PUBLIC
2021-09-23

SAP Job Scheduling Service

THE BEST RUN SAP

Content

1 What Is SAP Job Scheduling Service

Define and manage one-time and recurring jobs or Cloud Foundry tasks.

SAP Job Scheduling service allows you to define and manage jobs that run once or on a recurring schedule. Use this runtime-agnostic service to schedule action endpoints in your application or long-running processes using Cloud Foundry tasks. Use REST APIs to schedule jobs, including long-running jobs asynchronously, and create multiple schedule formats for simple and complex recurring schedules. Manage jobs and tasks and manage schedules with a web-based user interface.

Environment

This service runs in the Cloud Foundry environment.

Features

Use flexible schedule formats	Flexibly choose between cron and human-readable date formats for your schedules.
Get secure access	Run jobs on behalf of an application by using an OAuth authentication mechanism.
Schedule synchronously or asynchronously	Define and manage jobs synchronously or asynchronously as well as run long-running CF tasks asynchronously.
Optimize your resources usage	Run Cloud Foundry tasks asynchronously and thereby optimize your resources usage.
App and scheduling logic seperated	USE REST APIs to perform CRUD operations on jobs and schedules and to retrieve run logs of executed schedules.
Dashboard for managing jobs and tasks	Use the web-based user interface to create, edit, or delete jobs and tasks for a service instance. Create, manage, and monitor schedules for your jobs and tasks. Check the logs to see if a schedule was executed successfully.
Supports Node-js client libraries	Convenient client library to execute CRUD operations, including create job, from your Node.js application.

Related Information

Initial Setup [page 20]
Getting Started [page 22]
Guided Answers
SAP Job Scheduling Service REST APIs [page 26]
Cloud Foundry Tasks
Secure Access [page 75]
Schedule Formats [page 11]
Asynchronous Mode [page 8]

2 What's New for SAP Job Scheduling Service

2021

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Job Scheduling Service	Extension Suite - Development Efficiency	Cloud Foundry	Support for custom identity providers with platform users	The SAP Job Scheduling service supports the use of custom identity providers with platform users. For more information, see Manage Jobs with Service Dashboard [page 64] and Supported Tools and Services When Using Custom Identity Providers for Platform Users[Feature Set A].	New	2021-05-20
Job Scheduling Service	Extension Suite - Development Efficiency	Cloud Foundry	CF tasks are stopped according to default asynchronous execution timeout	By default, jobs executed in an asynchronous mode are stopped if the application doesn't provide a response within 30 minutes. This now applies also for CF tasks, that means a task running longer than 30 minutes is stopped. You can change this setting for the selected service instance on the *Configurations* page in the service dashboard. For more information, see Manage Jobs with Service Dashboard [page 64].	New	2021-02-25
Job Scheduling Service	Extension Suite - Development Efficiency	Cloud Foundry	Binding level secrets	New clientsecrets are created after you bind an application to the SAP Job Scheduling service. You can use this to rotate clientsecrets by unbinding and then binding the SAP Job Scheduling service instance again to the application. For more information, see *Binding Level Secrets* in SAP Job Scheduling Service REST APIs [page 26].	New	2020-12-17

3 Concepts

SAP Job Scheduling service offers you flexible scheduling capabilities for action endpoints in your application or long-running processes using Cloud Foundry tasks.

Job A collection of schedules with an action endpoint

A job is configured with a URL, which means that your job invokes this URL, for example, every night. A job can also be configured with a Cloud Foundry task.

Synchronous requests are used if the service calls the action endpoint of the application and the application logic is executed in a short span of time. Jobs can be executed in an asynchronous (or batch) mode for job runs with a large span of time.

A service plan within SAP Job Scheduling service considers the number of schedules within a job.

Action Endpoint An HTTP or REST endpoint which is exposed by the application that is bound to the SAP Job Scheduling service and invoked by the service to perform the defined action.

Cloud Foundry Task An app or script whose code is included as part of a deployed app, but runs independently in its own container and is designed to use minimal resources.

Schedule A one-time or a recurring entity within a job without an action endpoint.

Type	Description
Schedule types	Runs one-time and recurring jobs or Cloud Foundry tasks.
Schedule formats	Supports several flexible schedule formats.
Schedule lifecycle	A schedule passes through three lifecycle states for each run.

Multitenancy You can use the service with multitenant applications developed in the context of a Platform-as-a-Service (PAAS) tenant or as a Software-as-a-Service (SaaS) tenant.

Related Information

Asynchronous Mode [page 8]
Schedule Types [page 9]
Schedule Formats [page 11]

Schedule Lifecycle [page 16]
Multitenancy in SAP Job Scheduling Service [page 17]
About Cloud Foundry Tasks

3.1 Asynchronous Mode

The SAP Job Scheduling service executes jobs that support action endpoints in a synchronous mode or in an asynchronous (or batch) mode.

Synchronous Mode

Synchronous requests are used if the SAP Job Scheduling service calls the action endpoint of the application and the application logic is executed in a short span of time.

> ✤ Example
>
> The exchange rate application fetches the latest currency exchange rates frequently using scheduled jobs.

> i Note
>
> - When the scheduler invokes the endpoint, the application must return the response with an appropriate HTTP status code, indicating success or failure.
> - To indicate success, the application must use a suitable standard status code between 200 and 399, except 202-ACCEPTED.
> - To indicate an execution failure, the application must use one of the server error codes as outlined in the HTTP protocol specification.

Asynchronous or Batch Mode

Asynchronous requests are used for job runs with a large span of time.

> ✤ Example
>
> Action endpoints that trigger long-running processes
>
> SAP Job Scheduling service operates in asynchronous mode if your application contains a module that synchronizes database tables, which are long-running tasks.

> i Note
>
> - When the scheduler invokes the endpoint, it passes the request headers values “x-sap-job-id”, “x-sap-job-schedule-id”, “x-sap-job-run-id”, and “x-sap-scheduler-host” for the Job ID, Job Schedule ID, Job Run ID, and the SAP Job Scheduling service Host URI, respectively. The application must extract the

header values and store them using a suitable mechanism, such as in-memory storage that uses caches or libraries, or persistent storage that uses a database.

- The application must return an acknowledgement response with the HTTP status code 202-ACCEPTED. The response indicates to the scheduler that the application has accepted and is processing the request. If the application returns a server error code, the scheduler interprets it as a failure of the job run.
- After the application completes the job processing, it must invoke the Update of Job Run Log API to indicate success or failure and (optionally) create log text for the job run. For information about this API, see Update Job Run Log [page 40]. The path parameters for the Update of Job Run Log API call must be the values of the headers as described above.
- If the application doesn't invoke the Update of Job Run Log API, the scheduler isn't notified of the status of the job run and, after a configurable time interval, reverts the job to the status UNKNOWN.

> **i** Note
>
> Cloud Foundry tasks always run asynchronously.

Related Information

About Cloud Foundry Tasks

3.2 Schedule Types

SAP Job Scheduling service runs one-time and recurring jobs or Cloud Foundry tasks.

One-time Schedules

Schedules that run only once. You can create one-time schedules using any one of the following options:

- **Human-readable text:** A human-readable text, denoting the specified execution time. For more information see, Human-Readable Date Formats [page 14].

  > Example
  >
  > 10 hours from now, 3.30pm
  >
  > Friday at 2am

- **Date String Formats :**
  A valid date representations in the ISO-8601 or IETF-compliant RFC 2822 format.

  > Example
  >
  > 1994-11-05T08:15:30-05:00

Recurring Schedules

Schedules that run periodically at specified times, dates, or intervals. You can create recurring schedules using any one of the following options:

- **repeatInterval:** Denotes the interval between schedules in a human-readable text format (for example, 2 minutes).
  If there is a delay, for example, because the load is very large and the run takes longer, the next run starts at a later point of time. The defined interval stays the same.
  For more information, see Human-Readable Date Formats [page 14]

  > ### Example
  > - "10 hours"
  > - "2 days"
  > - "3 seconds"

- **cron:** Denotes a crontab expression, which represents a set of times, that determines when the job is executed.
  For more information, see Cron Format [page 11].
- **repeatAt:** Denotes an exact time, every day, when a job is executed. This field can be specified using Human-Readable Time Format and Date String Formats (ISO-8601 or IETF-compliant RFC-2822).

  > ### Example
  > "4.40pm"
  >
  > "18.40"
  >
  > "6.20am"

Related Information

Schedule Lifecycle [page 16]
Service Behavior [page 71]

3.3 Schedule Formats

The service supports several scheduling formats.

3.3.1 Cron Format

The cron format enables you to define the job run schedule.

The format is like the standard Linux cron but provides more functional flexibility for scheduling application-based jobs. The following table explains the order of fields used in the cron and lists the permitted values for each field:

Cron field (Left to right)	Description
Year	4-digit representation for year. Example: 2015
Month	Numeric representation for a month. Permitted values are 1 to 12.
Day	Numeric representation for day of the month with values permitted from -31 to 31. Negative values are calculated from the end of the month.
DayOfWeek	3-letter representation for day of the week. Example: mon, tue, and so on.
Hour	Numeric representation for an hour. Permitted values are 0 to 23.
Minute	Numeric representation for a minute. Permitted values are 0 to 59.
Second	Numeric representation for a second. Permitted values are 0 to 59.

The table below displays the syntax allowed to define cron expressions:

Expression	Usage	Value
*	anywhere	any value
*/a	anywhere	any a-th value
a:b	anywhere	Values in the range a to b.
a:b/c	anywhere	Every c-th value between a and b.

Expression	Usage	Value
a.y	Day of the week	On the a-th occurrence of the weekday y (a = -5 to 5)
a,b,c	anywhere	a or b or c

Example

The table below displays the crontab expression with its description:

Expression	Description
2015 * * fri 12 0 0	Run the schedule every Friday in 2015 at 12:00.
* * 3:-2 * 12:14 0 0	Run every hour between 12:00 and 14:00 every day between the third and second-to-last day of the month.
* * * -1.sun 9 0 0	Run the schedule on the last Sunday of every month at 09:00.

3.3.2 Date and Time Format

You can provide date and time parameters such as startTime, endTime, or time for schedules as data type string or object.

Date Object

- For an object data type, a mandatory date field denoting the date as a string, and an optional format field denoting a date-time format for correctly parsing the user-provided date value, must be passed. The following constraints apply to object formats:
  - If you provide only a date field, the string is checked against both IETF-compliant RFC 2822 timestamps and ISO-8601 standards.

