IBM SPSS Collaboration and Deployment Services
8.3

RESTful Process Management Service Developer's Guide

Note

Before using this information and the product it supports, read the information in “Notices” on page 37.

Product Information

This edition applies to version 8, release 3, modification 0 of IBM® SPSS® Collaboration and Deployment Services and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Process Management Service

Process Management Service overview

The Process Management Service allows a client to manage the artifacts associated with running jobs. The client can submit an existing job for processing, and retrieve the execution meta-data and results. In addition, schedules can be created and assigned to jobs, allowing execution at a future date or on a recurring basis.

Accessing the Process Management Service from a client

Access to the Process Management Service is through standard HTTP and HTTPS methods. A client initiates a request to a server; the server processes the requests and returns the appropriate responses.

The request and response are built around the concept of a resource and its data.

- "Specifying a request URL" on page 1. The base URL for accessing the service is the IP address of an enabled server.
- "Specifying request headers" on page 2. Certain requests may require information in the request header.
- "Specifying a request body" on page 3. For operations that transmit data to the server, specify a request body in JSON format.
- "Receiving response headers" on page 3. Requests return headers as part of the response.
- "Receiving a response body" on page 4. For a GET operation, the response body contains the requested information in the format of a JSON object or array of objects.

Specifying a request URL

The base URL consists of a server and port specification appended with the service location.

The base URL for accessing the Process Management Service is the following:

```
http://{server}:{port}/process/rest/job
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository

Send the base URL appended with `/getVersion` as a GET request to obtain the version information in your IBM SPSS Collaboration and Deployment Services Repository. Most of the remaining URL requests include the identifier for a job or job execution. The following sample URL requests illustrate typical use.

- `GET http://cdssvr:80/process/rest/job/getVersion`

  Gets version information about the Process Management Service available for your system.
- `GET http://cdssvr:80/process/rest/job/getExecutionDetails`

  Retrieves the metadata and results for a specified execution.
- `GET http://cdssvr:80/process/rest/job/getJobParameters`

  Retrieves the variable definitions for a specified job.
- `GET http://cdssvr:80/process/rest/job/getTimeBasedSchedules`

Retrieves all time-based schedules for a job.

- `POST http://cdssvr:80/process/rest/job/submitJob`

  Submits a designated job for processing, generating an execution from which the results can be obtained. Any job variables are processed using their default values.

- `POST http://cdssvr:80/process/rest/job/submitJobWithOptions`

  Submits a designated job for processing using specified options. Options include the following:

  – Values for any job variables
  – Whether or not notifications should be sent

- `POST http://cdssvr:80/process/rest/job/cancelExecution`

  Cancels the processing of an execution.

The URL encoding for your application server must be defined to support the characters used in your URL strings. For example, if a job name contains multi-byte characters and the application server URL encoding is defined as ISO-8859-1, the server returns an error indicating that the job cannot be found. Setting the URL encoding to UTF-8 eliminates this problem. For information about setting the URL encoding for a specific server, see your application server documentation.

Specifying request headers

Certain requests may require headers.

The following table identifies fields commonly used in request headers.

Table 1. Request header values

Name	Value
Accept	Indicates the payload type to return to the client. Valid values include `application/json`, `text/xml`, and `application/xml`.
Authorization	Authentication string encoded using Base64
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information

Accept

The *Accept* field defines the MIME type of the information contained in the body of the response returned to the client application.

Authorization

The Process Management Service includes a security layer to limit access to authorized users. This layer uses HTTP basic access authentication, requiring the specification of a valid user name and password to be included in any service request.

To add authentication information to a request, include the *Authorization* field in the request header. The value for this field consists of the following components:

- The string *Basic* to indicate basic access authentication is being used
- A space
- The Base64 encoding of the concatenation of the user name, a colon, and the password

For example, the Base64 encoding for a user name of *user* and a password of *pass* is the following:

```
Base64 Encode(user:pass) = dXNlcjpwYXNz
```

The *Authorization* value for this user would be specified as the following string:

```
Basic dXNlcjpwYXNz
```

Globalization

The Process Management Service can return responses containing globalized information. To specify a language for the response information, include either the `Client-Accept-Language` or the `Accept-Language` fields in the request header. For both fields, specify a language value in accordance with RFC 1766.

The `Client-Accept-Language` field is used by other components of IBM SPSS Collaboration and Deployment Services that interact with the Process Management Service. If your request includes both language fields, this field is evaluated before the `Accept-Language` field, which is the standard HTTP field for language specification.

For example, to create a response in German, include the following header field in your request:

```
Client-Accept-Language: de
```

If your request also includes the following header field:

```
Accept-Language: fr
```

The response will still be in German as the `Client-Accept-Language` field takes precedence. In this case, you would either need to change the value of `Client-Accept-Language` to `fr` or omit this field entirely to receive a response in French.

Specifying a request body

For operations that transmit data to the server, such as POST, specify a request body in JavaScript Object Notation, or JSON, format.

For information on JSON formatting, see "Receiving a response body" on page 4. For information about the specific JSON elements and objects used by the Process Management Service, see Chapter 4, "JSON reference for the Process Management Service ," on page 33.

This sample POST request goes to submit a designated job for processing using specified options. Options include the following:

- Values for any job variables
- Whether or not notifications should be sent

```
{
  "jobLocationURI": "spsscr:///jobA",
  "jobOptions": {
    "setNotificationEnabled": true,
    "jobParameterValue": [
      {
        "Name": "parameterName1",
        "Value": "parameterValue1"
      }
    ]
  }
}
```

Receiving response headers

Requests return headers as part of the response.

The following table lists response header values.

Table 2. Response header values for GET requests	
Name	**Value**
Status Code	*200 OK* or an error code
Server	The name of the server processing the request
X-Powered-By	Identifies the technology underlying the application
Content-Type	The MIME type of the content in the response body. For example, for JSON, the value could be *application/json;charset=ISO-8859-1*.
Content-Length	Length of the response body in bytes
Date	Current date and time
Set-Cookie	Specifies an HTTP session cookie

Receiving a response body

The response body contains the requested information in the format specified by the **Accept** field in the request header. For JSON, the body is typically an object or an array of objects.

The following sections provide basic formatting rules for JSON. These rules also apply to the request body.

See http://www.json.org for a complete description of JSON. For information about the specific JSON elements and objects used by the Process Management Service, see Chapter 4, "JSON reference for the Process Management Service ," on page 33.

Elements

An element consists of a name-value pair having the following structure:

```
"<name>":<value>
```

- *<name>* is a string that identifies the element
- *<value>* corresponds to the value for the specific element. The value may be a boolean, string, number, object, or array.

String values must be enclosed in quotation marks and may contain backslash escape characters. Numeric values are not enclosed in quotation marks. A boolean value is either true or false, and is not enclosed in quotation marks.

For example, the following *label* element corresponds to a string value of *LATEST*:

```
"label":"LATEST"
```

The following *value* element corresponds to a numeric value of *38.0*:

```
"value":38.0
```

The following *isRequired* element corresponds to a boolean value of *true*:

```
"isRequired":true
```

Objects

An object represents an unordered collection of elements. Objects have the following structure:

```
{<element1>, ... ,<elementN>}
```

- *<element#>* corresponds to a particular element in the object. Individual object elements are separated by commas.

Forexample, the following `ExecutionDetails` object consists the elements of `executionState`, `executionSuccess`, `endDateTime`, and `uuid`:

```
{
   "uuid": "006fcff6564bb75b00000178ef458c5fe737",
   "artifactLocation": [],
   "log": null,
   "executionState": "ENDED",
   "queuedDateTime": "2021-04-23 02:24:18",
   "startDateTime": "2021-04-23 02:24:18",
   "endDateTime": "2021-04-23 02:24:19",
   "completionCode": 0,
   "executionSuccess": true,
   "eventUuid": "006fcff6564bb75b00000178ef458c5fe708",
   "eventName": "general",
   "notificationEnabled": true,
   "executionWarning": false,
   "userName": "admin",
   "hasIterations": false
}
```

Arrays

An array represents an ordered collection of values, elements, objects, or other arrays. Arrays have the following structure:

```
[<entry1>, ... ,<entryN>]
```

- <entry#> corresponds to a particular entry in the array. Individual array entries are separated by commas.

Forexample, the value for the following `jobStepExecutions` element is an array containing two objects of `executionDetails`:

```
[
      {
      "uuid": "006fcff6564bb75b00000178ef458c5fe73a",
      "artifactLocation": [],
      "log":       {
         "logAsString": "--- PRMS ---\ncd\nThe process terminated with the exit code 0.\n--- STDOUT ---\n--- STDERR ---\n",
         "logAsAttachment": null
      },
      "executionState": "ENDED",
      "queuedDateTime": 1619169858000,
      "startDateTime": 1619169858000,
      "endDateTime": 1619169858000,
      "completionCode": 0,
      "executionSuccess": true,
      "eventUuid": "006fcff6564bb75b00000178ef458c5fe709",
      "eventName": "Event 1",
      "notificationEnabled": true,
      "executionWarning": false,
      "userName": null,
      "hasIterations": false
   },
      {
      "uuid": "006fcff6564bb75b00000178ef458c5fe739",
      "artifactLocation": [],
      "log":       {
         "logAsString": "--- PRMS ---\npwd\nThe process terminated with the exit code 0.\n--- STDOUT ---\n/appserv/WebSphere9/
profiles/AppSrv01/pasw_execution_root/006fcff6564bb75b00000178ef458c5fe740\n--- STDERR ---\n",
         "logAsAttachment": null
      },
      "executionState": "ENDED",
      "queuedDateTime": 1619169858000,
      "startDateTime": 1619169858000,
      "endDateTime": 1619169859000,
      "completionCode": 0,
      "executionSuccess": true,
      "eventUuid": "006fcff6564bb75b00000178ef458c5fe70b",
      "eventName": "Event 2",
      "notificationEnabled": true,
      "executionWarning": false,
      "userName": null,
      "hasIterations": false
   }
]
```

Each object in the array consists of several elements having values describing individual output fields.

Processing a response body

The content of the JSON information depends on the parser that is used to generate the JSON data. As a result, you might encounter subtle differences in the JSON information that is transmitted by the service, particularly regarding null or missing values.

For example, one parser might return an empty array if an element has no values. In the following JSON, the **categoricalValues** element has no values.

```
"metadataInputField":[
   {
      "categoricalValues":[],
      "isRequired":<boolean>,
      "type":"<string>",
      "name":"<string>",
      "description":"<string>"
   }
]
```

A different parser might omit the element entirely if it has no values. In the following JSON, the **categoricalValues** element is absent from the object.

```
"metadataInputField":[
   {
      "isRequired":<boolean>,
      "type":"<string>",
      "name":"<string>",
      "description":"<string>"
   }
]
```

Alternatively, a parser might include a `null` value if an element has no values. In the following JSON, the **categoricalValues** element has a `null` value.

```
"metadataInputField":[
   {
      "categoricalValues":null,
      "isRequired":<boolean>,
      "type":"<string>",
      "name":"<string>",
      "description":"<string>"
   }
]
```

To ensure your application processes the response correctly, review how your parser behaves. To make your application more robust, consider defensive programming approaches to avoid problems with `null` or missing values in the JSON data.

Chapter 2. Process management concepts

Jobs

A job is a container for a set of work, or events, to be executed. Each individual piece of work in a job is commonly referred to as a job step. Submission of the job results in execution of the steps it contains. When a job executes, the system creates **executions** for the overall job and for the individual steps. The executions report on the job status and provide information about any output generated. The execution of a job can be defined to occur on a recurring basis using **schedules**.

For example, a job may consist of a sequence of steps that invoke IBM SPSS Statistics for initial data processing and graphical displays, followed by IBM SPSS Modeler steps for modeling and scoring. Submitting the job creates a set of executions in the system from which the results can be obtained. The job can be scheduled to run automatically every week to generate executions containing updated graphs, models, and scores using the most current data.

The IBM SPSS Collaboration and Deployment Services Repository provides the storage mechanism for jobs to be executed. The Process Management Service references a job by its content repository URI.

Content repository Uniform Resource Identifiers

Resources within the IBM SPSS Collaboration and Deployment Services Repository are often referenced using a uniform resource identifier. A content repository URI consists of the following items:

- The scheme *spsscr:*
- A hierarchical specification consisting of an authority definition and an optional object path
- An optional query specifying an object identifier
- Optional fragments defining version information

The URI has the following format:

```
spsscr://[host][:port]/[path/filename [?hierarchyType=type] | ?id=repositoryID][#l.label | #m.marker]
```

The hierarchical portion begins with two slashes, followed by the authority definition. This information identifies the host name and port number for the repository containing the object, followed by a slash. The authority definition may be omitted, in which case the URI indicates a relative location within the repository processing the service request.

```
spsscr:///[path/filename [?hierarchyType=type] | ?id=repositoryID][#l.label | #m.marker]
```

The URI continues with either the full path to the object, including its name, or a question mark and a query term consisting of the key *id*, an equals sign, and the repository resource identifier for the object. This identifier can be obtained from the information returned by the `getResource` operation of the Content Repository Service.

If the URI specifies an object path, the path may be followed by a query parameter designating the type of hierarchy containing the object. This parameter begins with a question mark, followed by the key *hierarchyType*, an equals sign, and the hierarchy type designator. Valid hierarchy types include *folder*, *topic*, *configuration*, *server*, *credential*, *datasource*, *enterprise*, and *submitted*. If the *hierarchyType* parameter is omitted, the *folder* hierarchy is used by default. The *hierarchyType* parameter is valid only when using the path to identify the object.

Optional version fragments follow the object information. The fragments begin with a hash symbol (#), followed by a single letter indicating whether the fragment is a version label (`l`) or a version timestamp

marker (m). The fragment ends with a period and the actual label or marker for the version. Replace any spaces in the label or marker with escape characters. For example, the URI:

```
spsscr://myserver:80/marketing/campaign1#m.0:2006-10-08%2012:34:10.223
```

refers to the version of the *campaign1* job in the *marketing* folder saved at 12:34 on October 8, 2006. A URI that does not include a version fragment references the latest version of the object. For instance, the URI:

```
spsscr://localhost/campaign2
```

refers to the latest version of the job *campaign2*.

Job variables

Job variables define parameters whose values can be passed to any step within the job. Using variables, any job can be used as an iterative consumer, in which values external to the job can be used to control job processing. Values for the variables can be defined:

- When initiating the job
- In schedules associated with the job
- In other jobs executing before the job

Each job variable can be characterized by the following properties:

- **Variable Name.** The name of the variable defined for the job.
- **Default Value.** The default value for a job variable. If a variable has no specified default value and a value has not been assigned in some other fashion, the user is prompted for a value during job execution.
- **Description.** Informative text about a variable typically used to aid in identifying the variable.

The Process Management Service includes operations for retrieving variable definitions and for specifying variable values during job submission.

Schedules

Schedules provide a triggering mechanism to initiate job execution. Currently, a job is the only type of object that can be scheduled, using triggers based on either time or JMS messages. A job that is not associated with a schedule must be executed manually.

The Process Management Service includes operations for creating, retrieving, updating, and deleting schedules.

Time-based schedules

A time-based schedule for a job designates a time, date, and optional recurrence pattern for execution of the job. When the specified date/time is reached, the schedule activates, causing any associated jobs to execute. Time-based schedules that can be assigned to jobs include the following:

- **Once.** Executes a job a single time at a specified date/time.
- **Minutely.** Executes a job on a minute basis. The recurrence pattern indicates the number of minutes between executions. For example, set the interval to 10 to execute the job every 10 minutes.
- **Hourly.** Executes a job on an hourly basis. The recurrence pattern indicates the number of hours between executions. For example, set the interval to 1 to execute the job every hour. To execute the job every other hour, set the interval to 2.
- **Daily.** Executes a job at a specified time of day. The recurrence pattern indicates the daily interval between executions. For example, set the interval to 1 to execute the job every day. For execution every other day, set the interval to 2.

- **Weekly.** Executes a job at a specified time of a designated weekday. The recurrence pattern indicates the weekly interval between executions. For example, set the interval to 1 to execute the job on the specified weekday every week. To execute the job every other week, set the interval to 2.
- **Monthly.** Executes a job at a specified time on a designated day of the month. The recurrence pattern indicates the monthly interval between executions. For example, set the interval to 1 to execute the job on the specified day of every month. For execution every other month, set the interval to 2. To execute the job quarterly, use an interval of 3.
- **Yearly.** Executes a job at a specified time, day, and month combination. The recurrence pattern indicates the yearly interval between executions. For example, set the interval to 1 to execute the job every year. To execute the job every other year, set the interval to 2.

Time-based schedules can be defined to end on a specific date. In the absence of an end date, the schedule will obey its recurrence pattern indefinitely.

Message-based schedules

Message-based schedules allow external applications to initiate execution of jobs and job steps stored within the IBM SPSS Collaboration and Deployment Services Repository using the Java Message Service (JMS). JMS allows two primary approaches to messaging, point-to-point and publish-subscribe. In the former, a single message producer sends a message to a single message consumer using a queue. In the latter, one or more producers send messages to one or more consumers using topics. IBM SPSS Collaboration and Deployment Services employs the publish-subscribe model for messaging using the JMS server capabilities of the application server hosting the IBM SPSS Collaboration and Deployment Services Repository.

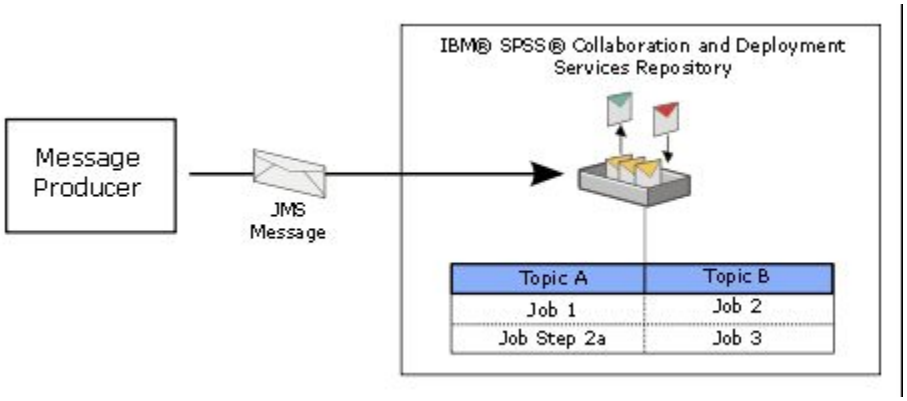

Figure 1. IBM SPSS Collaboration and Deployment Services messaging

A message producer publishes a message to a defined topic in the JMS server. Message consumers, or subscribers, that have subscribed to the topic receive the message and respond accordingly. In IBM SPSS Collaboration and Deployment Services, jobs and job steps correspond to the consumers and execution is the response to a message received. A message-based schedule for a job identifies a message destination to monitor for activity. When a message is received at that destination, the schedule activates, causing any job associated with the schedule to execute. For example, in the figure, if the producer sends a message to topic A, the subscribers Job 1 and Job Step 2a execute. Job 2 and Job 3 only execute when a message is sent to topic B.

Subscribing a consumer to a message involves associating the consumer with a message domain. The message domain defines the properties associated with destinations for JMS messaging, such as the following:

- **Destination Name.** The name of the topic or queue.
- **Naming Factory.** Application server specific string designating the JMS Java class. For example, for JBoss application server the naming factory is org.jnp.interfaces.NamingContextFactory.
- **Naming Service.** The URI location of the naming service. For example, for JBoss application server the naming service is `jnp://localhost:1099`.

- **Credentials.** Optional credentials. The credentials will only be required in certain instances, depending upon the configuration of the JMS server and how we are required to connect to the JMS server (whether the JMS Message topic is secured or not).

Any message sent to the destination defined for a domain will trigger a job associated with the domain. To control which subscribed jobs execute, use message filters. To create a message domain, use the Content Repository Service.

Subscriptions to message domains can be either nondurable or durable. Nondurable subscriptions will not receive any messages sent by producers while IBM SPSS Collaboration and Deployment Services Repository is not running. Those messages are effectively lost. In contrast, for a durable subscription, the JMS server saves the messages until the consuming application is running and the messages can be delivered.

JMS message structure

A JMS message consists of header information, optional properties, and the message body. The header for a message includes information used for identification and routing. Standard header fields for JMS messages include the following:

- *JMSDestination*. The destination for the message.
- *JMSDeliveryMode*. The mode for delivery specified by the message producer.
- *JMSExpiration*. The message expiration time.
- *JMSPriority*. A number from 0 to 9 indicating the priority associated with the message. Higher numbers represent higher priorities, with 4 being the default priority value.
- *JMSMessageID*. A unique identifier for the message.
- *JMSTimestamp*. The time the message was available for sending.
- *JMSCorrelationID*. Allows linking of messages by specifying a related message identifier or string.
- *JMSReplyTo*. The location to which to send any responses to the message.
- *JMSType*. A string indicating the message content type.
- *JMSRedelivered*. Indicates whether or not the message was resent.

Message properties are optional header fields often classified into three categories. **Application properties** are any custom properties created by an application and usually only have meaning to the applications sending and receiving the messages. **Provider properties** correspond to proprietary fields for specific vendors offering JMS producers. Finally, **standard properties** are optional fields defined by the JMS specification but which may not be supported by all message producers and consumers. Standard properties include the following:

- *JMSXUserID*. String identifying the user sending the message.
- *JMSXAppID*. String identifying the application sending the message.
- *JMSXDeliveryCount*. Integer indicating the number of times message delivery has been attempted.
- *JMSXGroupID*. String identifying a message group for the message.
- *JMSXGroupSeq*. Integer indicating the sequential position of the message in its group.
- *JMSXProducerTXID*. String identifying the producer transaction that created the message.
- *JMSXConsumerTXID*. String identifying the consumer transaction that received the message.
- *JMSXRcvTimestamp*. The time the message was delivered to the consumer.
- *JMSXState*. Integer indicating the state of the message as waiting (1), ready (2), expired (3) or retained (4).

The message body contains the payload as one of the following types:

- **Stream**. A stream of Java primitive values
- **Map**. A set of name-value pairs.
- **Text**. A string.

- **Object**. A serializable Java object.
- **Bytes**. A stream consisting of uninterpreted bytes.

Example JMS Message

An example message with a text body follows:

```
Header {
   jmsDestination  : TOPIC.testTopic
   jmsDeliveryMode : 2
   jmsExpiration   : 0
   jmsPriority     : 4
   jmsMessageID    : ID:33-11903898478381
   jmsTimeStamp    : 1190389847838
   jmsCorrelationID: null
   jmsReplyTo      : null
   jmsType         : null
   jmsRedelivered  : false
   jmsProperties   : { TaskType=ETL, TaskStatus=complete}
   jmsPropReadWrite: false
   msgReadOnly     : true
   producerClientId: ID:33
}
Body {
   text            :ETL complete
}
```

For a complete discussion of the content of JMS messages, see the JMS Specification .

Message filters

All messages sent to a topic get passed to all consumers subscribed to that topic. This implies that every job or job step subscribed to a message domain will execute when any message is sent to that domain. However, it may be desirable to use the message properties or content to dictate which subscribers should activate. For example, job *Run_Report* executes if the message indicates success and job *Notify_Admin* executes for a failure message. This selective execution of message-based jobs can be accomplished using message filters for subscribers. IBM SPSS Collaboration and Deployment Services offers text and selector filters.

A text filter specifies text that must be contained within the body of a JMS text message for subsequent triggering of subscribers. For example, subscribers specifying a text filter of *success* would only activate if a message containing the string *success* in its text body was received.

A selector filter defines a conditional expression that must be met for subscriber triggering. The expression is based on the message header field and property values using a subset of SQL-92 syntax. For example, if a message contains the property *TaskStatus*, a selector for a property value of *complete* is:

```
TaskStatus='complete'
```

For any subscriber, text and selector filters can be used in conjunction to restrict execution to messages that contain specific text and also have specific field and property values.

Executions

An execution, or **event execution**, contains the results of executing an event. Every time an event executes, a new execution is created. The execution includes the following:

- The date and time when the event began execution
- The date and time when the event completed execution
- The completion code
- An indicator of whether the event succeeded or not
- Artifacts produced by the execution
- A log detailing the execution

Use the execution to access the results for an event.

The Process Management Service includes operations for retrieving and deleting executions for an event. In addition, you can cancel the execution for a specified event.

Queries

The Process Management Service provides a query mechanism for retrieving schedules and executions from the IBM SPSS Collaboration and Deployment Services Repository that meet specified criteria. The information contained in the search result set can be customized to be as broad or focused as needed. For example, the query can return all available execution information for all jobs in the system, or for all jobs having a specific label that have failed. In addition, large result sets can be returned as individual pages containing a specified number of hits to optimize client performance.

Queries return information as field/value pairs. These fields can generally be classified into three categories: general, execution, and schedule. General fields indicate properties of the file or job associated with the returned schedule or execution.

Table 3. General fields

Field Name	Description
`file.objid`	Identifier for the file associated with the schedule or execution
`job.objid`	Identifier for the job associated with the schedule or execution
`file.name`	Name of the file
`file.path`	Path to the associated file in the repository
`jobpath`	Path to the associated job in the repository
`fileversion.marker`	The timestamp for the file or job
`fileversion.label`	The label for the file or job, if any
`fileversion.expirationDate`	Expiration date of the file
`usesfiledep.label`	The label, if any, for the file or job

Execution fields report information about job executions. These properties include the execution time and state.

Table 4. Execution fields

Field Name	Description
`eventexe.objId`	Identifier for the execution object
`eventexe.mode`	Method for initiating the execution
`eventexe.state`	State of the execution at the time of the query
`eventexe.executionSuccess`	Boolean indicating whether or not the execution succeeded
`eventexe.completionCode`	Value reflecting the state of the execution at completion
`eventexe.startDateTime`	Date and time that the execution began
`eventexe.endDateTime`	Date and time that the execution ended
`eventexe.queuedDateTime`	Date and time for the next execution
`eventexe.userName`	User under which the execution occurs

Table 4. Execution fields (continued)	
Field Name	**Description**
executionRuntime	Number of seconds to complete the job execution
scheduleEventExecution.state	State of the time-based schedule execution at the time of the query
scheduleEventExecution.executionSuccess	Boolean indicating whether or not the time-based execution succeeded
scheduleEventExecution.startDateTime	Date and time that the time-based execution began
scheduleEventExecution.endDateTime	Date and time that the time-based execution ended
scheduleEventExecution.queuedDateTime	Date and time for the next time-based execution
msgEventExecution.state	State of the message-based execution at the time of the query
msgEventExecution.executionSuccess	Boolean indicating whether or not the message-based execution succeeded
msgEventExecution.startDateTime	Date and time that the message-based execution began
msgEventExecution.endDateTime	Date and time that the message-based execution ended
msgEventExecution.queuedDateTime	Date and time for the next message-based execution

Time-based schedule fields report information about job schedules. These properties include the schedule frequency and start date.

Table 5. Time-based schedule fields	
Field Name	**Description**
schedule.objId	Identifier for the schedule
schedule.frequency	Frequency
schedule.interval	Recurrence interval
schedule.month	The month in the year in which to run
schedule.dayOfMonth	The day in the month in which to run
schedule.daysOfWeek	Days of the week in which to run
schedule.timeOfDay	Time at which the schedule triggers
schedule.scheduleStartDate	Start date
schedule.scheduleEndDate	Date at which the schedule terminates
schedule.nextScheduleTime	Time of the next schedule triggering
schedule.scheduleEnabled	Indicator of whether or not the schedule is enabled

Message-based job fields report information about jobs initiated by messages. These properties include the message filters and domain information.

Table 6. Message-based job fields	
Field Name	**Description**
`msgDrivenJob.objid`	Identifier of a message-driven job
`msgDrivenJob.messageSelector`	Conditional expression that must be satisfied for the subscription to activate
`msgDrivenJob.messageText`	Filter text that must be matched for the subscription to activate
`messageDomain.name`	Name of the message domain
`messageDomain.destinationName`	Name of the message destination
`messageDomain.namingService`	URI location of the naming service
`messageDomain.namingFactory`	String denoting the JMS Java class
`trigger.summary`	For scheduled jobs, this is the frequency of the schedule (e.g. once, daily, weekly, monthly). For message-driven jobs, this is the message domain.
`next.trigger`	For scheduled jobs, this is the next time the schedule will run. For message-driven jobs, this is the selector and message text.
`lastrun.startDateTime`	The queuedDateTime attribute for the last run of the job
`lastrun.status`	An indicator of the job success and state
`credentials.name`	Name of the credentials under which the job runs

Whether querying for executions or for schedules, the criterion for the query must be specified. A criterion consists of a return specifier and a page selector. The former identifies the structure of the information returned by the query. The latter specifies any filtering criteria that must be satisfied by matching objects, also known as **hits**, as well as paging requirements.

Return specifiers

The return specifier for a query defines the following criteria:

- **Sort column**. The name of the field on which to sort the returned information. For a list of available field names, see “Queries” on page 12.
- **Sort order**. Whether the results should be sorted in ascending or descending order.
- **Page size**. The maximum number of matching objects to return.

The return specifier criteria are optional. If absent, the system uses defaults.

Page selectors

A query criterion can include a **page selector** to reduce the number of returned objects. A page selector allows both filtering and paging of result sets.

A selector **filter** restricts the returned set to objects having a specified value for a designated property. Available filters include the following:

- **Version label.** Restricts the results to objects having a specified version label.
- **Execution state.** Returns only executions having a designated state. Valid state values for this filter include *dummy*, *initializing*, *queued*, *running*, *ended*, *cascading*, *error*, *cascade_error*, *canceling*, *canceled*, *cancel_pending_cascade*, *cascade_canceled*, *joining*, and *never_joined*.

- **Completion status.** Limits the results to executions having a specified status. Valid status values for this filter include *success* and *failure*.
- **Job URI.** Limits the results to a specific job.
- **Date range.** Limits the results to those occurring within a specified date/time range.
- **Execution mode.** Limits the results to executions resulting from schedule triggers, manual execution, or submission.

A **page request** for a selector indicates the row of the result set at which to begin returning results. In addition to the row number, the request includes a key identifying the result set being accessed. The key allows clients to page through the results. For example, a client application could display ten query results in a single display. The 11th through the 20th results could be returned by initiating a query involving a page request beginning at row eleven using the same key as the first set of results.

Page results

The Process Management Service returns query results in a table containing matching objects in the rows. The columns correspond to the field values. The table illustrates the structure for a query returning two executions of the same job. Some columns are omitted in the interest of space.

Table 7. Example Execution Query Results

Job ID	Job Marker	State	Completion Code	Start	End
0A0B32E0CFC42076000 0010EC3B57CAF81C3	1:2006-11-10 12:37:08.385	4	2	2006-11-10 12:37:16.927	2006-11-10 12:40:55.007
0A0B32E0CFC42076000 0010EC3B57CAF81C3	2:2006-11-10 13:11:04.257	4	2	2006-11-10 13:11:09.947	2006-11-10 13:13:46.707

Page results information can be classified into four categories: general metadata, column, row, and navigator. General metadata includes the following:

- The total number of hits in the result set
- The maximum number of hits for any page
- The page number for the current page
- The column used for sorting the hits
- The sort order as *ascending* or *descending*
- A client key value, which is an internal identifier used to synchronize requests for specific pages

Information for columns consists of the following:

- The display name for the column, such as *Title* or *Author*
- The internal field name for the column.
- An indicator of the type of information reported in the column. The type is *string* for all fields.

Row information returned by the Process Management Service includes the following:

- The row number
- The values for the individual cells in an order corresponding to the order of the columns. For example, the second value for each hit in the Table 7 on page 15 table would be the value for the second column, *Job Marker*.

Navigators serve to facilitate the creation of user interfaces to display the results for multiple pages. This information consists of characteristics of each page in the results, as well as data for the preceding and following pages.

Chapter 3. API reference for the Process Management Service

getVersion API

Returns the version number of the service.

Request

```
GET http://{server}:{port}/process/rest/job/getVersion
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository

Table 8. Request header values

Name	Value
Accept	Indicates the payload type to return to the client.Valid values include `application/json`,`text/xml`,and `application/xml`.
Authorization	Authentication string encoded using Base64
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information

The HTTP request body should be empty.

Response

A successful request returns the following information:

- Statuscode 200
- Response body is in the plain text format containing the version number of Process Management Service

An unsuccessful request returns a response code other than 200.

Example

This request retrieves information about the Process Management Service.

```
GET http://cdssvr:80/process/rest/job/getVersion
```

This is the JSON response object, indicating that there are two scoring providers installed.

```
8.3.0.0.x
```

The response indicates that the Process Management Service is started and its version is 8.3.0.0.x.

submitJob API

Submits a designated job for processing, generating an execution from which the results can be obtained. Any job variables are processed using their default values.

