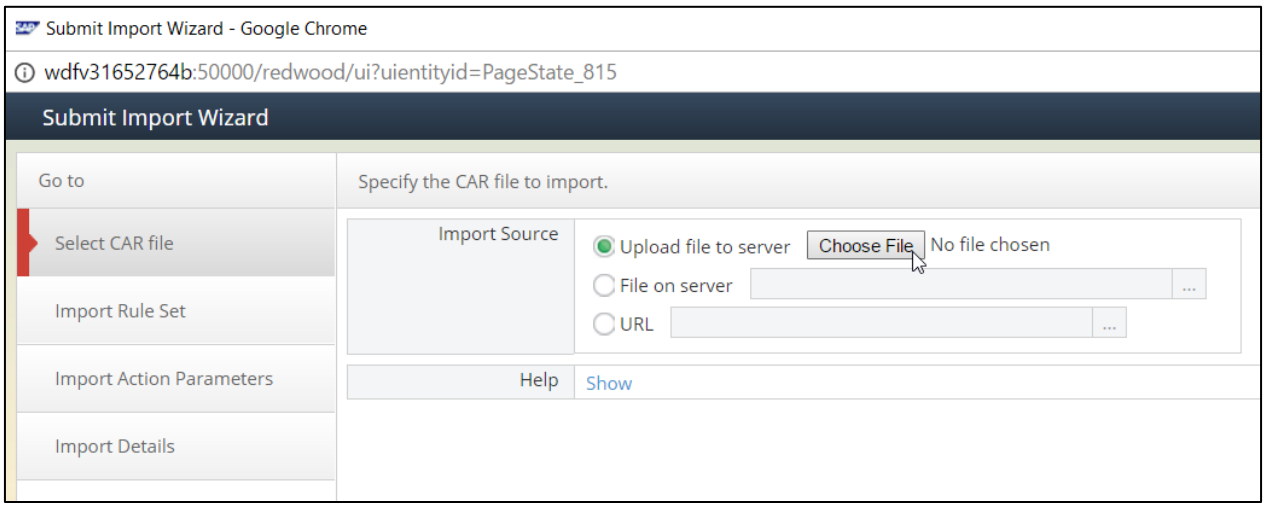
Prerequisite

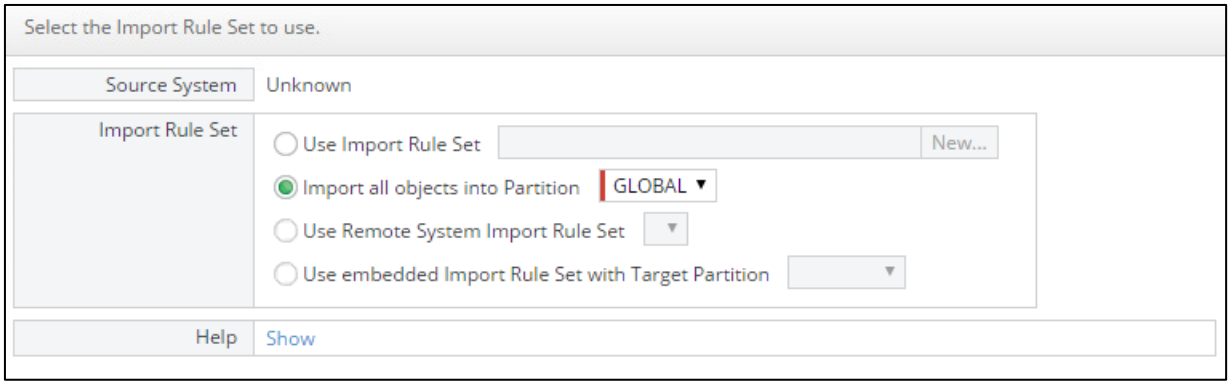
You need Redwood BPA 9.1 minimum, plus the correct car files which provide the connector / features for scheduling IBP jobs.