    Note

    For dates that are compliant with ISO-8601, calendar dates (example: 2013-02-08), week dates (example: 2013-W06-5), ordinal dates (example: 2013-039), and time-based dates (example: 2013-02-08 09+07:00) are all supported. If the date string is of an unknown format, an error is thrown.
  - If the format of the date string provided in the date field is customized, an optional format string can be passed. The allowed parsing tokens include the following:

Input Token	Example	Description
YYYY	2015	4-digit year
YY	14	2-digit year
Q	1-4	Quarter year. Sets month to first month of the quarter.
M MM	1-12	Month number
MMM MMMM	January-Dec	Month name in locale
D DD	1-31	Day of month
Do	1st - 31st	Day of month with ordinal
DDD DDDD	1-365	Day of year
X	1410715640.579	UNIX timestamp
x	1410715640579	UNIX timestamp (ms)
gggg	2015	Locale 4-digit week year
gg	15	Locale 2-digit week year
w ww	1-53	Locale week of the year
e	1-7	Locale day of the week
GGGG	2015	ISO 4-digit week year
GG	15	ISO 2-digit week year
W WW	1-53	ISO week of year
E	1-7	ISO day of week
H HH	0-23	24-hour time
h hh	1-12	12-hour time used with "a A"
a A	am pm	Before or after noon
m mm	0-59	Minutes
s ss	0-59	Seconds
S	0-9	Tenths of a second
SS	0-99	Hundredths of a second

Input Token	Example	Description
SSS	0-999	Thousandths of a second
Z ZZ	+12:00	Offset from UTC as +-HH:mm, +-HHmm, or Z

Example

Format	Description
"startTime": {"date": "2015-10-20 04:30 +0000", "format": "YYYY-MM-DD HH:mm Z"}	4.30 UTC on 20th Oct 2015
"endTime": {"date": "2015-W06-5"}	Friday, February 06, 2015

Date String

For a string data type, the parameters must be a valid date representation in the ISO-8601 or IETF-compliant RFC 2822 format.

Example

1994-11-05T08:15:30-05:00

3.3.3 Human-Readable Date Formats

Job Scheduler supports human-readable dates and ranges for configuring both job and task schedules.

These dates and ranges are applicable for the parameters time, repeatAt, and repeatInterval. Job Scheduler uses an embedded English language date parser for this facility.

Specification for Schedule Parameters

- Time
  The string value should designate a particular timestamp for running a job schedule.

  Note

  If an invalid string is provided, the results can be unpredictable.

- "10 hours from now"
- "20 minutes from now"
- "in 2 hours"
- "tomorrow at 4pm"
- "next week monday at 5am"
- "9pm tonight"
- "3.30pm"

- RepeatAt
  This parameter represents a convenient way to create daily timestamp-based schedules. The string should designate a particular timestamp for repeatedly running a job schedule. This follows the same pattern as the recommendations for the **time** parameter, with exception: while the text for the time parameter must denote something concrete and in the future, the repeatAt must designate a timestamp, which is valid and constant daily.

  > **i Note**
  >
  > If an invalid string is used, the scheduler falls back to the current timestamp and runs the schedule immediately.

  > **Example**
  >
  > - "4.40pm"
  > - "18.40"
  > - "6.20am"
  > - "17.20:30"
  >
  >   > **⚠ Caution**
  >   >
  >   > Second-based precision may not always be accurately timed

- RepeatInterval
  This parameter should designate an interval for repeating the job execution.

  > **i Note**
  >
  > Number names to denote numeric values are not supported yet.

  > **Example**
  >
  > Instead of “twenty minutes”, use “20 minutes” to denote the interval.

  Supported time-units for this parameter are "years", "months", "weeks", "days", "hours", "minutes", "seconds".

  > **Example**
  >
  > - "10 hours"
  > - "2 days"
  > - "3 seconds"

3.4 Schedule Lifecycle

A schedule passes through three lifecycle states for each run.

The lifecycle states are described in the following table:

State	Description
Scheduled	The schedule has been queued for a future run.
Running	The schedule is currently being executed.
Completed	The schedule run has been completed.

Each lifecycle state has a transitional state. The combination of these states helps you analyze and understand the schedule run. You can view the state details corresponding to a schedule in the schedule logs.

Each combination is described in the following table:

State	Transitional State	Description
Scheduled	Scheduled	The schedule has been queued for a future run.
Running	Triggered	The scheduler has triggered a request to the job action endpoint.
	ACK_RECVD	The application sends an "202-Accepted" message thus acknowledging the request sent by the scheduler.
	ACK_NOT_RECVD	Even after a certain time, the application hasn't sent an "202-Accepted" message acknowledging the request sent by the scheduler.
Completed	Request_Error	Error encountered while invoking the job action endpoint.
	Success	The application has successfully executed the job and has replied with a status code (ranges from 200 to 399 except 202-ACCEPTED) to the scheduler.
	Error	The application has encountered some error during the execution and has sent one of the server error codes as outlined in the HTTP protocol specification.
	Unknown	The application hasn't invoked the Job Run Log Update API to update the scheduler about the status of the job execution. The status will be set as UNKNOWN, after a configurable time interval.

The following table helps you understand the status of actions corresponding to Cloud Foundry tasks:

State	Transitional State	Description
Scheduled	Scheduled	The schedule has been queued for a future run.
Running	Triggered	The scheduler has triggered a request to the cloud controller for creating Cloud Foundry tasks.
	ACK_RECVD	The cloud controller successfully creates a task.
Completed	Request_Error	Error encountered while creating a Cloud Foundry task.
	Success	Cloud Foundry task successfully completed.
	Failed	Task didn't complete successfully resulting in a non-zero exit code.
	Unknown	Task wasn't available for SAP Job Scheduling service to fetch the status.

Related Information

Manage Jobs with Service Dashboard [page 64]

3.5 Multitenancy in SAP Job Scheduling Service

The SAP Job Scheduling service is a multitenant-aware service. You can use the service with multitenant applications developed in the context of Platform-as-a-Service (PAAS) tenant or as a Software-as-a-Service (SaaS) tenant.

A multitenant application deployed on the cloud platform can create/view/edit/delete jobs corresponding to a tenant which has subscribed to the application. The SAP Job Scheduling service provides the tenant context access token to such multitenant applications. This helps the application to identify the tenant for which it has to perform a specific job or action.

Adopting Multitenancy Features

To make a multitenant application available for subscription to SaaS customers or consumers, as an application provider you need to register the application in the Cloud Foundry environment. This is a one-time activity and done using the SaaS Registry Service.

As required by the SaaS registry service, the **getDependencies** callback must return all dependencies of the application. The SAP Job Scheduling service must be added as a dependency to support the multitenancy feature. You can add dependency as shown below:

Sample Code

```
[
    {
        "appId": "<xsappname-provided-by-jobscheduler>",
        "appName": "sap-jobscheduler"
    }
]
```

Note

You can find the **xsappname** from the credentials provided by the SAP Job Scheduling service while binding the service instance with your application. Replace the placeholder with the actual application name.

Create Jobs on behalf of Tenants

Multitenant applications create jobs using access tokens obtained from XSUAA. These are details such as **client, client secret,** and **url** provided by the SAP Job Scheduling service while binding the application (credentials.uaa) and **client_credentials** or **user_token** as grant types. While calling the SAP Job Scheduling service API, the access token should be provided in the Authorization header as **Bearer <access token>**.

Note

The access token obtained will contain tenant details of SaaS customers for the **user_token** grant type. It will also contain tenant details of the multitenany application (PaaS tenant) for the **client_credentials** grant type.

SAP Job Scheduling service APIs and Multitenancy

The multitenant application calls the SAP Job Scheduling service APIs using the access token obtained as explained in the previous section. However, the following constraints are applicable:

- Jobs created by the multitenant application for a SaaS customer (using grant type **user_token**) isn't accessible to other SaaS customers of the application. The APIs return results filtered by tenant information provided by the access token used to call the API.
- The multitenant application can't create/read/edit/delete the data of a SaaS customer using an access token obtained on behalf of another SaaS customer.
- API calls using an access token obtained with **client_credentials** grant type with SaaS subdomain is applicable for SaaS customers of the application. The APIs return results filtered by tenant information provided by the access token used to call the API.

- API calls using an access token obtained with **client_credentials** grant type with PaaS subdomain is applicable for the service instance. The multitenant application can use such access tokens to create/read/edit/delete the data for the service instance without any restrictions of SaaS customers.

4 Initial Setup

You can get started with the SAP Job Scheduling service using the standard procedures for the Cloud Foundry environment.

Prerequisites

- You've set up your global account and subaccount on SAP BTP.
  For an overview of the required steps, see Getting Started in the Cloud Foundry Environment and Get a Global Account.
- You're a global account administrator.
- You've purchased quota for the SAP Job Scheduling service and for the SAP Authorization and Trust Management service.
- You've created a space within a subaccount in which is enabled.
  For more information, see Managing Orgs and Spaces Using the Cockpit and Account Administration in the Cloud Foundry Command Line Interface.
- Within that space, your user is assigned to the *Space Developer* role for the subaccount. You need this role to execute the configuration steps and to start the dashboard.
  For more information, see Managing Authorization in Global Accounts and Subaccounts [Feature Set B] and Assign Role Collections.
- You've deployed an application on SAP BTP.
- You've exposed an action endpoint in your application that can be called from the SAP Job Scheduling service.
- The application and the SAP Job Scheduling service are in the same space.

Enabling the Service

To enable the SAP Job Scheduling service and the SAP Authorization and Trust Management service, distribute entitlements and quotas across subaccounts within your global account.

Distribute the quota for the service plan *standard* of the SAP Job Scheduling service to your subaccount.

For more information, see Managing Entitlements and Quotas Using the Cockpit and Configure Entitlements and Quotas for Subaccounts.

> → Remember
>
> You must also enable the SAP Authorization and Trust Management service.

Setting Up Roles and Authorization

To create and bind service instances to applications, you must be assigned the **Space Manager** or the **Space Developer** role in the space.

For more information, see About Roles in the Cloud Foundry Environment.

Getting Started

Once you've completed the initial setup, see Getting Started [page 22] and our tutorial for beginners.

Related Information

Getting Started [page 22]
Using SAP Job Scheduling Service [page 26]

5 Getting Started

Once you've completed the initial setup for SAP Job Scheduling service, to use the service you create a service instance and bind it to your application.

In the Cloud Foundry environment, you enable services by creating a service instance using either the SAP BTP cockpit or the Cloud Foundry command line interface (CLI), and binding that instance to your application. A service instance is a single instantiation of a service running on SAP BTP. For more information, see About Services.

Determine how to proceed based on whether you're using a new application or an existing application that is already bound to an xsuaa service instance:

> → Remember
>
> You must perform the following steps in the given order for the SAP Job Scheduling service to work.

Using a New Application:

If you're using a new app, proceed as follows:

1. Create an instance of the SAP Job Scheduling service.
2. Create an xsuaa service instance.

   > i Note
   >
   > If you already have an xsuaa service instance, remember to update it first with the updated `xs-security.json` file.
   >
   > This is necessary as the content in the `xs-security.json` in which the scope is granted to the SAP Job Scheduling service with the `grant-as-authority-to-apps` has changed.
3. Deploy the application.

   > → Tip
   >
   > When you deploy an application to a space and want to use the SAP Job Scheduling service, you must first create a SAP Job Scheduling service instance in the space you're working in. Then, when you push (or upload) your application, you can bind your application to the SAP Job Scheduling service instance.
   >
   > For more information, see Binding Service Instances to Applications.
4. Bind the xsuaa service instance to your application.
5. Bind the SAP Job Scheduling service instance to your application.