Request

```
POST http://{server}:{port}/process/rest/job/submitJob
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository

Table 9. Request header values

Name	Value
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client.Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accep-Language	Tag defining the language for the response information.

The HTTP request body should contain the input data for submiting job. A JSON request object has the following general structure:

```
{
  "jobLocationURI": "string",
  "notificationEnabled": true/false
}
```

`jobLocationURI` must have the prefix `spsscr://` appended with the folder name and job name (`/folderA/jobA`). For example: `spsscr:///folderA/jobA`

Response

A successful request returns the following information:

- Statuscode 200
- Response body containing the job submission result for the request, like:

```
Job is submitted successfully and execution ID is xxxx
```

An unsuccessful request returns a response code other than 200.

Example

This request submits a job:

```
POST http://cdssvr:80/process/rest/job/submitJob
```

This is the JSON response object, indicating that the specified job is submitted:

```
Job is submitted successfully and execution ID is 006fcff6153b1e79000001791502452ef9f9
```

submitJobWithOptions API

Submits a designated job for processing using specified options. Options include the following:

- Values for any job variables
- Whether or not notifications should be sent

Request

```
POST http://{server}:{port}/process/rest/job/submitJobWithOptions
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository

Table 10. Request header values

Name	Value
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

The HTTP request body should contain the input data for submitting a job with options. A JSON request object has the following general structure:

```
{
  "jobLocationURI": "spsscr:///xxxx/xxxx",
  "jobOptions": {
    "setNotificationEnabled": true,
    "jobParameterValue": [
      {
        "Name": "string",
        "Value": "string"
      }
    ]
  }
}
```

`jobLocationURI` must have the prefix `spsscr://` appended with folder name and job name like `/folderA/jobA` (for example, `spsscr:///folderA/jobA`).

Response

A successful request returns the following information:

- Status code 200
- Response body containing the job submission result for the request. For example:

```
Job is submitted with options and execution ID is xxxx
```

An unsuccessful request returns a response code other than 200.

Example

This request submits a job:

```
POST http://cdssvr:80/process/rest/job/submitJobWithOptions
```

This is the JSON response object, indicating the specified job is submitted:

```
Job is submitted with options and execution ID is 006fcff6153b1e79000001791502452ef9f9
```

getExecutionDetails API

Retrieves the metadata and results for a specified execution.

Request

```
GET http://{server}:{port}/process/rest/job/getExecutionDetails?executionID={execution_id}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{execution_id}* is the execution ID of a job, job step, or job iteration

Table 11. Request header values

Name	Value
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

Response

A successful request returns the following information:

- Status code 200
- Response body containing the metadata and results for a specified execution

An unsuccessful request returns a response code other than 200.

Example

This request retrieves the metadata and results for a specified execution:

```
GET http://cdssvr:80/process/rest/job/getExecutionDetails?
executionID=006fcff6153b1e79000001791502452ef9f9
```

This is the JSON response object:

```
{
   "uuid": "006fcff6153b1e79000001791502452ef9f9",
   "artifactLocation": [],
   "log": null,
   "executionState": "ENDED",
   "queuedDateTime": "2021-04-30 02:42:36",
   "startDateTime": "2021-04-30 02:42:36",
   "endDateTime": "2021-04-30 02:42:37",
   "completionCode": 0,
   "executionSuccess": true,
   "eventUuid": "006fcff6153b1e79000001791502452ef6ea",
   "eventName": "general",
   "notificationEnabled": true,
   "executionWarning": false,
   "userName": "admin",
   "hasIterations": false
}
```

getJobStepExecutions API

Retrieves the metadata and results for the executions resulting from each step of a job.

Request

```
GET http://{server}:{port}/process/rest/job/getJobStepExecutions?executionID={execution_id}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{execution_id}* is the execution ID of a job

Table 12. Request header values

Name	Value
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.

Table 12. Request header values (continued)	
Name	**Value**
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

Response

A successful request returns the following information:

- Status code 200
- Response body containing the metadata and results for a specified job execution

An unsuccessful request returns a response code other than 200.

Example

This request retrieves the metadata and results for the executions resulting from each step of a job:

```
GET http://cdssvr:80/process/rest/job/getJobStepExecutions?
executionID=006fcff6153b1e79000001791502452ef9f9
```

This is the JSON response object:

```
[
      {
      "uuid": "006fcff6153b1e79000001791502452efa84",
      "artifactLocation": [],
      "log":       {
         "logAsString": "--- PRMS ---\ncd\nThe process terminated with the exit code 0.\n---
STDOUT ---\n--- STDERR ---\n",
         "logAsAttachment": null
      },
      "executionState": "ENDED",
      "queuedDateTime": 1619778870000,
      "startDateTime": 1619778875000,
      "endDateTime": 1619778877000,
      "completionCode": 0,
      "executionSuccess": true,
      "eventUuid": "006fcff6153b1e79000001791502452efa57",
      "eventName": "Event 2",
      "notificationEnabled": true,
      "executionWarning": false,
      "userName": null,
      "hasIterations": false
   },
      {
      "uuid": "006fcff6153b1e79000001791502452efa89",
      "artifactLocation": [],
      "log":       {
         "logAsString": "--- PRMS ---\ncd\nThe process terminated with the exit code 0.\n---
STDOUT ---\n--- STDERR ---\n",
         "logAsAttachment": null
      },
      "executionState": "ENDED",
      "queuedDateTime": 1619778877000,
      "startDateTime": 1619778879000,
      "endDateTime": 1619778880000,
      "completionCode": 0,
      "executionSuccess": true,
      "eventUuid": "006fcff6153b1e79000001791502452efa55",
      "eventName": "Event 1",
      "notificationEnabled": true,
      "executionWarning": false,
      "userName": null,
      "hasIterations": false
```

```
    }
]
```

getJobStepChildExecutions API

Retrieves the metadata and results for the executions resulting from job iterations. Job step execution IDs used as input to this operation can be obtained from the execution details returned by the `getJobStepExecutions` operation.

Request

```
GET http://{server}:{port}/process/rest/job/getJobStepChildExecutions?
jobStepExecutionID={job_step_execution_id}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{job_step_execution_id}* is the execution ID of the job

Table 13. Request header values

Name	Value
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

Response

A successful request returns the following information:

- Status code 200
- Response body containing the metadata and results for the executions resulting from job iterations

An unsuccessful request returns a response code other than 200.

Example

Thisrequest retrieves the metadata and results for the executions resulting from job iterations.

```
GET http://cdssvr:80/process/rest/job/getJobStepChildExecutions?
jobStepExecutionID={job_step_execution_id}
```

This is the JSON response object:

```
[
      {
```

```
      "uuid": "006fcff6153b1e79000001791502452efa84",
      "artifactLocation": [],
      "log":       {
         "logAsString": "--- PRMS ---\ncd\nThe process terminated with the exit code 0.\n---
STDOUT ---\n--- STDERR ---\n",
         "logAsAttachment": null
      },
      "executionState": "ENDED",
      "queuedDateTime": 1619778870000,
      "startDateTime": 1619778875000,
      "endDateTime": 1619778877000,
      "completionCode": 0,
      "executionSuccess": true,
      "eventUuid": "006fcff6153b1e79000001791502452efa57",
      "eventName": "Event 2",
      "notificationEnabled": true,
      "executionWarning": false,
      "userName": null,
      "hasIterations": false
   },
      {
      "uuid": "006fcff6153b1e79000001791502452efa89",
      "artifactLocation": [],
      "log":       {
         "logAsString": "--- PRMS ---\ncd\nThe process terminated with the exit code 0.\n---
STDOUT ---\n--- STDERR ---\n",
         "logAsAttachment": null
      },
      "executionState": "ENDED",
      "queuedDateTime": 1619778877000,
      "startDateTime": 1619778879000,
      "endDateTime": 1619778880000,
      "completionCode": 0,
      "executionSuccess": true,
      "eventUuid": "006fcff6153b1e79000001791502452efa55",
      "eventName": "Event 1",
      "notificationEnabled": true,
      "executionWarning": false,
      "userName": null,
      "hasIterations": false
   }
]
```


getJobParameters API

Retrieves the variable definitions for a specified job.

Request

```
GET http://{server}:{port}/process/rest/job/getJobParameters?jobLocationURI={job_uri}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{job_uri}* is the URI location of a job (for example, `spsscr:///folderA/jobA`)

Table 14. Request header values

Name	Value
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.