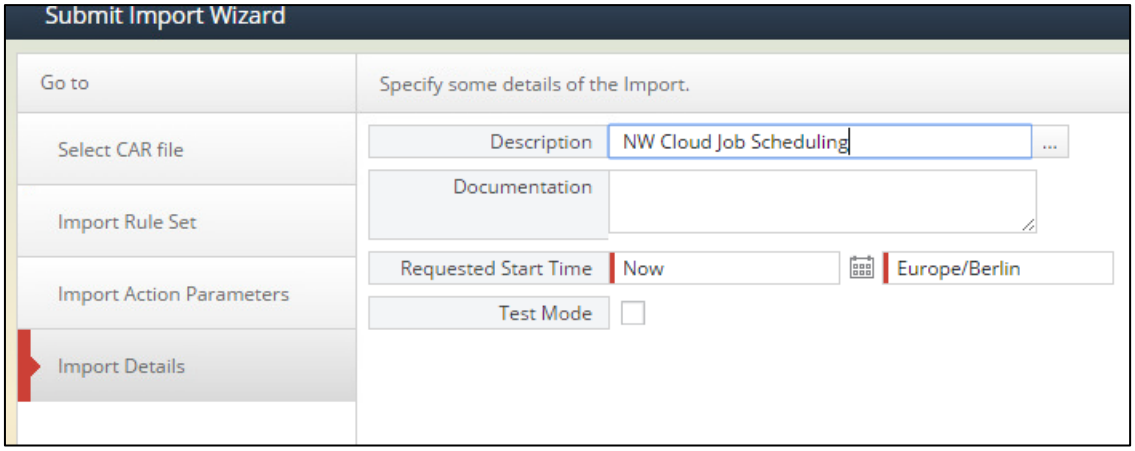
Then install the external scheduling feature for IBP via Promotion > Imports and click "Submit Import Process…"

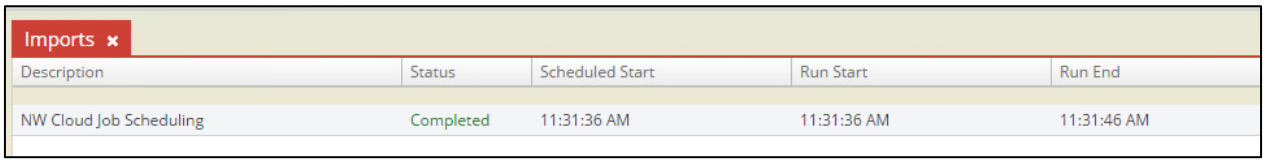


Then click on "Choose File" and select the CAR-file for the external scheduling feature