   > i Note
   >
   > If you don't bind your application to the SAP Job Scheduling service instance, you can't call the SAP Job Scheduling service from your own application.

Using an Existing Application:

If you're using an existing app, proceed as follows:

1. Create an instance of the SAP Job Scheduling service.
2. Update your xsuaa service instance.

   > **i Note**
   >
   > This is necessary as the content in the `xs-security.json` where the scope is granted to the SAP Job Scheduling service with the `grant-as-authority-to-apps` has changed.

3. Bind the SAP Job Scheduling service instance to your application.

Related Information

Create a Service Instance in SAP BTP Cockpit [page 23]
Create a Service Instance Using CF CLI [page 24]
Define and Grant Scopes to SAP Job Scheduling Service [page 77]
Update a Service Instance
Initial Setup [page 20]

5.1 Create a Service Instance in SAP BTP Cockpit

To use the SAP Job Scheduling service with action endpoints in your application or with Cloud Foundry tasks, you create an instance of the SAP Job Scheduling service using the SAP BTP cockpit and bind the service instance to your application.

Prerequisites

You've logged on to the SAP BTP cockpit cockpit. For more information, see Navigate to Orgs and Spaces.

Procedure

1. In the *Service Marketplace*, choose *Job Scheduling Service*.
2. From the *Job Scheduling Service* tile, open the *Actions* menu and choose *Create*.

   The *New Instance* wizard appears.

3. Choose the service plan *standard*.
4. Choose a name for your service instance, then choose *Next*.
5. To enable User Account and Authentication (xsuaa), in *Configure instance parameters* specify the `enable-xsuaa-support` parameter as true.

   You can specify a JSON file to upload containing the parameter or enter details in JSON format as follows :

   `{"enable-xsuaa-support":true}`.
6. Choose *Next* and verify the instance details. Then choose *Create*.
   You've created a service instance in your Cloud Foundry Space that's now available in the service instances section.
7. Choose a deployed application that you want to bind to the new instance of SAP Job Scheduling service.

   For more information, see Binding Service Instances to Cloud Foundry Applications.

Results

Your service instance is now bound to your application and you can start using the SAP Job Scheduling service.

Related Information

Deploy Business Applications in the Cloud Foundry Environment
Creating Service Instances in Cloud Foundry

5.2 Create a Service Instance Using CF CLI

To use the SAP Job Scheduling service with action endpoints in your application or with Cloud Foundry tasks, you create an instance of the SAP Job Scheduling service with the service plan `<standard>` using the Cloud Foundry Command Line Interface (CF CLI) and bind the service instance to your application.

Prerequisites

- Install CF CLI on your system. For more information, see Download and Install the Cloud Foundry Command Line Interface.
- Deploy the application that requires the SAP Job Scheduling service to the cloud platform. For more information, see Deploy Business Applications in the Cloud Foundry Environment.

Procedure

1. Open the Cloud Foundry console client.
2. Check the Service Marketplace to verify that the SAP Job Scheduling service service is listed: `cf marketplace`
3. Create an instance of the service: `cf create-service jobscheduler <standard> <instance name> -c '{"enable-xsuaa-support": true}'`.

   > **i Note**
   >
   > Enable xsuaa by specifying the `enable-xsuaa-support` parameter as true.

4. Bind the service instance to the application using the command: `cf bind-service <application name> <instance name>`
5. Restage the application using the command : `cf restage <application name>`
6. Verify that the instance is successfully bound to the application using the command: `cf env <application name>`

   > **i Note**
   >
   > If the instance is successfully bound, you see SAP Job Scheduling service details and credentials in the `VCAP_SERVICES` environment variable.

```
A sample VCAP_Services environment variable appears as below:
"VCAP_SERVICES" : {
            "jobscheduler" : [ {
              "name" : "scheduler-instance",
              "label" : "jobscheduler",
              "tags" : [ "jobscheduler" ],
              "plan" : "<service plan name>",
              "credentials" : {
                "clientid": "sb-a02e332f-f761-4051-a451-e078e9254fe3!b5618|
sap-jobscheduler!b21",
                "clientsecret": "<SECRET>",
                "url" : "rest-api-url of jobscheduler"
                "xsappname" : "xsappname-of-jobscheduler",
                "identityzone" : "identityzone-of-jobscheduler"
              }
            } ]
          }
```

6 Using SAP Job Scheduling Service

Define and manage one-time and recurring jobs using flexible schedules.

Related Information

Concepts [page 7]
SAP Job Scheduling Service REST APIs [page 26]
Manage Jobs with Service Dashboard [page 64]
Getting Started [page 22]

6.1 SAP Job Scheduling Service REST APIs

Use SAP Job Scheduling service REST APIs to create, manage, and monitor jobs and job schedules.

Authentication

Service Plan `lite`

The SAP Job Scheduling service REST APIs are guarded by HTTP Basic authentication, which requires a user name and a password. Both are fetched from the `VCAP_SERVICES` section of the application environment after binding it to the SAP Job Scheduling service.

Use the following command to view the `VCAP_SERVICES` content:

```
$ cf env <application-name>
{
    "VCAP_SERVICES": {
        "jobscheduler": [
            {
                "credentials": {
                    "clientid": "PRIVATE_DATA_HIDDEN",
                    "identityzone": "PRIVATE_DATA_HIDDEN",
                    "password": "PRIVATE_DATA_HIDDEN",
                    "url": "https://jobscheduler-
rest.cfapps.stagingaws.hanavlab.ondemand.com",
                    "user": "PRIVATE_DATA_HIDDEN",
                    "xsappname": "PRIVATE_DATA_HIDDEN"
                },
                "label": "jobscheduler",
                "name": "js-instance",
                "plan": "default",
                "provider": null,
                "syslog_drain_url": null,
```

```
                "tags": [
                    "jobscheduler"
                ]
            }
        ]
    }
}
```

All APIs are accessed through HTTPS.

> **i** Note
>
> All data is sent and received in the JSON format. When sending data through API requests, make sure that the HTTP header `Content-type` is set to `application/json`.

Service Plan `standard`

You can access the REST API for a `standard` instance by providing an OAuth token.

In the `VCAP_SERVICES` of your application, you can find the following section:

```
{
    "jobscheduler": [
        {
            "..": "..",
            "credentials": {
                "uaa": {
                    "apiurl": "<apiurl>",
                    "clientid": "<clientid>",
                    "clientsecret": "<clientsecret>",
                    "identityzone": "<identityzone>",
                    "identityzoneid": "<identityzoneid>",
                    "sburl": "<sburl>",
                    "tenantid": "<tenantid>",
                    "tenantmode": "dedicated",
                    "uaadomain": "<uaadomain>",
                    "url": "<url>",
                    "verificationkey": "<key>",
                    "xsappname": "<xsappname>"
                },
                "url": "https://jobscheduler-rest.<landscape-domain>"
            },
            "...": "..."
        }
    ]
}
```

1. Fetch an OAuth token from xsuaa by using the provided `clientid` and `clientsecret`:

   ```
   Method: POST
   Url: <value from credentials.uaa.url>
   Header:
       Content-Type: application/x-www-form-urlencoded
       Authorization: Basic <Base64 encoded clientid:clientsecret>
       Cache-Control: no-cache
   Body:
       grant_type=client_credentials
   ```

   The response should look as follows:

   ```
   {
       "access_token": "<token>",
       "token_type": "bearer",
   ```

```
    "expires_in": 43199,
    "scope": "<scope(s)>",
    "jti": "<hidden>"
}
```

2. Use the `access_token` in your request to the SAP Job Scheduling service REST API:

```
...
Url: <value from credentials.url>/scheduler/
Header:
    Authorization: Bearer <access_token>
...
```

Binding Level Secrets

New clientsecrets are created every time you bind a service.

> **i Note**
>
> The `clientsecret` included in the `VCAP_SERVICES` section of your application is specific to the binding.

This allows you to rotate secrets by unbinding and then binding the SAP Job Scheduling service instance again to the application.

The `clientsecret` assigned to the service instance becomes invalid if you unbind the service and the access token requests to the SAP Job Scheduling service REST API using it fail.

> **i Note**
>
> Before this change, service instances using instance level secrets continue to use these and they remain valid. It's possible to rotate these out for binding level secrets by rebinding the service instance to the application.

Pagination

All SAP Job Scheduling service APIs support pagination. You can retrieve all data if you don't pass any query parameter.

Time Zone

Run the SAP Job Scheduling service only in the UTC time zone.

> **⚠ Caution**
>
> Currently, no other time zones than UTC are supported.

Overview of All SAP Job Scheduling service REST APIs

6.1.1 Create Job

This API creates a job by accepting one or more job schedules to be created.

Routes

POST /scheduler/jobs

Request Parameters

BODY

Parameter	Required	Data Type	Description
`name`	Yes	string	Name of the job. Name must not contain special characters or only numbers. **i Note** The job creation request fails if a job with the same name already exists for the technical user. .
`description`	Yes	string	Provides more details about a job.
`action`	Yes	string	The fully qualified URL endpoint to be called when the job runs.
`active`	Yes	boolean	Activation status of the job (default value is false).
`httpMethod`	Yes	string	The HTTP method to be used to call the job action endpoint URL. Allowed values are “GET”, “POST” , “PUT”, and “DELETE”.
`startTime`	No	string/object	Start time for the job. The scheduler checks if a start time for a schedule is available apart from the start time available for a job. The schedule start time is used for determining the start of the schedule run. If the schedule start time is not available, the start time of the job is used. For information about the supported formats, see the section on Date and Time Format [page 12].

Parameter	Required	Data Type	Description
`endTime`	Yes	string/object	End time for the job. The scheduler checks if an end time for a schedule is available apart from the end time available for a job. The schedule end time is used for determining the end of the schedule run. If the schedule end time is not available, the end time of the job is used. For information about the supported formats, see the section on Date and Time Format [page 12].
`schedules`	Yes	array	Contains one or more job schedules to be created.
The schedules array can be used to create job schedules. Each job schedule is composed of the following parameters:			
`data`	No	object	Data value is passed to the job action endpoint when invoked by Job Scheduler. For the HTTP method "PUT" or "POST", the data parameters are sent in the request body while invoking the end-point. For the HTTP method "GET" or "DELETE", the data parameters are sent as query strings.
`description`	No	string	Provides more details about a schedule.
`active`	Yes	boolean	Activation status of the job schedule. The allowed values are true or false.
`startTime`	No	string/object	Start time for the job schedule. For information about the supported formats, see the section on Date and Time Format [page 12].
`endTime`	No	string/object	End time for the job schedule. For information about the supported formats, see the section on Date and Time Format [page 12].

`cron`, `time`, `repeatInterval`, and `repeatAt` are the different scheduling options available. Use any one of the option that suits your requirement.

> **→ Remember**
>
> You must specify at most one scheduling mode.

`cron`	-	string	Crontab pattern for triggering the schedule. For more information, see the section on Cron Format [page 11].