Table 14. Request header values (continued)	
Name	**Value**
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

Response

A successful request returns the following information:

- Status code 200
- Response body containing the job variables for a specified job

An unsuccessful request returns a response code other than 200.

Example

This request retrieves the variable definitions for a specified job:

```
GET http://cdssvr:80/process/rest/job/getJobParameters?jobLocationURI=”spsscr:///folderA/jobA”
```

This is the JSON response object:

```
{"parameter": [
      {
      "name": "VariableName1",
      "description": "",
      "default": "Value1"
   },
      {
      "name": "VariableName2",
      "description": "",
      "default": "Value2"
   }
]}
```

cancelExecution API

The `cancelExecution` operation cancels the processing of an execution.

Request

```
POST http://{server}:{port}/process/rest/job/cancelExecution?executionID={execution_id}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{execution_id}* is the identifier for the execution to be canceled. To cancel the execution of a job, supply the `cancelExecution` operation with a string corresponding to the identifier for the job being canceled.

Table 15. Request header values	
Name	**Value**
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

Response

A successful request returns the following information:

- Status code 200
- Response body containing the result for the cancellation request

An unsuccessful request returns a response code other than 200.

Example

This request retrieves the metadata and results for a specified execution:

```
GPOST http://cdssvr:80/process/rest/job/cancelExecution?
executionID=006fcff6153b1e79000001791502452ef9f9
```

This is the JSON response object:

```
Cancellation is submitted for execution 006fcff6153b1e79000001791502452ef9f9
```

deleteJobExecutions API

Deletes one or more job executions from the repository.

Request

```
POST http://{server}:{port}/process/rest/job/deleteJobExecutions?jobExecutionIDs={execution_ids}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{execution_ids}* are the execution IDs of the jobs to delete. To delete the executions of a job, supply the `deleteJobExecutions` operation with an array of strings corresponding to the identifiers for the job executions to delete.

Table 16. Request header values	
Name	**Value**
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

Response

A successful request returns the following information:

- Status code 200
- Response body containing the result for the deletion request

An unsuccessful request returns a response code other than 200.

Example

This request retrieves the metadata and results for a specified execution:

```
POST http://cdssvr:80/process/rest/job/deleteJobExecutions?
jobExecutionIDs=006fcff6153b1e79000001791502452ef9f9
```

This is the JSON response object:

```
Deletion submitted
```

getTimeBasedSchedules API

Retrieves all time-based schedules associated with a specified job.

Request

```
GET http://{server}:{port}/process/rest/job/getTimeBasedSchedules?jobLocationURI={job_uri}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{job_uri}* is the URI for a job

Table 17. Request header values	
Name	**Value**
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

Response

A successful request returns the following information:

- Status code 200
- Response body containing the metadata of all time-based schedules on a job

An unsuccessful request returns a response code other than 200.

Example

This request retrieves all time based schedules associated with a specified job:

```
GET http://cdssvr:80/process/rest/job/getTimeBasedSchedules?jobLocationURI=”spsscr:///folderA/
jobA”
```

This is the JSON response object:

```
{
   "schedule":    [
            {
         "uuid": "3e0198a5dfceabf400000179277304d69d92",
         "nextScheduledDateTime": 1620284520000,
         "scheduleEnabled": true,
         "scheduledLabel": "LATEST",
         "credentialID": "3e0198a5dfceabf400000179277304d69d8e",
         "interval": 2,
         "startDateTime": 1620283800000,
         "endDateTime": null,
         "timeOfDay": 114600000
      },
            {
         "uuid": "3e0198a5dfceabf400000179277304d69e22",
         "nextScheduledDateTime": 1620288060000,
         "scheduleEnabled": true,
         "scheduledLabel": "LATEST",
         "credentialID": "3e0198a5dfceabf400000179277304d69d8e",
         "interval": 1,
         "startDateTime": 1620284460000,
         "endDateTime": null,
         "timeOfDay": 28860000
      }
   ],
   "reserved": null
}
```

getTimeBasedSchedule API

Returns information about the schedule corresponding to a specified identifier.

Request

```
GET http://{server}:{port}/process/rest/job/getTimeBasedSchedule?scheduleID={schedule_id}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{schedule_id}* is the identifier for a schedule, which can be obtained from the `getTimeBasedSchedules` API

Table 18. Request header values

Name	Value
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

The HTTP request body should be empty.

Response

A successful request returns the following information:

- Status code 200
- Response body containing information about the schedule

An unsuccessful request returns a response code other than 200.

Example

This request retrieves the information about the schedule corresponding to a specified identifier:

```
GET http://cdssvr:80/process/rest/job/getTimeBasedSchedule?
scheduleID=3e0198a5dfceabf400000179277304d69d92
```

This is the JSON response object:

```
{
   "schedule":    {
      "uuid": "3e0198a5dfceabf400000179277304d69d92",
      "nextScheduledDateTime": 1620287040000,
      "scheduleEnabled": true,
```

```
      "scheduledLabel": "LATEST",
      "credentialID": "3e0198a5dfceabf400000179277304d69d8e",
      "interval": 2,
      "startDateTime": 1620283800000,
      "endDateTime": null,
      "timeOfDay": 114600000
   },
   "reserved": null
}
```

getMessageDrivenSchedules API

Retrieves all message-based schedules associated with a specified job.

Request

```
GET http://{server}:{port}/process/rest/job/getMessageDrivenSchedules?jobLocationURI={job_uri}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{job_uri}* is the URI for a job

Table 19. Request header values

Name	Value
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

Response

A successful request returns the following information:

- Status code 200
- Response body containing the metadata of all message-based schedules for a job

An unsuccessful request returns a response code other than 200.

Example

This request retrieves all message-based schedules associated with a specified job:

```
GET http://cdssvr:80/process/rest/job/getMessageDrivenSchedules?jobLocationURI="spsscr:///
folderA/jobA"
```

This is the JSON response object:

```
{
   "messageDrivenJob": [   {
      "uuid": "3e0198a5dfceabf400000179277304d69daf",
      "enabled": true,
      "label": "LATEST",
      "errorFlag": true,
      "errorMessage": "",
      "credentialID": "3e0198a5dfceabf400000179277304d69d8e",
      "messageDomainID": "3e0198a5dfceabf400000179277304d69d98",
      "messageText": "text1",
      "messageSelector": "selector1",
      "durableSubscription": false,
      "jmsHeaderMapping": false
   }],
   "reserved": null
}
```

getMessageDrivenSchedule API

Returns information about the message-driven job corresponding to a specified identifier.

Request

```
GET http://{server}:{port}/process/rest/job/getMessageDrivenSchedule?scheduleID={schedule_id}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{schedule_id}* is the identifier for a schedule, which can be obtained from the `getMessageDrivenSchedules` API

Table 20. Request header values

Name	Value
Content-Type	Indicates the payload type of the submitting job request that is sent from the client to the server. Valid value is `application/json`.
Accept	Indicates the payload type to return to the client. Valid value is `application/json`.
Authorization	Authentication string encoded using Base64.
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information.

Response

A successful request returns the following information:

- Status code 200
- Response body containing the information about a schedule

An unsuccessful request returns a response code other than 200.

Example

This request retrieves the information about the schedule corresponding to a specified identifier:

```
GET http://cdssvr:80/process/rest/job/getMessageDrivenSchedule?
scheduleID=3e0198a5dfceabf400000179277304d69daf
```

This is the JSON response object, indicating that there are two scoring providers installed.

```
{
   "messageDrivenJob":    {
      "uuid": "3e0198a5dfceabf400000179277304d69daf",
      "enabled": true,
      "label": "LATEST",
      "errorFlag": true,
      "errorMessage": " ",
      "credentialID": "3e0198a5dfceabf400000179277304d69d8e",
      "messageDomainID": "3e0198a5dfceabf400000179277304d69d98",
      "messageText": "text1",
      "messageSelector": "selector1",
      "durableSubscription": false,
      "jmsHeaderMapping": false
   },
   "reserved": null
}
```

Chapter 4. JSON reference for the Process Management Service

Describes all JSON objects used in the Process Management Service REST API.

The submitJobInput object

Describes the input format for the `submitJob` API.

Table 21. Child elements for submitJobInput

Element name	Value type	Description
`jobLocationURI`	string	URI for a job.
`notificationEnabled`	boolean	Indicator of whether or not notifications associated with the job should be sent as a result of the job submission.