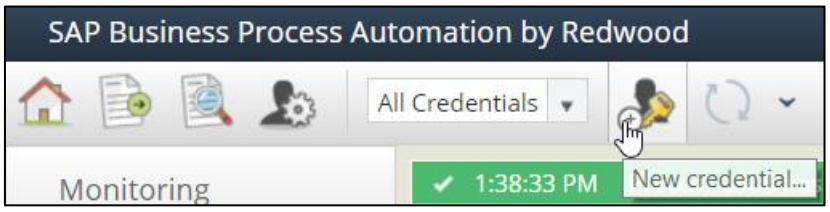


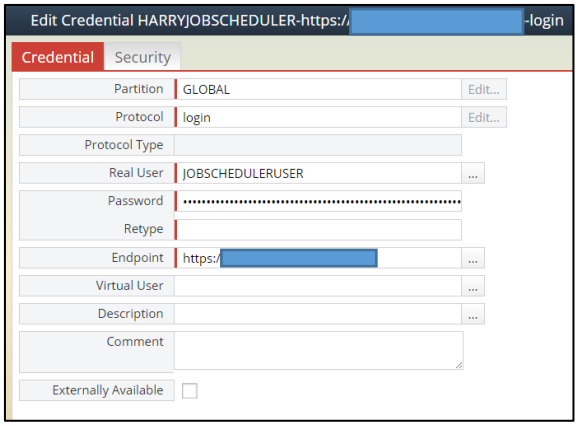




Setup

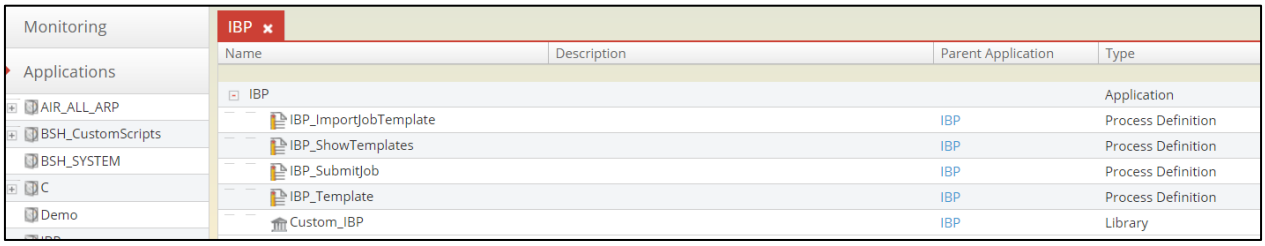
Create an entry in the credentials via Security > Credentials via the icon





Put in the user & password created in the IBP systems Fiori App Communication System (or Communication User) as shown above.

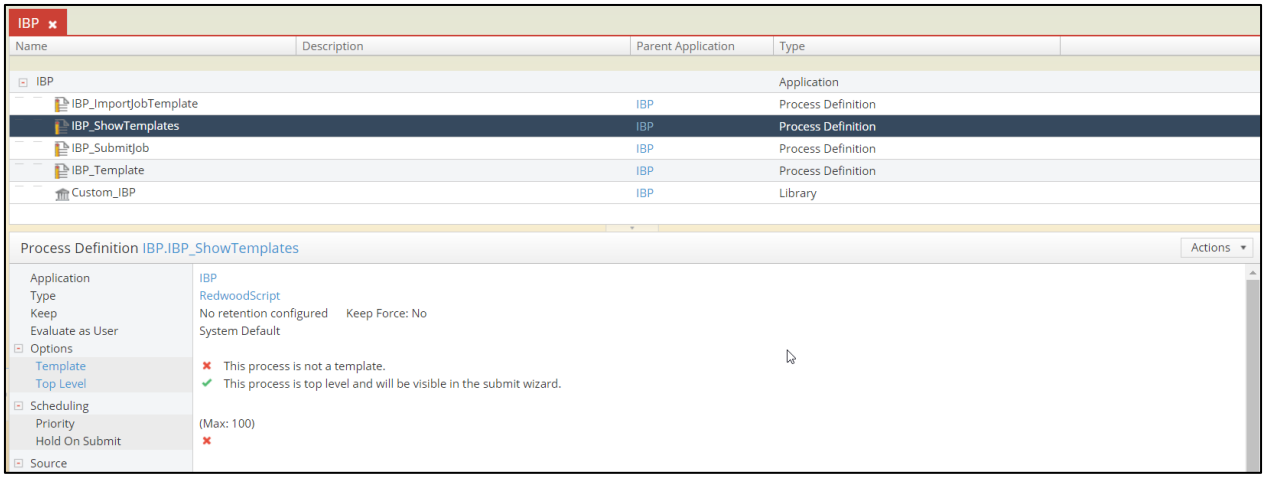
Make sure you have application IBP and library IBP available in Applications > IBP:



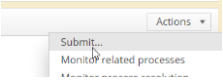
Simple Test

As a first test, we simply retrieve the list of job templates available for scheduling.

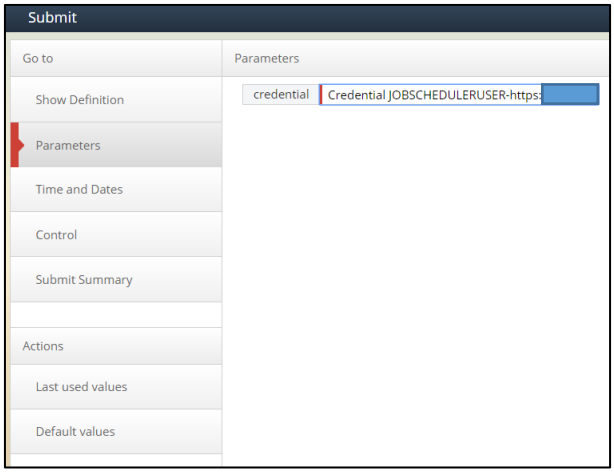
In the IBP application, click on Applications > IBP > IBP_ShowTemplates.