Parameter	Required	Data Type	Description
`time`	-	string/object	For one-time schedules, this denotes the task execution time. You can use human-readable text to denote a specific time. Example: "3.30pm", "tomorrow at 2am". For information about human readable dates and the supported readable strings, see the section on Human-Readable Date Formats [page 14] . If an object is used, you must specify the date and time formats as strings. For information about the supported formats, see the section on Date and Time Format [page 12].
`repeatInterval`	-	string	Used to run schedules repeatedly at some interval. Human-readable texts must comply with the rules outlined for human readable dates. For information about the supported formats, see the section on Date and Time Format [page 12].
`repeatAt`	-	string	For recurring schedules, this denotes the exact time when the job schedule must run. The human readable strings must comply with the rules outlined in the section Human Readable Dates. For information about human readable dates and the supported readable strings, see the section on Date and Time Format [page 12].

Example

POST /scheduler/jobs

```
{
  "name": "validateSalesOrder",
  "description": "cron job that validates sales order requests",
  "action": "http://salesOrderApp.hana.ondemand.com:40023/salesOrders/validate",
  "active": true,
  "httpMethod": "PUT",
  "schedules": [
    {
      "cron": "* * * * * */10 0",
      "description": "this schedule runs every 10 minutes",
      "data": {
        "salesOrderId": "1234"
      },
      "active": true,
      "startTime": {
        "date": "2015-10-20 04:30 +0000",
        "format": "YYYY-MM-DD HH:mm Z"
      }
    }
  ]
}
```

Responses

Status Code 201

The call was successful and the Job has been created. The API responds with a **Location** header which represents the relative resource URI for finding the job details.

Example

```
{
  "name": "validateSalesOrder",
  "action": "http://<application-url>/action",
  "active": true,
  "httpMethod": "PUT",
  "description": "cron job that validates sales order requests",
  "startTime": null,
  "endTime": null,
  "signatureVersion": 0,
  "schedules": [
    {
      "active": true,
      "startTime": "2015-10-20 04:30:00",
      "endTime": null,
      "description": "every 10 seconds, every 2 minutes",
      "data": "{\"salesOrderId\":\"1234\"}",
      "cron": "* * * * * */10",
      "type": "recurring",
      "scheduleId": "schedule ID details"
    }
  ],
  "_id": 3
}
```

Status Code 400

The API was unable to process the request due to invalid data provided.

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]

Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.2 Configure Job (Using ID)

This API configures a job with the updated runtime information using job ID.

You can also use this API to create a job by providing a job name along with the job in the URI. If you use the API to create a job, the parameters must conform to the same constraints as provided in the create job API. You must also ensure that the job name in the request URI matches with the job name in the request body.

Routes

PUT /scheduler/jobs/ {jobId}

Request Parameters

Path

Parameter	Required	Data Type	Description
jobId	Yes	integer	ID of the job to be updated

Body

> → Remember
>
> At least one among the following parameters are mandatory.

Parameter	Data Type	Description
`active`	boolean	Determines if a job must be activated or deactivated. If the parameter is not provided in the request, the existing status is retained.
`httpMethod`	string	HTTP method of the job action endpoint URL. Allowed values are “GET”, “POST”, “PUT”, and “DELETE”. If the parameter is not present in the request, the existing method is retained.

Parameter	Data Type	Description
`startTime`	string/object	Start time for the job. The scheduler checks if a start time for a schedule is available apart from the start time available for a job. The schedule start time is used for determining the start of the schedule run. If the schedule start time is not available, the start time of the job is used. For more information, see Date and Time Format [page 12].
`endTime`	string/object	End time for the job. The scheduler checks if an end time for a schedule is available apart from the end time available for a job. The schedule end time is used for determining the end of the schedule run. If the schedule end time is not available, the end time of the job is used. For information about supported formats, see Date and Time Format [page 12].

Example

PUT /scheduler /jobs/3

```
{
    "active": true,
    "httpMethod": "GET"
}
```

Responses

Status Code: 200

The call was successful.

Example

```
{
  "success": true
}
```

Status Code: 400

The API was unable to process the request due to invalid data provided.

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using Name) [page 36]

6.1.3 Configure Job (Using Name)

This API configures a job with the updated runtime information using job name.

If you use this API to create a job, the parameters must conform to the same constraints as provided in the create job API. You must also ensure that the job name in the request URI matches with the job name in the request body.

> **i** Note
>
> If the job name already exists, the API will replace the job content with the new content retaining the same job ID and name.

[Additional information about the method, if needed.]

Routes

PUT /scheduler/jobs/ {name}

Request Parameters

Path

Parameter	Required	Data Type	Description
`name`	Yes	string	Name of the job to be updated.

Body

The parameters are the same as in Create Job API.

Example

PUT /scheduler/jobs/newSalesOrderJob

```
{
  "name": "newSalesOrderJob",
  "description": "Sales order Job",
  "action": "http://<application-url>/action",
  "active": true,
  "httpMethod": "PUT",
  "schedules": [
    {
      "cron": "* * * * * */10 0",
      "description": "this schedule runs every 10 minutes",
      "data": {
        "order_id": "1234"
      },
      "active": true,
      "startTime": {
        "date": "2015-10-20 04:30 +0000",
        "format": "YYYY-MM-DD HH:mm Z"
      }
    }
  ]
}
```

Responses

The response body is the same as specified in Create Job [page 30] API.

Status Code: 200

Status Code: 200

The call was successful.

Example

```
{
  "name": "newSalesOrderJob",
  "action": "http://sales.acme.com/orders",
  "active": true,
  "httpMethod": "PUT",
  "description": "Sales order Job",
  "startTime": null,
  "endTime": null,
  "signatureVersion": 0,
  "schedules": [
    {
      "active": true,
      "startTime": "2015-10-20 04:30:00",
      "endTime": null,
      "description": "this schedule runs every 10 minutes",
      "data": "{\"order_id\":\"1234\"}",
      "cron": "* * * * * */10 0",
      "type": "recurring",
      "scheduleId": "63c05418-f399-49ad-b141-e128f1a3b114"
    }
  ],
  "_id": 4
}
```

Status Code: 201

If you use the API to create job, this response code appears indicating that a new job was created.

Status Code: 400

The API was unable to process the request due to invalid data provided.

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.4 Delete Job

This API deletes a job and all its runtime information such as schedules and logs.

It also terminates processing of all schedules (active and inactive) of the job.

Routes

DELETE /scheduler/jobs/{jobId}

Request Parameters

Path

Parameter	Required	Data Type	Description
jobId	Yes	integer	ID of the job to be deleted

Example

```
DELETE /scheduler/jobs/4
```

Responses

Status Code: 200

The job was deleted successfully.

Example

```
{
  "success": true
}
```

Status Code: 404

Passing invalid Job ID.

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.5 Update Job Run Log

This API is used by the application to inform the Job Scheduler about the status of an asynchronous, long-running job.

An application must call this API after completion of the asynchronous execution of the job. The API expects the status of the job run and optionally an appropriate message.

Routes

PUT /scheduler/jobs/ {jobId}/schedules/{scheduleId}/runs/{runId}

> **i** Note
>
> Job Scheduler passes jobId, scheduleId, and runId as part of the request headers while invoking the job action. For more information, see Asynchronous Mode [page 8].

Request Parameters

Path

Parameter	Required	Data Type	Description
jobId	Yes	integer	ID of the job to which the scheduleId belongs
scheduleId	Yes	string	ID of the schedule to which runId belongs
runId	Yes	string	ID of the run whose logs are to be retrieved

Body

Parameter	Required	Data Type	Description
`success`	Yes	boolean	Indicates the status of the job run. Allowed values are true or false.
`message`	No	string	Information that accompanies the job run status.

Request Example

```
PUT /scheduler/jobs/3/schedules/cb5c9def-e2a0-4294-8a51-61e4db373f99/runs/
ea16b621-eaa8-4824-8629-ff6e6221bb56
{
  "success": true,
  "message": "Long running operation to process employee salaries succeeded"
}
```

Responses

Status Code: 200

The log for the job run is successfully updated.

Example

```
{
  "success": true
}
```

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.6 Retrieve Job Details

This API retrieves the saved configuration settings of a specified job, optionally with its schedules.

It retrieves the job schedules if the `displaySchedules` parameter is true.

Routes

GET /scheduler/jobs/{jobId}

You can also use `jobId` or `name` as query parametes to fetch the job details.

- *GET /scheduler/jobs?name={jobName}*
- *GET /scheduler/jobs?jobId={jobId}*

> **i** Note
>
> If both the parameters are used, ensure that the D matches with the corresponding name.

Request Parameters

Path

Parameter	Required	Data Type	Description
jobId	Yes	number	ID of the job to be retrieved

Query

Parameter	Required	Data Type	Description
`displaySchedules`	No	boolean	Controls if the API should return all the schedule details of the job. Allowed value is true or false.
`jobId`	Yes	number	ID of the job to be retrieved.
`name`	No	string	Name of the job to be retrieved.

Using `jobId` as path parameter

```
GET /scheduler/jobs/3?displaySchedules=true
```

Using `name` as query parameter

```
GET /scheduler/jobs?name=mySalesOrder&displaySchedules=true
```

Responses

Status Code: 200

The call was successful and returns the details of the job (and its schedules, based on the request parameters).

Example

```
{
"name": "mySalesOrderJob",
  "description": "cron job that validates sales orders",
  "action": "http://<application-uri>/action",
  "user": "abc",
  "httpMethod": "GET",
  "startTime": null,
  "endTime": null,
  "signatureVersion": 0,
  "active": true,
```

```
  "_id": 3,
  "schedules": [
    {
      "data": {
        "customData": "some_value"
      },
      "type": "recurring",
      "cron": "* * * * * */2 */10",
      "active": false,
      "startTime": "2015-10-19T23:00:00.000Z",
      "endTime": null,
      "scheduleId": "<schedule ID details>",
      "description": "every 10 seconds, every 2 minutes"
    }
  ]}
}
```

Status Code: 404

Passing invalid Job ID.

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.7 Retrieve Job Run Logs

This API retrieves the details for a specified job schedule.

This API supports pagination, if required parameters are set.

Routes

GET /scheduler/jobs/{jobId}/schedules/{scheduleId}/runs

Request Parameters

Path

Parameter	Required	Data Type	Description
`jobId`	Yes	number	ID of the job to which scheduleId belongs
`scheduleId`	Yes	string	ID of the schedule whose run-logs has to be retrieved

Query

Parameter	Required	Data Type	Description
`page_size`	No	integer	Number of job run logs that the API retrieves. The default number is 10. If `page_size` is not set, and only `offset` is specified, the `page_size` parameter will take a default value of 10 records.
offset	No	integer	Number of job run log records to skip. The default number is 1. The default `offset` is an indicator to fetch results without skipping any records.

Example

```
GET /scheduler/jobs/2/schedules/5f58c2fb-a428-4f4b-9e1d-312e3be8952c/runs?
page_size=5&offset=1
```

Responses

Status Code: 200

The call was successful and returns the collection of job run logs.