Example

```
{
  "jobLocationURI": " spsscr:///xxxx/xxxx",
  "notificationEnabled": true
}
```

The submitJobWithOptionsInput object

Describes the input format for the `submitJobWithOptionsInput` API.

Table 22. Child elements for submitJobWithOptionsInput

Element name	Value type	Description
`jobLocationURI`	string	URI for a job.
`jobOptions`	jobOptions	Options for running a job.

Example

```
{
  "jobLocationURI": "spsscr:///xxxx/xxxx",
  "jobOptions": {
    "setNotificationEnabled": true,
    "jobParameterValue": [
      {
        "Name": "string",
        "Value": "string"
      }
    ]
  }
}
```

The jobOptions element

Describes an input table for `jobOptions`.

Table 23. Child elements for jobOptions		
Element name	**Value type**	**Description**
`setNotificationEnabled`	boolean	Indicates whether or not notifications associated with the job should be sent as a result of the job submission.
`jobParameterValue`	`jobParameterValue`	The list of parameters for this job.

Example

```
"jobOptions": {
  "setNotificationEnabled": true,
  "jobParameterValue": [
    {
      "Name": "string",
      "Value": "string"
    }
  ]
}
```

The jobParameterValue element

Describes an input table for `jobParameterValue`.

Table 24. Child elements for jobParameterValue		
Element name	**Value type**	**Description**
`jobParameter`	`jobParameter`	One parameter of a job. Consists of parameter name and its value.

Example

```
  "jobParameterValue": [
    {
      "Name": "string",
      "Value": "string"
    }
  ]
```

Appendix A. Deprecated features

If you are migrating from an earlier release of IBM SPSS Collaboration and Deployment Services, you should be aware of the various features that have been deprecated since the last version.

If a feature is deprecated, IBM Corp. might remove this capability in a subsequent release of the product. Future investment will be focused on the strategic function listed under the recommended migration action. Typically, a feature is not deprecated unless an equivalent alternative is provided.

No features have been deprecated in this release. For reference purposes, the following table indicates features that were deprecated in recent previous versions of the product. Where possible, the table also indicates the recommended migration action.

Table 25. Features deprecated in previous versions

Deprecation	**Recommended migration action**
Security Provider: Active Directory with local override, which supports extended groups and allowed users	Use the standard Active Directory security provider with any necessary groups added
IBM SPSS Collaboration and Deployment Services Enterprise View	Use the Analytic Data View feature
IBM SPSS Collaboration and Deployment Services Enterprise View Driver	Use the Analytic Data View feature
Scenario files	Scenario files (`.scn`) are no longer supported. Enterprise View source nodes cannot be modified in Deployment Manager. Old scenario files can be modified in IBM SPSS Modeler client and resaved as stream files. Also, scoring configurations that used a scenario file must be deleted and recreated based on a stream file.
Web Install for IBM SPSS Deployment Manager	Use the standalone installer
BIRT Report Designer for IBM SPSS	None
BIRT Report Designer for IBM SPSS viewer	None
IBM SPSS Collaboration and Deployment Services Portlet	Use the IBM SPSS Collaboration and Deployment Services Deployment Portal directly, or use the web services APIs
IBM SPSS Collaboration and Deployment Services Web Part	Use the IBM SPSS Collaboration and Deployment Services Deployment Portal directly, or use the web services APIs
Scoring Service V1 API	Scoring Service V2 API
Scheduling Server Service	None
Reporting Service	None
Authentication Service `login` operation	Authentication Service `doLogin` operation
Search Service `search` operation	Search Service `search2.5` operation
SPSS AXIS/Castor web services client jar	Use the tools provided with the Java Runtime Environment, Integrated Development Environment, or Eclipse Web Tools Platform (WTP)

Table 25. Features deprecated in previous versions (continued)	
Deprecation	**Recommended migration action**
`clemrtl_setLogFile()` API function	None

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBMproducts. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be

trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other product and service names might be trademarks of IBM or other companies.

Glossary

This glossary includes terms and definitions for IBM SPSS Collaboration and Deployment Services.

The following cross-references are used in this glossary:

- See refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- See also refers you to a related or contrasting term.

To view glossaries for other IBM products, go to www.ibm.com/software/globalization/terminology (opens in new window).

A

access control list (ACL)
In computer security, a list associated with an object that identifies all the subjects that can access the object and their access rights.

ACL
See access control list.

action
A permission for an aspect of system functionality. For example, the ability to set up notifications is defined as an action. Actions are grouped and assigned to users through roles. See also role.

Active Directory (AD)
A hierarchical directory service that enables centralized, secure management of an entire network, which is a central component of the Microsoft Windows platform.

AD
See Active Directory.

allowed user
A subset of the users defined in a remote directory, such as SiteMinder or Windows Active Directory, that are allowed access to SPSS Predictive Enterprise Services. Allowed users are defined when only a few users in a remote directory need access to the application.

API
See application programming interface.

appender
A component that receives logging requests from a logger and writes log statements to a specified file or console. See also logger.

application programming interface (API)
An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

B

batch file
A file that contains instructions that are processed sequentially, as a unit.

binary large object (BLOB)
A data type whose value is a sequence of bytes that can range in size from 0 bytes to 2 gigabytes less 1 byte. This sequence does not have an associated code page and character set. BLOBs can contain, for example, image, audio, or video data.

BLOB
See binary large object.

break group
A set of rows of returned data that are grouped according to a common column value. For example, in a column of states, the rows of data for each state are grouped together.

burst report
A report that generates multiple output files during a single run by using multiple input parameters taken from break groups in the report.

C

cascading permission
A permission of a parent folder in the content repository that has been propagated to its child objects.

character large object (CLOB)
A data type whose value is a sequence of characters (single byte, multibyte, or both) that can range in size from 0 bytes to 2 gigabytes less 1 byte. In general, the CLOB data type is used whenever a character string might exceed the limits of the VARCHAR data type.

CLOB
See character large object.

common warehouse metamodel (CWM)
A metamodel written to be a common standard by the Object Management Group (OMG).

content repository
A centralized location for storing analytical assets, such as models and data. Content repository includes facilities for security and access control, content management, and process automation.

context data
Input data that is passed with a scoring request in real time. For example, when a score is requested for a customer based on credit rating and geocode, the credit score and geocode will be the context data for the request.

credential
Information acquired during authentication that describes a user, group associations, or other security-related identity attributes, and that is used to perform services such as authorization, auditing, or delegation. For example, a user ID and password are credentials that allow access to network and system resources.

CWM
See common warehouse metamodel.

D

data warehouse
A subject-oriented collection of data that is used to support strategic decision making. The warehouse is the central point of data integration for business intelligence. It is the source of data for data marts within an enterprise and delivers a common view of enterprise data.

distinguished name (DN)
The name that uniquely identifies an entry in a directory. A distinguished name is made up of attribute:value pairs, separated by commas. For example, CN=person name and C=country or region.

DN
See distinguished name.

Document Object Model (DOM)
A system in which a structured document, for example an XML file, is viewed as a tree of objects that can be programmatically accessed and updated. See also Simple API for XML.

document type definition (DTD)
The rules that specify the structure for a particular class of SGML or XML documents. The DTD defines the structure with elements, attributes, and notations, and it establishes constraints for how each element, attribute, and notation can be used within the particular class of documents.

DOM
See Document Object Model.

dormant schedule
A schedule associated with a deleted or unlabeled version of a job. A dormant schedule cannot be used until it is associated with a valid labeled job version.

DTD
See document type definition.

E

EAR
See enterprise archive.

enterprise archive (EAR)
A specialized type of JAR file, defined by the Java EE standard, used to deploy Java EE applications to Java EE application servers. An EAR file contains EJB components, a deployment descriptor, and web archive (WAR) files for individual web applications. See also Java archive, web archive.

execution server
A server that enables analytical processing of resources stored in the repository. For example, to execute an IBM SPSS Statistics syntax in an IBM SPSS Collaboration and Deployment Services job, an IBM SPSS Statistics execution server must be designated.

export
The process of storing objects and metadata from the content repository to an external file.

extended group
A locally-defined group of remote users. Extended groups are defined when groups in the remote directory are not fine-grained enough.

Extensible Markup Language (XML)
A standard metalanguage for defining markup languages that is based on Standard Generalized Markup Language (SGML).