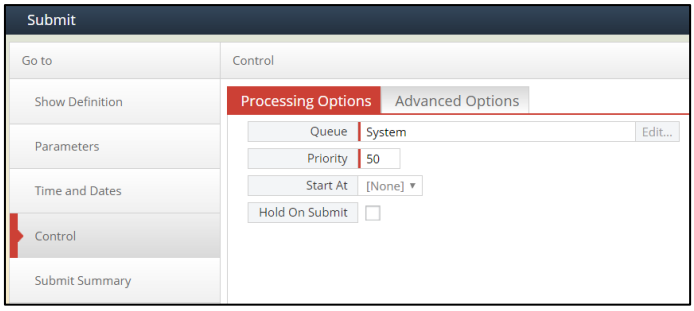
Then choose Actions > Submit.



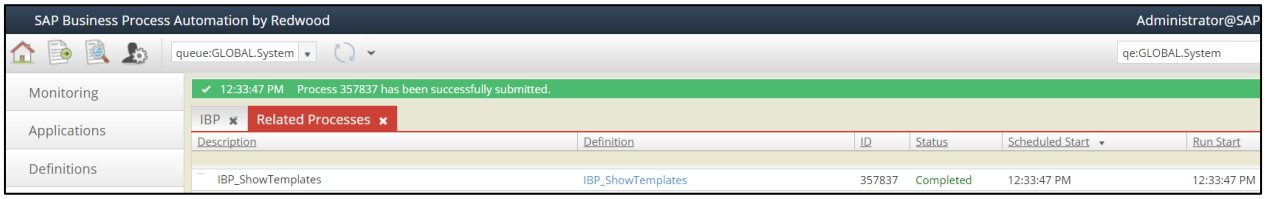
In the popup-window choose the correct credentials.



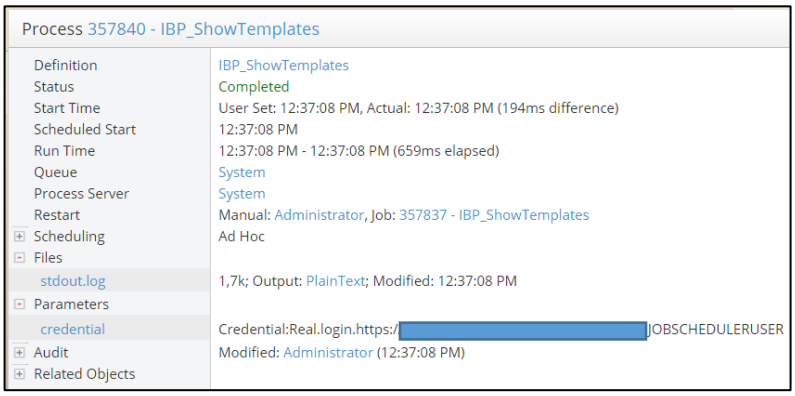
Then choose Now in Time and Dates. Choose System as queue and the default priority and click Submit.



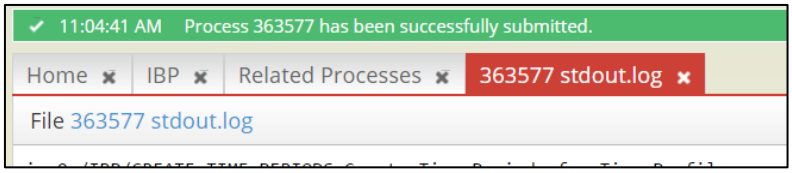
Then see the process status in the monitor, which shows up automatically:



Click on the description, and the lower part of the screen displays something like this:



Click on the stderr.log, and you will see the output as follows:



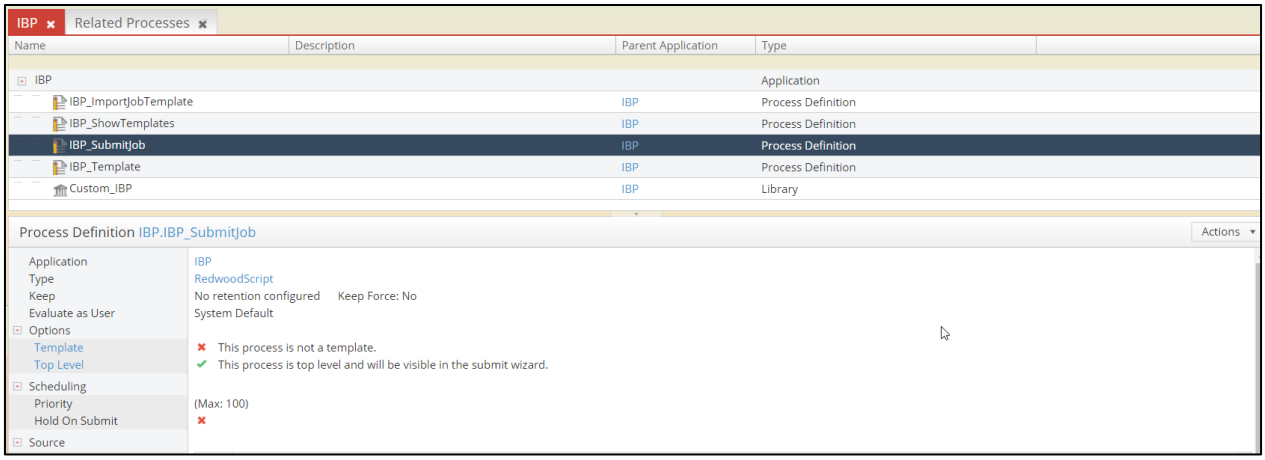
…

```
i: 47 ZHDVKOF2BIAPONJPTEHJC26PCR4 EH_HCI
i: 48 ZHDVKOF2BIAPONP6WRISFKGBHII MANDY_1_STEP
i: 49 ZHDVKOF2BIAPONLGPKBLE4ITVNY DS_Run_JK1
i: 50 ZHDVKOF2BIAPONLXGJEEQBT3KVU FCst
i: 51 ZHDVKOF2BIAPNNPWPA72DZRT2WQ HarryExternalSchedulingNoChain
i: 52 ZHDVKOF2BIAPNNOHZ6GCODPZ5BI IO Group Operator
i: 53 ZHDVKOF2BIAPONLXGV6WGMANMJI AATEST
i: 54 ZHDVKOF2BIAPOLON5H3XYG3FMHU Gustav
i: 55 ZHDVKOF2BIAPNNLOUFPSIXHP3JI test1
i: 56 ZHDVKOF2BIAPNNLOUFXJHAAH3JI New Job Template
i: 57 ZHDVKOF2BIAPNNKETCZUQHVRSHQ DELETE_LOGS_FOR_JOHN_KANG
i: 58 ZHDVKOF2BIAPONJPTD3U3FCNCRQ EH_DS6A_COPY
i: 59 ZHDVKOF2BIAPNNLOUF722RLH3KI purge
i: 60 ZHDVKOF2BIAPNNLOUGKYD5ER3KI New Job Template 01
i: 61 ZHDVKOF2BIAPOLO7UMEJSFAD5TU test_MBL
```

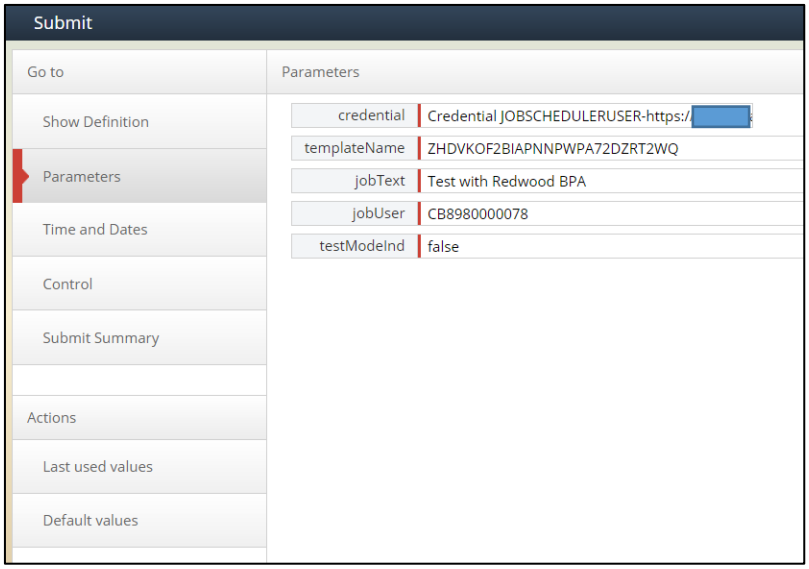
Here you can find the value later used as templateName – it's the value after the sequence number and before the template title.