Example

```
{
"total": 6,
"results": [
    {
        "runId": "5646889BB133728EE10000000A61A0D8",
        "runText": "Something wrong happened",
        "httpStatus": 500,
        "executionTimestamp": "2015-11-14T04:15:22",
        "runStatus": "COMPLETED",
        "runState": "ERROR",
        "scheduleTimestamp": "2015-11-14T04:13:22",
```

```
        "completionTimestamp": "2015-11-14T04:15:22"
    },
    {
        "runId": "56468B1AB133728EE10000000A61A0D8",
        "runText": "Request Error: Code:-ENOTFOUND, Message:-getaddrinfo
ENOTFOUND www.app.acme.com",
        "httpStatus": 404,
        "executionTimestamp": "2015-11-14T04:17:22",
        "runStatus": "COMPLETED",
        "runState": "REQUEST_ERROR",
        "scheduleTimestamp": "2015-11-14T04:15:22",
        "completionTimestamp": "2015-11-14T04:17:22"
    },
    {
        "runId": "56468DB7B133728EE10000000A61A0D8",
        "runText": "Sales order processed",
        "httpStatus": 200,
        "executionTimestamp": "2015-11-14T04:19:22", //indicates when actually
the scheduler invoked action endpoint
        "runStatus": "COMPLETED",
        "runState": "SUCCESS",
        "scheduleTimestamp": "2015-11-14T04:17:22", //indicates when the
schedule was picked up for calculation of next-run
        "completionTimestamp": "2015-11-14T04:19:22" //indicates when the
scheduler received response from the action endpoint
    }

  ]
}
```

Status Code: 404

Passing invalid Job ID.

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.8 Retrieve Job Run Log Details

This API retrieves the details for a specified job run log.

Routes

GET /scheduler/jobs/{jobId}/schedules/{scheduleId}/runs/{runId}

Request Parameters

Path

Parameter	Required	Data Type	Description
jobId	Yes	Integer	ID of the job to which scheduleId belongs
scheduleId	Yes	string	ID of the schedule to which runId belongs
runId	Yes	string	ID of the run whose logs are to be retrived

Example

```
GET /scheduler/jobs/2/schedules/5f58c2fb-a428-4f4b-9e1d-312e3be8952c/runs/
5646889BB133728EE10000000A61A0D8
```

Responses

Status Code: 200

The call was successful and returns details of the job run log.

Example

```
{
        "runId": "56468DB7B133728EE10000000A61A0D8",
        "runText": "Sales order processed",
        "httpStatus": 200,
        "executionTimestamp": "2015-11-14T04:19:22", //indicates when actually
the scheduler invoked action endpoint
        "runStatus": "COMPLETED",
        "runState": "SUCCESS",
        "scheduleTimestamp": "2015-11-14T04:17:22", //indicates when the
schedule was picked up for calculation of next-run
        "completionTimestamp": "2015-11-14T04:19:22" //indicates when the
scheduler received response from the action endpoint
}
```

Status Code: 404

Passing invalid Job ID.

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.9 Create Job Schedule

This API creates a job schedule for a specified job.

The job configuration settings (Action URL, HTTP Method, User, Password, and Job Activation Status) are valid for the new schedule.

Routes

POST /scheduler/jobs/{jobId}/schedules

Request Parameters

Path

Parameter	Required	Data Type	Description
`jobId`	Yes	Integer	ID of the job for which schedules are to be created.

Body

Parameter	Required	Data Type	Description
`data`	No	object	Data value is passed to the job action endpoint when invoked by Job Scheduler. For the HTTP method “PUT” or “POST”, the data parameters are sent in the request body while invoking the endpoint. For the HTTP method “GET” or “DELETE”, the data parameters are sent as query strings .
`description`	No	string	Provides more details about a schedule.
`active`	Yes	boolean	Activation status of the job schedule. The allowed values are true or false.
`startTime`	No	string/object	Start time for the job schedule. For information about the supported formats, see Date and Time Format [page 12].
`endTime`	No	string/object	End time for the job. For information about the supported formats, see Date and Time Format [page 12].
`Cron`, `time`, `repeatInterval`, and `repeatAt` are the different scheduling modes available. Use any one that suits your requirement. You must specify at most one scheduling mode.			
`cron`	-	string	Crontab pattern for triggering the schedule. For more information, see the section on Cron Format [page 11].
`time`	-	string	For one-time schedules, this denotes the task execution time. You can use human-readable text to denote a specific time. Example: "3.30pm", "tomorrow at 2am". For information about human readable dates and the supported readable strings, see Date and Time Format [page 12]. If an object is used, you must specify the date and time formats as strings. For information about the supported formats, see Date and Time Format [page 12].

Parameter	Required	Data Type	Description
`repeatInterval`	-	string	Used to run schedules repeatedly at some interval. Human-readable texts must comply with the rules outlined for human readable dates. For information about the supported formats, see Human-Readable Date Formats [page 14].
`repeatAt`	-	string	For recurring schedules, this denotes the exact time when the job schedule must run. The human readable strings must comply with the rules outlined in the section Human Readable Dates. For information about human readable dates and the supported readable strings, see Human-Readable Date Formats [page 14].

Example

```
PUT /scheduler/jobs/3/schedules
{
    "repeatInterval":"2 hours",
    "active":true,
    "description":"New Schedule",
    "startTime": {
        "date": "2017-08-21",
        "format": "YYYY-MM-DD"
    }
}
```

Responses

Status Code: 201

The API call was successful and a schedule has been created.

The API responds with a Location header representing the relative resource URI for finding the schedule details.

Example

```
{
  "repeatInterval": "2 hours",
  "repeatAt": null,
  "time": null,
  "cron": null,
  "data": "{\"order_id\":\"abcd\"}",
  "description": "New Schedule",
  "type": "recurring",
  "active": true,
  "startTime": "2015-04-20 18:30:00",
  "endTime": null,
  "jobId": 3,
  "scheduleId": "<schedule ID details>"
}
```

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.10 Configure Job Schedule

This API configures/updates the runtime information of a job schedule for a specified job.

All job configuration settings (Action URL, HTTP Method, User, Password, and Job Activation Status) will remain valid for the modified schedule. If a job schedule is in the execution queue, calling this API stops the execution process. If the schedule was identified for execution, the updated schedule is executed immediately.

Routes

PUT /scheduler/jobs/{jobId}/schedules/{scheduleId}

Request Parameters

Path

Parameter	Required	Data Type	Description
`jobId`	Yes	Integer	ID of the job to which scheduleId belongs
`scheduleId`	Yes	string	ID of the schedule that has to be updated

Body

> **Note**
>
> At least one among the following parameters is mandatory.

Parameter	Data Type	Description
`data`	object	Data value is passed to the job action endpoint when invoked by Job Scheduler. For the HTTP method "PUT" or "POST", the data parameters are sent in the request body while invoking the endpoint. For the HTTP method "GET" or "DELETE", the data parameters are sent as query strings
`description`	string	Provides more details about a schedule.
`active`	boolean	Activation status of the job schedule.The allowed values are true or false.
`startTime`	string/object	Start time for the job schedule. For information about the supported formats, see Date and Time Format [page 12].
`endTime`	string/object	End time for the job schedule. For information about the supported formats, see Date and Time Format [page 12].

`Cron`, `time`, `repeatInterval`, and `repeatAt` are the different scheduling modes available.

> **Note**
>
> The API does not support changing the original scheduling mode for the job schedule. Example: Modifying the mode of the schedule from `cron` to `repeatInterval` is prohibited. However, changing the value for the existing scheduling parameter is supported and results in an immediate re-scheduling.

`cron`	string	Crontab pattern for triggering the schedule. For more information, see Cron Format [page 11].

Parameter	Data Type	Description
`time`	string	For one-time schedules, this denotes the task execution time. You can use human-readable date formats to denote a specific time. Example: "3.30pm", "tomorrow at 2am". For information about human readable dates and the supported readable strings, see Human-Readable Date Formats [page 14]. You may also use date string in the ISO-8601 or IETF-compliant RFC 2822 format.
`repeatInterval`	string	Used to run schedules repeatedly at some interval. Human-readable texts must comply with the rules outlined for human readable dates. For information about the supported formats, see Human-Readable Date Formats [page 14].
`repeatAt`	string	For recurring schedules, this denotes the exact time when the job schedule must run. The human readable strings must comply with the rules outlined in the section Human Readable Dates. For information about human readable dates and the supported readable strings, see Human-Readable Date Formats [page 14].

Example

```
PUT /scheduler/jobs/3/schedules/0e29c67c-563e-4931-af08-43acb10813e8
{
  "description": "Edited Schedule that runs every 2 hours",
  "startTime": {
    "date": "2015-12-08 09:30:26.123"
  },
  "endTime": {
    "date": "2016-06-08 09:30:26.123"
  },
  "active": true,
  "repeatInterval": "2 hours"
}
```

Responses

Status Code: 200

The API call was successful and returns the updated schedule information.

Response Example

```
{
  "repeatInterval": "2 hours",
  "data": {
    "order_id": "abcd"
  },
  "description": "Edited Schedule that runs every 2 hours",
  "type": "recurring",
  "active": true,
```

```
  "startTime": "2015-12-08 04:00:26",
  "endTime": "2016-06-08 04:00:26",
  "jobId": 3,
  "scheduleId": "<schedule ID details>"
}
```

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.11 Delete Job Schedule

This API deletes the specified job schedule.

All related information such as job schedule configurations and logs are deleted. The processing of the schedule is also immediately terminated.

Routes

DELETE /scheduler/jobs/{jobId}/schedules/{scheduleId}

Request Parameters

Path

Parameter	Required	Data Type	Description
jobId	Yes	Integer	ID of the job
scheduleId	Yes	Integer	ID of the job schedule to be deleted

Example

```
DELETE /scheduler/jobs/3/schedules/0e29c67c-563e-4931-af08-43acb10813e8
```

Responses

Status Code: 200

The API call was successful and the job schedule was deleted.

Example

```
{
  "success": true
}
```

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.12 Activate or Deactivate All Job Schedules

This API activates or deactivates all the existing schedules for a job.

Routes

POST /scheduler/jobs/{jobId}/schedules/activationStatus

Request Parameters

Path

Parameter	Required	Data Type	Description
`jobId`	Yes	Integer	ID of the job whose schedules are to be activated/deactivated

Body

Parameter	Required	Data Type	Description
`activationStatus`	Yes	boolean	Requested activation status of the job schedules for the specified job. The values allowed are: • false: it indicates that all the schedules for the job need to be deactivated. • true: it indicates that all the schedules for the job need to be activated.

Example

```
POST /scheduler/jobs/3/schedules/activationStatus
{
  "activationStatus": false
}
```

Responses

Status Code: 200

The API call was successful and the activation status of all the schedules were updated.

Example

```
{
  "success": true
```

```
}
```

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.13 Delete All Job Schedules

This API deletes all the schedules of the specified job.

Routes

DELETE /scheduler/jobs/{jobId}/schedules

Request Parameters

Path

Parameter	Required	DataType	Description
`jobId`	Yes	Integer	ID of job whose schedules are to be deleted

Example

```
DELETE /scheduler/jobs/3/schedules
```

Responses

Status Code: 200

The API call was successful and the schedules were deleted.