Extensible Stylesheet Language (XSL)
A language for specifying style sheets for XML documents. Extensible Stylesheet Language Transformation (XSLT) is used with XSL to describe how an XML document is transformed into another document.

F

field content assist
A feature that provides predefined system and variable values for entry fields.

G

general job step
A method for running native operating system commands and executable programs on a host or a remote process server. General jobs have access to files stored within the repository and on the file system and can be used to control the input/output of analytical processing.

I

import
The process of adding objects and metadata defined in an external file generated by export, to the content repository.

iterative consumer reporting job step
A job step that is passed a set of input values generated by a preceding iterative producer reporting job step. The report in iterative consumer job step is executed for each tuple in the received data set.

iterative producer reporting job step
A job step that generates a set of values passed as input parameters to a following iterative consumer job step.

J

JAAS
See Java Authentication and Authorization Service.

JAR
See Java archive.

Java archive (JAR)
A compressed file format for storing all of the resources that are required to install and run a Java program in a single file. See also enterprise archive, web archive.

Java Authentication and Authorization Service (JAAS)
In Java EE technology, a standard API for performing security-based operations. Through JAAS, services can authenticate and authorize users while enabling the applications to remain independent from underlying technologies.

Java Generic Security Services (JGSS)
A specification that provides Java programs access to the services that include the signing and sealing of messages and a generic authentication mechanism.

Java Naming and Directory Interface (JNDI)
An extension to the Java platform that provides a standard interface for heterogeneous naming and directory services.

JGSS
See Java Generic Security Services.

JNDI
See Java Naming and Directory Interface.

job
A mechanism for automating analytical processing. A job consists of job steps, executed sequentially or conditionally. Input parameters can be defined for a job. A job can be run on demand or triggered by time-based or message-based schedules, with records of job execution stored as job history.

job step
A discrete unit of processing in a job. Depending on the type, job steps can be run on the content repository host or specially defined execution or remote process servers. Objects stored in the repository or the file system can provide input for a job step, and job step output can be stored in the repository or written to the file system.

K

KDC
See key distribution center.

Kerberos
A network authentication protocol that is based on symmetric key cryptography. Kerberos assigns a unique key, called a ticket, to each user who logs on to the network. The ticket is embedded in messages that are sent over the network. The receiver of a message uses the ticket to authenticate the sender.

key distribution center (KDC)
A network service that provides tickets and temporary session keys. The KDC maintains a database of principals (users and services) and their associated secret keys. It is composed of the authentication server and the ticket granting ticket server.

keystore
In security, a file or a hardware cryptographic card where identities and private keys are stored, for authentication and encryption purposes. Some keystores also contain trusted or public keys.

L

LDAP
See Lightweight Directory Access Protocol.

Lightweight Directory Access Protocol (LDAP)
An open protocol that uses TCP/IP to provide access to directories that support an X.500 model and that does not incur the resource requirements of the more complex X.500 Directory Access Protocol (DAP). For example, LDAP can be used to locate people, organizations, and other resources in an Internet or intranet directory.

lock
The process by which integrity of data is ensured by preventing more than one user from accessing or changing the same data or object at the same time.

logger
A component that prepares log statements to be written to console or log file. See also appender.

M

message-based schedule
A schedule that is used to trigger job execution by an event signalled by a Java Messaging Service (JMS) message. For example, when a job relies on the input from a third-party application, the application must send a JMS message when the input file is ready for processing.

metamodel
A model that defines the language for expressing a model.

meta-object
An instance of an XMI class as defined in the metamodel.

meta-object facility (MOF)
A generalized facility and repository for storing abstract information about concrete object systems; dealing mostly with construction, standardized by the Object Management Group (OMG).

MIME
See Multipurpose Internet Mail Extensions.

MOF
See meta-object facility.

Multipurpose Internet Mail Extensions (MIME)
An Internet standard that allows different forms of data, including video, audio, or binary data, to be attached to email without requiring translation into ASCII text.

N

notification
A mechanism that is used to generate email messages informing users of specific types of system events, such as changes to content repository objects and processing success and failure. Unlike subscriptions, notifications can be set up to send email to multiple users.

O

Object Management Group (OMG)
A non-profit consortium whose purpose is to promote object-oriented technology and the standardization of that technology. The Object Management Group was formed to help reduce the complexity, lower the costs, and hasten the introduction of new software applications.

ODS
See Output Delivery System.

OMG
See Object Management Group.

Output Delivery System (ODS)
A method of controlling the destination for output within SAS. ODS can route SAS output to a SAS data file, a text listing file, HTML files, and files optimized for high-resolution printing.

P

package
An installable unit of a software product. Software product packages are separately installable units that can operate independently from other packages of that software product.

principal
An entity that can communicate securely with another entity. A principal is identified by its associated security context, which defines its access rights.

R

remote process server
A remote system that is designated for running native operating system commands and executable programs.

repository content adapter
An optional software package that enables storing and processing content from other IBM SPSS applications, such as Statistics, Modeler, and Data Collection, as well as third parties.

repository database
A relational database that is used for storing content repository objects and metadata.

resource
A content repository object.

resource definition
A subset of content repository resources used to enable analytical processing, such as definitions of data sources, credentials, execution servers, and JMS message domains.

role
A set of permissions or access rights. See also action.

S

SAX
See Simple API for XML.

schedule
A content repository object that triggers job execution.

scoring configuration
A configuration that defines model-specific settings for generating real-time scores, such as input data, processing rules, outputs, logging, etc.

security provider
A system that performs user authentication. Users and groups can be defined locally (in which case, IBM SPSS Collaboration and Deployment Services itself is the security provider) or derived from a remote directory, such as Windows Active Directory or OpenLDAP.

service provider interface (SPI)
An API that supports replaceable components and can be implemented or extended by a third party.

SGML
See Standard Generalized Markup Language.

shell script
A program, or script, that is interpreted by the shell of an operating system.

Simple API for XML (SAX)
An event-driven, serial-access protocol for accessing XML documents, used. A Java-only API, SAX is used by most servlets and network programs to transmit and receive XML documents. See also Document Object Model.

single sign-on (SSO)
An authentication process in which a user can access more than one system or application by entering a single user ID and password.

SOAP
A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP can be used to query and return information and invoke services across the Internet.

SPI
See service provider interface.

SSO
See single sign-on.

Standard Generalized Markup Language (SGML)
A standard metalanguage for defining markup languages that is based on the ISO 8879 standard. SGML focuses on structuring information rather than presenting information; it separates the structure and content from the presentation. It also facilitates the interchange of documents across an electronic medium.

stop word
A word that is commonly used, such as "the," "an," or "and," that is ignored by a search application.

subscription
Email notices and Really Simple Syndication (RSS) feeds that repository users create to receive when the state of an asset changes.

T

TGT
See ticket-granting ticket.

ticket-granting ticket (TGT)
A ticket that allows access to the ticket granting service on the key distribution center (KDC). Ticket granting tickets are passed to the principal by the KDC after the principal has completed a successful request. In a Windows 2000 environment, a user logs on to the network and the KDC will verify the principal's name and encrypted password and then send a ticket granting ticket to the user.

time-based schedule
A schedule that triggers job execution at a specified time or date. For example, a time-based schedule may run a job at 5:00 pm every Thursday.

U

Universally Unique Identifier (UUID)
The 128-bit numeric identifier that is used to ensure that two components do not have the same identifier.

UUID
See Universally Unique Identifier.

V

Velocity
A Java-based template engine that provides a simple and powerful template language to reference objects defined in Java code. Velocity is an open source package directed by the Apache Project.

W

W3C
See World Wide Web Consortium.

WAR
See web archive.

web archive (WAR)
A compressed file format, defined by the Java EE standard, for storing all the resources required to install and run a web application in a single file. See also enterprise archive, Java archive.

Web Services Description Language (WSDL)
An XML-based specification for describing networked services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

World Wide Web Consortium (W3C)
An international industry consortium set up to develop common protocols to promote evolution and interoperability of the World Wide Web.

WSDL
See Web Services Description Language.

X

XMI
See XML Metadata Interchange.

XML
See Extensible Markup Language.

XML Metadata Interchange (XMI)
A model-driven XML integration framework for defining, interchanging, manipulating, and integrating XML data and objects. XMI-based standards are in use for integrating tools, repositories, applications, and data warehouses.

XSL
See Extensible Stylesheet Language.

Index

R

S

U

W

Y