Schedule & Wait for Finished

In the IBP application choose IBP_SubmitJob.



Then click Actions > Submit and provide the parameters for the scheduling



Remember to use the correct job user – with this user, the job is scheduled and, for example, the IBP visibility filter settings are take from this user. Also, this is the user ID as seen in the Fiori App Maintain Business Users.

The template name is usually a GUID-like ID, the non-GUID-like template names are the SAP delivered templates, which usually have empty parameter values.

Usage of SAP Solution Manager

SAP Solution Manager can be used to schedule and monitor SAP IBP jobs. Please refer to the public documentation here:

https://support.sap.com/content/dam/support/en_us/library/ssp/sap-solution-manager/container/application-operations/JobandBIMonitoring/set-up-and-consumption-of-ibp-appl-job-mon-on-solman-72.pdf

Using POSTMAN to Access the External Scheduler

In this tutorial, we will show you how to use the POSTMAN tool to access the External Scheduler API. POSTMAN is a free application designed to interact with web APIs via HTTP verbs like GET and POST.

Once you have taken care of the prerequisites, the first step to interacting with the External Scheduler API is to fetch the list of job templates. You must select one template ID from among the provided list. Second, you can execute a new job based on the selected template. Finally, you can monitor the status of the new job—to know when it stops executing.

Prerequisites

To get started you need to perform the following steps:

- Download and install POSTMAN.
- Follow the steps in section "Preparation". Set up your communication user to use a username and password. Do not worry about certificates. Note down the communication arrangement's *Service URL*. We'll refer to this as *<SERVICE_URL>* in examples below.

How to Fetch the List of Job Templates

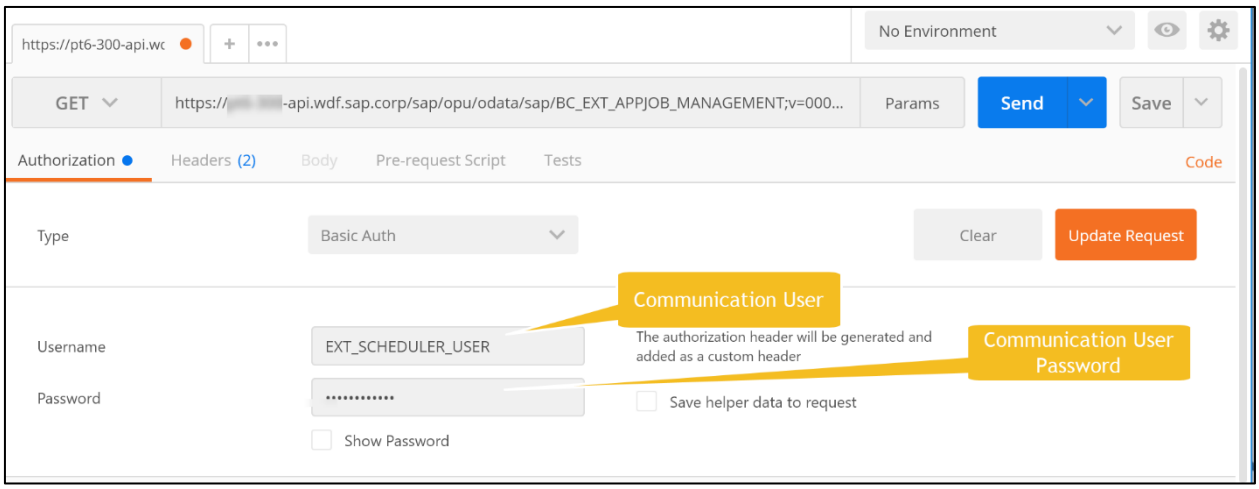
The first step to interacting with the External Scheduler API is to send an HTTP GET request to fetch the list of Job Templates, so that you can obtain a template ID. Follow these steps:

1. In POSTMAN, select the HTTP method **GET**.
2. Enter the URL:

   `<SERVICE_URL>/JobTemplateSet`

   a. Replace `<SERVICE_URL>` with the communication arrangement's *Service URL*.