Example

```
{
  "success": true
}
```

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.14 Retrieve Job Schedule Details

This API retrieves the saved configuration settings of a specified job schedule.

It retrieves the schedule logs if the `displayLogs` parameter is true.

Routes

GET /scheduler/jobs/{jobId}/schedules/{scheduleId}

Request Parameters

Path

Parameter	Required	Data Type	Description
`jobId`	Yes	Integer	ID of the job to which scheduleId belongs
`scheduleId`	Yes	string	ID of the schedule whose details are to be retrieved

Body

Parameter	Required	Data Type	Description
`displayLogs`	Yes	boolean	Controls whether the API should return all the generated logs for the schedule. Allowed value is true or false.

Example

```
GET /scheduler/jobs/3/schedules/cb5c9def-e2a0-4294-8a51-61e4db373f99?
displayLogs=true
```

Responses

Status Code: 200

The call was successful and returns the details of the schedule (and its logs, based on the request parameter).

Example

```
{
   "scheduleId": "<schedule ID details>",
    "description": "this schedule runs every 1 hour",
    "data": "{\"salesOrderId\":\"1234\"}",
    "type": "recurring",
    "cron": "* * * * */1 0 0",
    "active": false,
    "startTime": "2015-10-20 04:30:00",
    "endTime": null,
    "nextRunAt": "2017-08-11 10:00:00",
    "logs": [
        {
            "runId": "7841efdb-7ee0-48a6-a8a5-cbcafa1035b9",
            "httpStatus": null,
            "executionTimestamp": null,
            "runStatus": "SCHEDULED",
            "runState": "SCHEDULED",
```

```
            "statusMessage": "The job has been scheduled for a future run",
            "scheduleTimestamp": "2017-08-10 11:00:00",
            "completionTimestamp": null,
            "runText": "[{\"time\":\"2017-08-10 11:00:00\",\"type\":\"SCHEDULED
\",\"text\":\"\",\"code\":null}]",
            "locale": "en"
        },
        {
            "runId": "a68b86cf-1944-4e78-9cde-889e6bb713b3",
            "httpStatus": 404,
            "executionTimestamp": "2017-08-10 11:00:00", //indicates when
actually the scheduler invoked action endpoint
            "runStatus": "COMPLETED",
            "runState": "REQUEST_ERROR",
            "statusMessage": "Error encountered while sending job execution
request to the endpoint",
            "scheduleTimestamp": "2017-08-10 10:00:00", //indicates when the
schedule was picked up for calculation of next-run
            "completionTimestamp": "2017-08-10 11:00:00", //indicates when the
scheduler received response from the action endpoint
            "runText": "[{\"time\":\"2017-08-10 10:00:00\",\"type\":\"SCHEDULED
\",\"text\":\"\",\"code\":null},{\"time\":\"2017-08-10 11:00:00\",\"type\":
\"TRIGGERED\",\"text\":\"\",\"code\":null},{\"time\":\"2017-08-10 11:00:00\",
\"type\":\"REQUEST_ERROR\",\"text\":\"Response: Cannot PUT /health_status\\n\",
\"code\":404}]",
            "locale": "en"
        }
    ]
}
```

Status Code: 404

Passing invalid Job ID.

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule [page 60]
Retrieve Jobs [page 62]

6.1.15 Retrieve Job Schedule

This API retrieves schedule details for a specified job.

The API supports pagination, if required parameters are set. If these parameters are not provided, all the schedules for the job are fetched.

Routes

GET /scheduler/jobs/{jobId}/schedules

Request Parameters

Path

Parameter	Required	Data Type	Description
jobId	Yes	Integer	ID of the job whose schedules has to be retrieved

Body

Parameter	Required	Data Type	Description
`page_size`	No	number	Number of job schedules that the API retrieves. The default number is 10. If `page_size` is not set, and only `offset` is specified, the `page_size` parameter will take a default value of 10 records.
`offset`	No	number	Number of job schedule records to skip. The default number is 1. The default `offset` is an indicator to fetch results without skipping any records.

Example

```
GET /scheduler/jobs/1/schedules?page_size=10&offset=1
```

Responses

Status Code: 200

The API call was successful and the schedules were retrieved.

Example

```
{
                    {
```

```
"total": 2,
"results": [
{
"scheduleId": "52cd418d-4a13-45fb-a9b7-b2b15f546d5f",
"description": "updated scheudle",
"data": "{\"order_id\":\"1234\"}",
"type": "recurring",
"cron": "* * * * 10 0 0",
"repeatInterval": null,
"repeatAt": null,
"active": true,
"startTime": "2015-10-20 00:00:00",
"endTime": null,
"time": null,
"nextRunAt": "2017-08-11 10:00:00"
}
],
"prev_url": "/scheduler/jobs/27/schedules?page_size=1",
"next_url": null
}
```


Status Code: 404

Passing invalid Job ID.

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Jobs [page 62]

6.1.16 Retrieve Jobs

Retrieve all jobs in a service instance.

> **i** Note
>
> This API supports pagination if you've set the required parameters. If you haven't set the required parameters, all jobs are fetched.

Routes

```
GET /scheduler/jobs
```

Request Parameters

Body

Body of Retrieve Jobs API

Parameter	Required	Data Type	Description
`page_size`	No	`number`	Number of job schedules the API retrieves. The default number is `10`. If you haven't set `page_size` and only `offset` is specified, the `page_size` parameter takes the default value of `10` records.
`offset`	No	`number`	Number of job schedule records to skip. The default number is `1`. The default offset is an indicator to fetch results without skipping any records.

Parameter	Required	Data Type	Description
`tenantId`	No	`string`	Tenant ID upon which the job lists are filtered. This parameter applies only if the API is invoked using an access token obtained on behalf of a PaaS tenant from XSUAA.

Example

```
GET /scheduler/jobs?page_size=10&offset=1&tenantId=8a324eec-94f1-43ae-
b24a-0430aa157e14
```

Responses

Status Code: 200

The API call was successful and the schedules were retrieved.

Example:

```
{
  "total": 5,
  "results": [
    {
      "jobId": 42,
      "name": "Example_Job",
      "description": "Example_Description",
      "action": "https://exampleapi.cfapps.sap.hana.ondemand.com/api",
      "active": true,
      "httpMethod": "GET",
      "user": null,
      "startTime": null,
      "endTime": null,
      "signatureVersion": 0,
      "jobType": "HTTP_ENDPOINT",
      "subDomain": null,
      "createdAt": "1970-01-01 12:55:55",
      "ACTIVECOUNT": 0,
      "INACTIVECOUNT": 2
    }
  ],
  "prev_url": null,
  "next_url": "/scheduler/jobs/?page_size=1&offset=1"
}
```

Status Code: 400

The API was unable to process the request due to invalid data provided.

Parent topic: SAP Job Scheduling Service REST APIs [page 26]

Related Information

Create Job [page 30]
Configure Job (Using ID) [page 34]
Configure Job (Using Name) [page 36]
Delete Job [page 38]
Update Job Run Log [page 40]
Retrieve Job Details [page 41]
Retrieve Job Run Logs [page 43]
Retrieve Job Run Log Details [page 46]
Create Job Schedule [page 47]
Configure Job Schedule [page 50]
Delete Job Schedule [page 53]
Activate or Deactivate All Job Schedules [page 55]
Delete All Job Schedules [page 56]
Retrieve Job Schedule Details [page 57]
Retrieve Job Schedule [page 60]

6.2 Manage Jobs with Service Dashboard

The SAP Job Scheduling service dashboard enables you to manage jobs and tasks for a service instance.

Prerequisites

You have the **SpaceDeveloper** role that allows you to access the dashboard.

Context

The service dashboard lists the available jobs. Choose a job or a task to create a schedule or to view existing schedules. You can use the dashboard to perform the following tasks:

- Create a schedule
- Update a job/task or a schedule
- View action history for a job or schedule
- View run logs

The SAP Job Scheduling service supports the use of custom identity providers with platform users. The IdP provider can be used to access the service dashboard. For more information, see Supported Tools and Services When Using Custom Identity Providers for Platform Users[Feature Set A] .

The screens available on the dashboard and their description are as below:

Configurations: Edit global configurations required for a specific SAP Job Scheduling service instance to work.

- *Asynchronous Execution Timeout (s):* This value indicates how long (in seconds) the SAP Job Scheduling service awaits response for the asynchronous job from the action endpoint as well as from the CF task. If the application doesn't provide a response in the specified duration, or if the CF task isn't complete in the specified duration, the run status is set to COMPLETED/UNKNOWN. For CF tasks, the task is stopped when the timeout is reached.

Jobs/Tasks: View the jobs/tasks for a specific service instance. You can create a job/task and deactivate or delete an existing job/task.

- *Schedules:* Create and configure schedules for a job/task. To access schedules, choose a job/task listed on the dashboard. For more information, see Schedule Types in the related information section.
  Choose a schedule to view its history and logs. To display run logs of schedule, choose *Run Logs*.
- *Action History:* View the history or schedules of a job/task.

Procedure

Use one of the options to access the service dashboard:

Access the Dashboard	Actions
Using SAP BTP cockpit	1. From the navigation area of the SAP BTP cockpit, choose *Services* > *Instances and Subscriptions*. 2. Choose *View Dashboard* from the *Actions* menu for the relevant service instance. The service dashboard appears on a new tab. 3. In the navigation area, choose *Jobs*. 4. From the *Names* column, choose a job. This takes you to the *Overview* page. You can view the details for the selected job and edit it.
Using command line interface (CLI)	1. Enter the command to fetch service instance details `cf service <service-instance-name>`. 2. Copy the URL form the *Dashboard* field. 3. Paste the URL in a web browser of your choice. 4. Enter the SAP BTP credentials to log on.

Related Information

Schedule Types [page 9]

6.3 Node.js Client Library

Directly access Node.js APIs.

For jobs that support action endpoints (HTTP jobs), the SAP Job Scheduling service provides an API client library in the Node.js language. This library allows you to directly access Node.js APIs and reduce the code required to consume the service. See https://registry.npmjs.org .

Use the following code snippet for all API calls:

```
const JobSchedulerClient = require('@sap/jobs-client');
  const scheduler = new JobSchedulerClient.Scheduler();
  let myJob = { /* Refer to the SAP Job Scheduling
                                service documentation for sample JSON Object.
*/ };
```

The following table shows all SAP Job Scheduling service APIs together with a Node.js sample code to access the API.

Node.js Samples for SAP Job Scheduling service APIs

API	Sample Code to Access the API

Create Job [page 30]	`var scJob = { job: myJob };`
	`  scheduler.createJob(scJob, function (error, body) {`
	`  if (error) {`
	`  return logger.log('Error registering new job %s', error);`
	`  }`
	`  // Job successfully created.`
	`  job.id = body._id;`
	`  });`

Update Job	`var req = {`
	`  jobId: 33, /* Job ID for the job you need to update*/`
	`  job: {`
	`  user : 'John',`
	`  password : 'secret',`
	`  active : 1`
	`  }`
	`  };`
	`  scheduler.updateJob(req, function(err, result) {`
	`  if(err){`
	`  return logger.log('Error updating job: %s', err);`
	`  }`
	`  //job was updated successfully`
	`  });`

API	Sample Code to Access the API
Delete Job [page 38]	`var req = { jobId: 33 }; scheduler.deleteJob(req, function(err, result) { if(err){ return logger.log('Error deleting job: %s', err); } //job was deleted successfully });`
Get Job Details	`var req = { //by Id jobId: 33 }; scheduler.fetchJob(req, function(err, result) { if(err){ return logger.log('Error retrieving job: %s', err); } //job details retrieved successfully }); var req = { //by name name: 'my job' }; scheduler.fetchJob(req, function(err, result) { if(err){ return logger.log('Error retrieving job: %s', err); } //job details retrieved successfully });`
Create Job Schedule [page 47]	`var mySchedule = { /* according to SAP Job Scheduling service documentation */ } var req = { jobId: 33, schedule: mySchedule }; scheduler.createJobSchedule(req, function(err, result) { if(err){ return logger.log('Error creating job schedule: %s', err); } //Schedule created successfully });`

API	Sample Code to Access the API
Update Job Schedule	`var req = { jobId: 33, scheduleId: 'ABC-DEF', schedule: { cron: "* * * * 4" } }; scheduler.updateJobSchedule(req, function(err, result) { if(err){ return logger.log('Error updating job schedule: %s', err); } //Schedule updated successfully });`
Delete Job Schedule [page 53]	`var req = { jobId: 33, scheduleId: 'ABC-DEF' }; scheduler.deleteJobSchedule(req, function(err, result) { if(err){ return logger.log('Error deleting schedule: %s', err); } //Schedule deleted successfully });`
Get all Jobs	`var req = {}; scheduler.fetchAllJobs(req, function(err, result) { if(err){ return logger.log('Error retrieving jobs: %s', err); } //Jobs retrieved successfully });`
Get Job Schedule Details	`var req = { jobId: 33, scheduleId: 'ABC-DEF', displayLogs: false }; scheduler.fetchJobSchedule(req, function(err, result) { if(err){ return logger.log('Error retrieving schedule: %s', err); } //Schedule retrieved successfully });`

API	Sample Code to Access the API
Get Schedules of Job	`var req = { jobId: 33 }; scheduler.fetchJobSchedules(req, function(err, result) { if(err){ return logger.log('Error retrieving all schedules: %s', err); } //All schedules retrieved successfully });`
Update Run Log of Schedule	`var req = { jobId: 33, scheduleId: 'ABC-DEF', runId: 1, data: data }; scheduler.updateJobRunLog(req, function(err, result) { if(err){ return logger.log('Error updating run log: %s', err); } //Run log updated successfully });`
Get Run Logs of Schedule	`var req = { jobId: 33, scheduleId: 'ABC-DEF' }; scheduler.getRunLogs(req, function(err, result) { if(err){ return logger.log('Error retrieving run logs: %s', err); } //Run log retrieved successfully });`
Delete All Job Schedules [page 56]	`var req = { jobId: 3 }; scheduler.deleteAllJobSchedules(req, function(err, result) { if(err){ return logger.log('Error deleting schedules: %s', err); } //All schedules deleted successfully });`

API	Sample Code to Access the API
Bulk Activation of Schedules of Job	`var req = { jobId: 3 }; scheduler.activateAllSchedules(req, function(err, result) { if(err){ return logger.log('Error activating bulk schedules: %s', err); } //All schedules activated successfully });`
Bulk Deactivation of Schedules of Job	`var req = { jobId: 3 }; scheduler.deactivateAllSchedules(req, function(err, result) { if(err){ return logger.log('Error deactivating bulk schedules: %s', err); } //All schedules deactivated successfully });`
Get Action Logs of Job	`var req = { jobId: 3 }; scheduler.getJobActionLogs(req, function(err, result) { if(err){ return logger.log('Error retrieving action logs: %s', err); } //All actionlogs logs retrieved successfully });`
Get Action Logs of Schedule	`var req = { jobId: 3, scheduleId: "ABC-DEF" }; scheduler.getScheduleActionLogs(req, function(err, result) { if(err){ return logger.log('Error retrieving action logs: %s', err); } //All actionlogs logs retrieved successfully });`

API	Sample Code to Access the API
Get Active and Inactive Job Count	`var req = { activeStatus: true // true- for getting active number of jobs and false- for getting inactive number of jobs }; scheduler.getJobCount(req, function(err, result) { if(err){ return logger.log('Error retrieving jobcount: %s', err); } //Active Job count retrieved successfully });`

Related Information

SAP Job Scheduling Service REST APIs [page 26]

6.4 Service Behavior

This topic helps you understand and analyze the behavior of the SAP Job Scheduling service under specific situations.

Behavior of One-time Schedules

These are schedules that run only once. The behaviors of this schedule are described in the following table:

Behavior	Description
Schedule deactivated after a successful run or after an error	After a schedule run, whether the run was successful or had an error, the schedule action logs consider the event as DEACTIVATION_ON_COMPLETION. The corresponding job remains active.
Schedule executed on configured time	One-time schedules override the *startTime* configured at the job or schedule level.
Schedule executed immediately	If time is configured in the past and the *endTime* configured for the job or the schedule level is in the future, the schedule is executed immediately.
Schedule isn't executed	If time is configured in the past, and *endTime* configured at the job or the schedule level is also in the past, the schedule won't be executed.

Behavior of Cloud Foundry Tasks

If the creation of a task fails, check if the memory quota available for the space is sufficient.

Service Downtime and Schedule Runs

The objective of all cloud-based services is to provide availability all the time but downtime is possible. This could be due to issues with platform availability, network stability, planned downtime, and so on. During such occurrences, the service behaves as follows:

Schedules	Behavior
One-time	• Schedules that were already executed before the occurrence of the downtime won't be executed again when the service comes up. • Schedules that were scheduled to run during the occurrence of the downtime are executed when the service comes up.
Recurring	All the scheduled runs to be executed during the occurrence of the downtime are executed when the service comes up.

> **i** Note
>
> A scheduled job has a service level agreement of about 20 minutes from the scheduled time.

Action on Job Run Logs

The SAP Job Scheduling service will automatically remove run logs 15 days after they're generated. You can archive run logs before the clean-up starts by downloading them to your local system using any of the following options:

- Use the Retrieve Job Run Logs API.
- In the service dashboard, navigate to *Schedules* > *Logs* pane and choose the *Download* option.

Related Information

Retrieve Job Run Logs [page 43]

6.4.1 Deactivated Schedules

A scheduled job might have a missed run.

This can happen if the SAP Job Scheduling service fails to process the schedule before the scheduled run time and the schedule is deactivated during this time. The SAP Job Scheduling service considers that this deactivated schedule isn't required to run again.

7 Security

Get an overview of the security-relevant information that applies to the SAP Job Scheduling service.

For information on the security features of SAP BTP, see Security Guide for SAP BTP.

Identity and Access Management

This section describes how to secure action endpoints based on OAuth 2.0 authentication. For more information, see Secure Access [page 75].

Data Protection and Privacy

Governments place legal requirements on industry to protect data and privacy. We provide features and functions to help you meet these requirements.

> **i Note**
>
> SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and data protection-relevant functions, such as blocking and deletion of personal data. In many cases, compliance with applicable data protection and privacy laws is not covered by a product feature. Furthermore, this information should not be taken as advice or a recommendation regarding additional features that would be required in specific IT environments. Decisions related to data protection must be made on a case-by-case basis, taking into consideration the given system landscape and the applicable legal requirements. Definitions and other terms used in this documentation are not taken from a specific legal source.

For general information about data protection and privacy on SAP BTP, see the SAP BTP documentation under Data Protection and Privacy.

> **⚠ Caution**
>
> The SAP Job Scheduling service does not provide the technical capabilities to support the collection, processing, and storage of personal data.

> **⚠ Caution**
>
> This service contains entry fields, which are not intended for storing personal data without additional technical and organizational measure to safeguard data protection and privacy. For example, the job name, schedule description, or the message text send back in the response payload from the action endpoint of the application to the SAP Job Scheduling service are stored in the run log.

Related Information

SAP Authorization and Trust Management Service in the Cloud Foundry Environment
Secure Access [page 75]

7.1 Secure Access

The SAP Job Scheduling service provides options to secure job actions with action endpoints as well as to secure Cloud Foundry tasks.

Secure Job Actions with Action Endpoints

The SAP Authorization and Trust Management service is used to secure action endpoints in your application based on OAuth 2.0 authentication.

> **i** Note
>
> When you create or update a SAP Job Scheduling service instance, set the `enable-xsuaa-support` parameter as `true`.

OAuth 2.0 Authentication

Prerequisites: The application is bound to an xsuaa service instance and a SAP Job Scheduling service instance.

- SAP Job Scheduling service invokes the action endpoint, passing an access token obtained from UAA in the **Authorization** header.

  > **i** Note
  >
  > Within the application which exposes an action endpoint that is called from the SAP Job Scheduling service, you can validate this token using the xsuaa instance bound to this application.

- SAP Job Scheduling service caches the token for up to twelve hours.

  > **i** Note
  >
  > Consequently, it may take some time until the updates made to the scopes already granted to the SAP Job Scheduling service take effect, for example, if the application changes the scope name in the `xs-security.json` security descriptor file. During this time, it's possible that a token with the old scope is sent to the application. This may lead to an error, if the application doesn't accept the older scope.

- The application must grant a scope to the SAP Job Scheduling service instance.

For this, in the `xs-security.json` security descriptor file that is used from the xsuaa service instance that is bound to the application, specify the following value for the `grant-as-authority-to-apps` property:
`$XSSERVICENAME(<jobscheduler instance name>)`

> **i Note**
>
> `$XSAPPNAME.<name>`: A scope name consists of `$XSAPPNAME.` and a name you can choose, for example, `<Jobs>`.

Add the following to the `scopes` section:

```
{
    "xsappname": "<app name>",
    "scopes": [{
        "name": "$XSAPPNAME.Jobs",
        "description": "SAP Job Scheduling
                        service Scope",
        "grant-as-authority-to-apps": [
            "$XSSERVICENAME(<jobscheduler instance name>)"
            ]
    }]
}
```

> **i Note**
>
> SAP Job Scheduling service uses the content-type application/json for its requests and accepts the response `Content-type application/json`.

Secure Cloud Foundry Tasks

The XSUAA isn't required as Cloud controller. The Cloud Foundry User Account and Authentication (CFUAA) is used for creating CF tasks.

Related Information

Define and Grant Scopes to SAP Job Scheduling Service [page 77]
Web Access Control
Application Security Descriptor Configuration Syntax

7.1.1 Define and Grant Scopes to SAP Job Scheduling Service

Create and grant a scope to SAP Job Scheduling service.

Prerequisites

You've created an xsuaa service instance and bound it to the application.