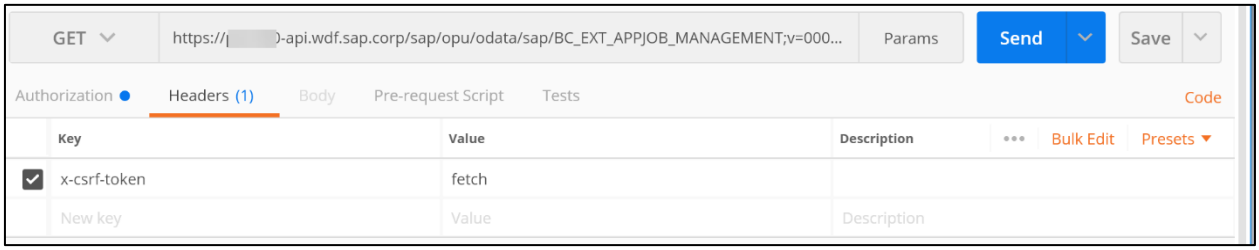
3. In the **Authorization** tab, select type **Basic Auth**.
4. Enter the *username* and *password* for your communication user.



5. In the **Headers** tab, input the key `x-csrf-token` and the value `fetch`

   The Cross-Site Request Forgery (CSRF) token is a security token that must be included in any request that makes a change to the server. We will fetch the token during the current read-only GET request, and then use the token during subsequent modification requests.

   Once you have fetched a CSRF token, you don't need to fetch the token again during the current session. Instead, replace the text `fetch` with the token value you will receive in the server response.



6. Click the **Send** button.

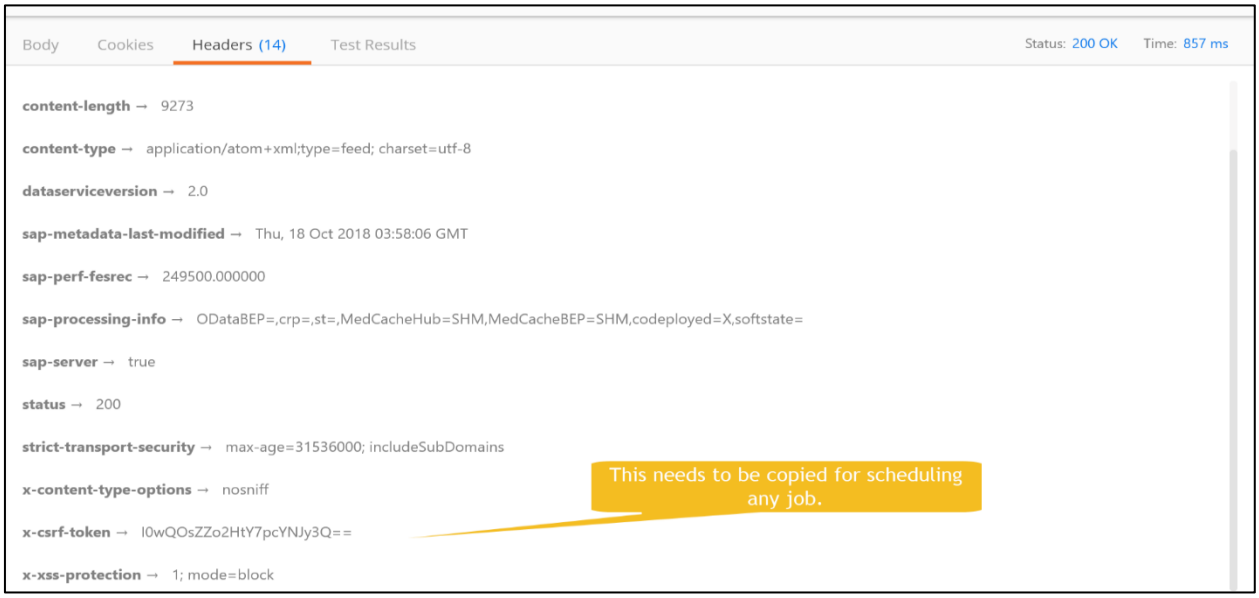
   POSTMAN will send the request to the server. If all goes well then you should receive a response. The screenshot below gives a partial view of the response's **Body** tab:

7. Scroll through the response body and find the ID for the job template that you wish to run.

   Each job template's ID is labelled with the text **JobTemplateName**. For example, in the screenshot above, the job template ID is: **Z7ILD5WNNRAPORG5WWHFQFM5O6I**

8. In the response's **Headers** tab, find the **x-csrf-token** field and take note of the value.

   In the example below, the token's value is: **I0wQOsZZo2HtY7pcYNJy3Q==**



How to Schedule a Job

Once you have a template ID in hand, you can send a request to the External Scheduler API to schedule a job based on the given template. The job will be scheduled for immediate execution. The External Scheduler API is not able to schedule a job that runs at regular time intervals.

1. In POSTMAN, select the HTTP method **POST**.
2. Enter the URL:

   `<SERVICE_URL>/JobSchedule?JobTemplateName='<TEMPLATE_ID>'&JobText='<JOB_NAME>'&JobUser='<JOB_USER>'`

   a. Replace `<SERVICE_URL>` with the communication arrangement's *Service URL*.
   b. Replace `<TEMPLATE_ID>` with the job template ID that you selected. For example: `Z7ILD5WNNRAPORG5WWHFQFM5O6I`
   c. Replace `<JOB_NAME>` with whatever name you want to give to the job.
   d. Replace `<JOB_USER>` with the business user ID of a user that has the privilege to schedule the targeted job. The business user ID can be found in the **Maintain Business Users** Fiori app (section **Identity and Access Management**). For example: `CB8980000046`

3. In the **Authorization** tab, select type **Basic Auth**.
4. Enter the *username* and *password* for your communication user.

5. In the **Headers** tab, enter the key `x-csrf-token` and the value that you obtained from the response to a CSRF fetch request (see section "How to Fetch the List of Job Templates").

6. Click the **Send** button to send the **POST** request to schedule the job.

   If all goes well then you should receive a response.

7. In the response's **Body** tab, you will see the data the server sent back to you. The job's unique ID is split into two parts: the **JobName** and **JobRunCount**. Take note of both IDs.

   You will need both IDs to fetch the job's status or cancel the job.

   In the example above, **JobName** is equal to **FA163ED9AD881EE8BCD1AA115CE53813** and **JobRunCount** is equal to **EFfeXAkD**.

How to Fetch a Job's Status

Given that you have a job's two IDs in hand (JobName and JobRunCount), you can send a request to the External Scheduler API to learn the job's status. This way, you can know if the job has finished executing and if it succeeded or failed.

1. In POSTMAN, select the HTTP method **GET**.
2. Enter the URL:

   ```
   <SERVICE_URL>/JobStatusGet?JobName='<JOB_NAME>'&JobRunCount='<JOB_RUN_COUNT>'
   ```

   a. Replace `<SERVICE_URL>` with the communication arrangement's *Service URL*.
   b. Replace `<JOB_NAME>` with the JobName value that you obtained when you scheduled the job.
   c. Replace `<JOB_RUN_COUNT>` with the JobRunCount value that you obtained when you scheduled the job.

3. In the **Authorization** tab, select type **Basic Auth**.
4. Enter the *username* and *password* for your communication user.
5. Click the **Send** button to send the **GET** request to obtain the job status.

   You should receive a server response.

6. In the response's **Body** tab, inside the JobStatus label, you will find the job's status letter.

   The letter **R** means Running (In Process) and **F** means Finished. The full list of status letters can be found in the section "OData Call to Check the Status of a Job".

Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

Accessibility

The information contained in the SAP documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of willful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: http://help.sap.com/disclaimer).

POSTMAN Screenshots

Copyright