For more information, see Getting Started [page 22].

Context

For the service plan *standard*, SAP Job Scheduling service uses User Account and Authentication (UAA) to facilitate the OAuth 2.0 authorization for job action endpoints. The action endpoint is secured via OAuth 2.0 scopes, hence, the application must grant a scope to the SAP Job Scheduling service.

> **i** Note
>
> The following steps provide an example.

Procedure

1. To grant scopes to SAP Job Scheduling service, open the `xs-security.json` file that is used for the xsuaa service instance bound to the application and add the following to the `scopes` section:

   > Sample Code
   >
   > Example `xs-security.json` file, **scope** section

   ```
   "scopes": [{
       "name": "$XSAPPNAME.JOBSCHEDULER",
       "description": "Job Scheduler Scope",
       "grant-as-authority-to-apps": ["$XSSERVICENAME(<jobscheduler instance
   name>)"]
       }]
   ```

2. Update the xsuaa service instance with the updated `xs-security.json`:

   ```
   cf update-service <xsuaa-instance-name> -c xs-security.json
   ```

Related Information

Update a Service Instance

8 Frequently Asked Questions

Find troubleshooting information in this section. We strongly recommend you use our **Guided Answers** for step-by-step solutions to some issues that may occur in your work with the SAP Job Scheduling service.

For more information, see Guided Answers.

General Questions

What is the component to raise a ticket for issues related to the SAP Job Scheduling service?

The support component for the SAP Job Scheduling service is **BC-XS-SRV-JBS**.

What are the prerequisites to use the SAP Job Scheduling service?

- You need access to the SAP Job Scheduling service. Check the Cloud Foundry marketplace to see if the service is available.
- For jobs with action endpoints, check if you have access and if you can create a service instance of the User Access and Authorization service (xsuaa). This service is available in the service marketplace.

What is the service level agreement (SLA) of a scheduled job?

A scheduled job has a service level agreement of about 20 minutes from the scheduled time.

As a subscription customer, what does the number of units selected on SAP Store indicate?

Currently, the number of units selected on SAP Store indicates the number of SAP Job Scheduling service instances that can be created.

> Example
>
> If you opt for a value of 5, it indicates that you can create 5 instances of the SAP Job Scheduling service. For each instance, you can create 10,000 job executions.

Service Dashboard

How to access the dashboard or the SAP Job Scheduling service?

You can use SAP BTP cockpit or the command line interface to access the dashboard or the SAP Job Scheduling service.

- 1. From the SAP BTP cockpit, choose your application.
  2. From the navigation panel, choose **Service Bindings**.
  3. Select the SAP Job Scheduling service instance and choose the  (Open Dashboard) icon from the *Actions* column. This launches the dashboard.

- From the command line interface
  1. Enter the command to fetch service instance details `cf service <service-instance-name>`.
  2. From the listed service instance details, copy the dashboard URL and paste the URL in a Web browser.
  3. Enter the credentials to log on to your SAP BTP account.

When I initially accessed the dashboard, I didn't grant the SAP Job Scheduling service any permissions. Now I get an error response (`500`). What can I do?

If your account is missing the authorization for a service, you can grant or revoke the access rights on the login route of the corresponding landscape:

```
https://login.<landscape.url.com>/profile
```

For eu10, for example, the login route looks as follows:

Sample Code

```
https://login.cf.eu10.sap.hana.ondemand.com/profile
```

For more information, see Regions.

How can I access my job logs from the dashboard?

From the dashboard, choose a job. This lists all the schedules for the job. Choose a schedule to view the logs.

Date-Time and Cron Formats

How to provide the cron format ?

SAP Job Scheduling service supports the SAP cron format. For more information, see Cron Format [page 11].

Does SAP Job Scheduling service support Linux Cron format?

No, SAP Job Scheduling service does not support the Linux cron format. Only the SAP cron format is supported.

Schedules and their Executions

Why is my schedule not getting triggered?

This can occur if the date-time format or the cron format is not appropriately specified or the specified endTime is in the past. Verify the schedule settings.

Why is my job not getting executed at the right time as per the specified format?

SAP Job Scheduling service runs jobs in the UTC time zone. Verify if the jobs are executed in the UTC time zone as per the format.

Why is my Job getting deactivated automatically?

This can happen if SAP Job Scheduling service is unable to reach an action endpoint. SAP Job Scheduling service makes multiple attempts to reach the action endpoint. If it consecutively fails in its attempts, it makes

the schedule inactive. The default number of attempts is 3. This value is set in the Maximum Invocation Attempts field available in the Configuration screen on the dashboard.

I receive run-log details with runStatus "Running" and runState "ACK_NOT_RECVD". Why does this happen?

This occurs in case of asynchronous jobs. SAP Job Scheduling service can access the action endpoint but the application does not update the status of the run log using Run Log Update API as expected. Thus, the job has the ACK_NOT_RECVD status.

To resolve the issue, update your application such that it notifies SAP Job Scheduling service after processing of the job is complete.

Multitenancy

General

What do you mean by multitenancy in the context of the SAP Job Scheduling service?

SAP Job Scheduling service invokes action endpoints of applications configured as job actions while registering the job.

A multitenant application in SAP BTP can register jobs on behalf of a tenant (or subaccount) who has subscribed to the application.

A multitenant application can retrieve information (tenant ID, subdomain, and so on) of the tenant (or subaccount) on behalf of which the application should execute the job when the SAP Job Scheduling service invokes the application's action endpoint.

Multitenancy in SAP Job Scheduling service enables multitenant applications in SAP BTP to register jobs on behalf of tenants (or subaccounts) who have subscribed to the application. It also enables the multitenant application to be aware of tenant information (tenant ID, subdomain, and so on) for whom the job has to be executed when the SAP Job Scheduling service invokes its action endpoint.

Platform as a Service Tenant

How can I register a job on behalf of a tenant (or subaccount) where my multitenant application is deployed?

The SAP Job Scheduling service instance receives the credentials required to register and manage jobs when the instance of the service is bound to a multitenant application.

To register a job on behalf of a tenant (subaccount), fetch an access token from the User Access and Authorization (xsuaa) service as follows:

- Fetch clientid, clientsecret, and URL from the UAA section of the SAP Job Scheduling service instance bound with your application
- Use Post /oauth/token
- Use body format: x-www-form-urlencoded
- Use parameters: client_id (as fetched from previous step)
- Use client_secret (as fetched from previous step)
- Use grant_type: client_credentials
- Use response_type: token

Using the access token received, invoke the SAP Job Scheduling service REST API to create a job. The access token has to be used in the header as **Authorization:Bearer**.

Such jobs will provide the tenant details of the multitenant application in the access token while invoking the configured job action.

How can I register a Job on behalf of a tenant, which has subscribed to my multitenant application?

A SAP Job Scheduling service instance bound to your multitenant application receives the credentials required to register and manage jobs. To register jobs on behalf of the tenant (subaccount) subscribed to your multitenant application, fetch an access token from the xsuaa service using the steps below:

Use the libraries provided by the xsuaa service.

Exchange the access token received from your logged user for a SAP Job Scheduling service token using the method **requestToken** and type as USER_TOKEN.

Refer library documentation for more specific details on how to obtain an access token using the user_token grant type. Using the access token received, invoke SAP Job Scheduling service REST API to create Job. The access token has to be used in header as authorization: bearer

The jobs thus created will provide the SaaS tenant (subaccount) details in the access token while invoking the configured job action.

How can I register a Job on behalf of a Tenant when a user is not logged on to my multitenant application?

If you want to create/delete/update/fetch a job on behalf of a SaaS tenant who has not logged on to your multitenant application, you can use client credentials flow with SaaS user subdomain. The xsuaa API to request a token with client credentials flow is mentioned at the following link: https://docs.cloudfoundry.org/api/uaa/version/4.23.0/index.html#client-credentials-grant

Note that you must use **jwt** as **token_format** (instead of **opaque**). You will receive a URL concatenated by **oauth/token** as described in the topic available at the link mentioned above. Further, remember that you need to replace the URL subdomain with the corresponding SaaS subdomain for which the token is requested. Using the access token that you receive, call the SAP Job Scheduling service REST API to Create Job. The access token has to be used in the header as **Authorization: Bearer**. Such created jobs will provide the SaaS tenant (or subaccount) details in the access token while calling the job action that you configured.

How can I fetch all Jobs or a specific Job (using id or name) registered in my SAP Job Scheduling service instance?

You can fetch all jobs registered in your service instance as client, client secret, and URL from the UAA section of the SAP Job Scheduling service instance bound with your application.

Use POST / oauth/token, body format: x-www-form-urlencoded, parameters: client_id (fetched from previous step), client_secret (fetched from previous step), grant_type is client_credentials, response_type as token.

Using the access token received, invoke SAP Job Scheduling service REST API to fetch all jobs using job ID or name. The access token has to be used in the header as authorization:bearer.

Can I filter on tenant while fetching Jobs registered for a Tenant?

Yes you can filter on tenant while fetching jobs registered for a tenant. If you have obtained an access token with grant type as client_credentials using the credentials from SAP Job Scheduling service instance binding, you can invoke the REST API to fetch all the jobs with tenantID as query parameter.

Example: GET https:///scheduler/jobs?tenantId=<tenantID>

This will retrieve all jobs registered on behalf of the tenant with tenant ID details. However, this query parameter based filter is not applicable if the token used belongs to a SaaS tenant and the API will return Status Code 400. This filtering is required to restrict access of a SaaS tenant to jobs created for another SaaS tenant.

Where do I find the tenant information when SAP Job Scheduling service invokes the action endpoint of the multitenant application?

The tenant information is provided in the access token sent by the SAP Job Scheduling service in the request headers while invoking the action endpoint. You may use the convenient client libraries provided by xsuaa to parse the information out of the access token. You will receive information like tenant id, subdomain, grant type and so on.

Which jobs does the Service Dashboard show when I launch it for a service instance from SAP BTP cockpit?

The service dashboard shows the jobs created in your service instance along with its tenant ID. It will also list the tenant information in the overview page for all the jobs. You may also sort the jobs based on tenant ID by clicking on the table header.

Can I create tenant-specific jobs from Service Dashboard?

No. Creating a job from service dashboard will always create a job specific to the PaaS tenant. If you are trying to create a job specific for a SaaS tenant who has subscribed to your multitenant application, you need to exchange the Jason Web Token with XSUAA and fetch a token specific for SAP Job Scheduling service (using client_id and client_secret provided to your application by SAP Job Scheduling service instance binding).

Software as a Service Tenant

How can I fetch Jobs registered for a SaaS Tenant?

You can fetch all jobs registered for a SaaS tenant using the libraries provided by XSUAA service, exchange the access token received from the logged user for the SAP Job Scheduling service token using the method requestToken and type as USER_TOKEN.

Refer the API Client library documentation for more specific details on how to obtain an access token using user_token grant type. Using the access token received, invoke the SAP Job Scheduling service REST API to fetch all jobs. The access token has to be used in header as authorization: bearer.

Related Information

Guided Answers

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP