IBM SPSS Collaboration and Deployment Services 5 Process Management Service Developer's Guide

Note: Before using this information and the product it supports, read the general information under Notices on p. 97.

This edition applies to IBM SPSS Collaboration and Deployment Services 5 and to all subsequent releases and modifications until otherwise indicated in new editions.

Adobe product screenshot(s) reprinted with permission from Adobe Systems Incorporated.

Microsoft product screenshot(s) reprinted with permission from Microsoft Corporation.

Preface

This guide is intended for developers working with the web services available in IBM® SPSS® Collaboration and Deployment Services. Users should have experience writing web service client applications and are assumed to have knowledge of IBM Corp. applications, Java and .NET development, data modeling concepts, and related technologies.

About IBM Business Analytics

IBM Business Analytics software delivers complete, consistent and accurate information that decision-makers trust to improve business performance. A comprehensive portfolio of business intelligence, predictive analytics, financial performance and strategy management, and analytic applications provides clear, immediate and actionable insights into current performance and the ability to predict future outcomes. Combined with rich industry solutions, proven practices and professional services, organizations of every size can drive the highest productivity, confidently automate decisions and deliver better results.

As part of this portfolio, IBM SPSS Predictive Analytics software helps organizations predict future events and proactively act upon that insight to drive better business outcomes. Commercial, government and academic customers worldwide rely on IBM SPSS technology as a competitive advantage in attracting, retaining and growing customers, while reducing fraud and mitigating risk. By incorporating IBM SPSS software into their daily operations, organizations become predictive enterprises – able to direct and automate decisions to meet business goals and achieve measurable competitive advantage. For further information or to reach a representative visit *http://www.ibm.com/spss*.

Technical support

Technical support is available to maintenance customers. Customers may contact Technical Support for assistance in using IBM Corp. products or for installation help for one of the supported hardware environments. To reach Technical Support, see the IBM Corp. web site at *http://www.ibm.com/support*. Be prepared to identify yourself, your organization, and your support agreement when requesting assistance.

Contents

1 Introduction to web services 1

What are web services? . . . 1
Web service system architecture . . . 1
Web service protocol stack . . . 2
Simple Object Access Protocol . . . 3
Web Service Description Language . . . 3
Proxies . . . 6

2 Process Management Service overview 8

Accessing the Process Management Service . . . 8
Calling Process Management Service operations . . . 8

3 Process management concepts 9

Jobs . . . 9
Content repository Uniform Resource Identifers . . . 9
Job variables . . . 10
Schedules . . . 10
Time-based schedules . . . 11
Message-based schedules . . . 11
Executions . . . 15
Queries . . . 15
Return specifers . . . 18
Page selectors . . . 18
Page results . . . 18

4 Operation reference 20

The cancelExecution operation . . . 20
The createMessageDrivenJob operation . . . 21
The createSchedule operation . . . 23
The deleteJobExecutions operation . . . 26
The deleteJobTrigger operation . . . 27
The deleteJobTriggers operation . . . 29

The deleteSchedule operation . . . 30
The deleteSchedules operation . . . 31
The finalizeRemoteWork operation . . . 33
The getCustomEventTypes operation . . . 33
The getExecutionDetails operation . . . 34
The getJobParameters operation . . . 37
The getJobStepChildExecutions operation . . . 38
The getJobStepExecutions operation . . . 41
The getMessageDrivenJob operation . . . 43
The getMessageDrivenJobs operation . . . 45
The getSchedule operation . . . 46
The getSchedules operation . . . 48
The getVersion operation . . . 50
The getWorkTypes operation . . . 51
The handleMessageDomainChanged operation . . . 53
The queryAllSchedules operation . . . 54
The queryExecutions operation . . . 58
The queryJobTriggers operation . . . 63
The querySchedules operation . . . 71
The querySubmittedExecutions operation . . . 77
The rerunFailedJobStepIterations operation . . . 81
The submitJob operation . . . 81
The submitJobWithOptions operation . . . 83
The submitWork operation . . . 85
The updateMessageDrivenJob operation . . . 87
The updateSchedule operation . . . 89

Appendices

A Microsoft® .NET Framework-based clients 92

Adding a service reference . . . 92
    Service reference modifications . . . 93
Configuring the web service endpoint . . . 94
Configuring endpoint behaviors . . . 95
Exercising the service . . . 95

B Notices *97*

Index *100*

Chapter

1

Introduction to web services

What are web services?

At a high level, a **web service** is a set of functionality distributed across a network (LAN or the Internet) using a common communication protocol. The web service serves as an intermediary between an application and its clients, providing both a standardized information structure and a standardized communication protocol for interaction between the two. Where other methods of distributed application architecture rely on a single programming language being used on both the application and its clients, a web service allows the use of loosely coupled services between non-homogenous platforms and languages. This provides a non-architecture-specific approach allowing, for example, Java services to communicate with C# clients, or vice-versa.

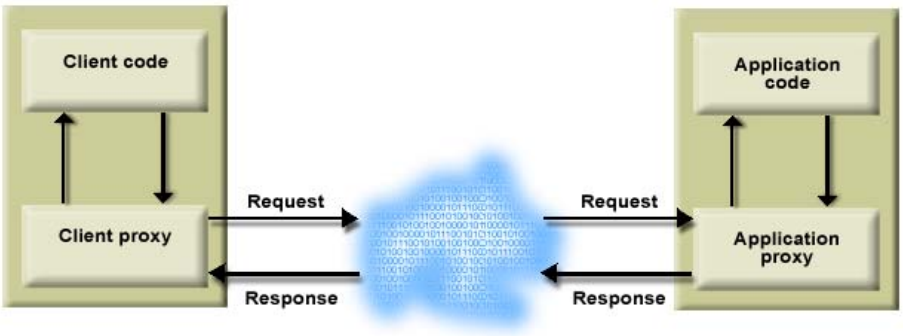
Advantages to implementing application functionality as web services include the following:

- Software written in different languages (Java or C#) running on different platforms (UNIX or Windows) can exchange services and data
- Application functionality can be accessed by a variety of clients. For example, both a thin-client interface and a rich-client interface can take advantage of the web service operations.
- Updates to the service are immediately available to all service clients

Web service system architecture

Web services are deployed and made publicly available using an application server, such as JBoss Application Server, WebSphere®, or Oracle WebLogic Server. The published web services are hosted by this application server to handle application requests, access permissions, and process load. A high-level architecture of how web services are implemented is displayed in the following diagram.

Figure 1-1
Web service architecture



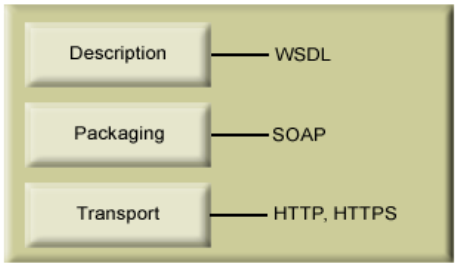
The client code supplies input to an operation offered by a proxy class. The proxy class generates a request containing a standardized representation of the input and sends it across the network to the application. A proxy class on the server receives the request and unmarshals the contents into objects for processing by the application. Upon completing the operation, the application supplies a proxy with the output. The proxy creates a standardized representation of that output and sends the response back to the client. The client proxy unmarshals the response into native objects for subsequent processing by the client code.

Standardizing the format of the information passing between the client and the application allows a client written in one programming language to communicate with an application written in another. The proxy classes, which are automatically generated from a web service description by a variety of toolkits, handle the translation between native programming objects and the standardized representation. For more information, see the topic Proxies on p. 6.

Web service protocol stack

A web service implementation depends on technologies often organized in a layered stack. The implementation itself defines a standard protocol for each technology layer, with each layer depending on the layers appearing below it in the stack.

Figure 1-2
Web service protocol stack



Beginning at the bottom of the stack, the Transport layer defines the technology standards for communication, allowing information to move across the network. HTTP or HTTPS are often used as the standard for the transport layer.

The Packaging layer rests on top of Transport and defines the standard for structuring information for transport across the network. The SOAP format is commonly used, which offers an XML structure for packaging the data. For more information, see the topic Simple Object Access Protocol on p. 3.

The topmost layer is Description and identifies the standards used by the layers below it in the stack, as well as providing the definition of the interface available for client use. The most common means of conveying this information is through the use of a WSDL file. For more information, see the topic Web Service Description Language on p. 3.

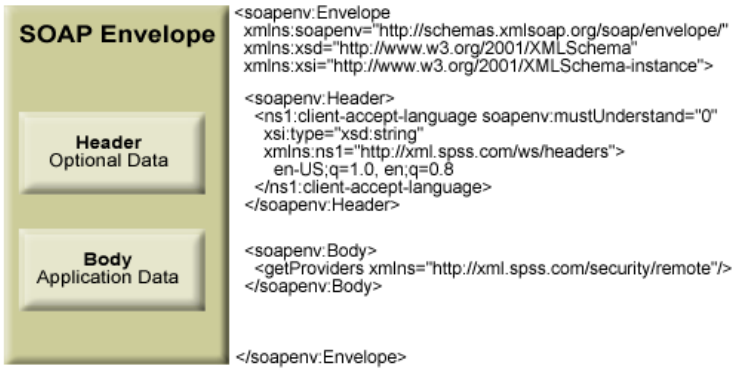
Simple Object Access Protocol

The Simple Object Access Protocol (SOAP) is a way to pass information between applications in an XML format. SOAP messages are transmitted from the sending application to the receiving application, typically over an HTTP session. The actual SOAP message is made up of the Envelope element, which contains a Body element and an optional Header element.

- **Envelope.** This mandatory element is the root of the SOAP message, identifying the transmitted XML as being a SOAP packet. An envelope contains a body section and an optional header section.
- **Header.** This optional element provides an extension mechanism indicating processing information for the message. For example, if the operation using the message requires security credentials, those credentials should be part of the envelope header.
- **Body.** This element contains the message payload, the raw data being transmitted between the sending and receiving applications. The body itself may consist of multiple child elements, with an XML schema typically defining the structure of this data.

A SOAP packet and the corresponding XML is structured in the following way:

Figure 1-3
An example SOAP packet



Web Service Description Language

A Web Service Description Language (WSDL) file provides an XML-based map of what functionality the published web service allows, separating the implementation in the service from the interface. The WSDL defines the following:

- The access location of the web service
- Operations the web service exposes
- Parameters the exposed operations accept
- Any request or response messages associated with the operations

The WSDL provides the information necessary to generate a client-side proxy in the desired programming language.

In accordance with the WSDL specification (*http://www.w3.org/TR/wsdl*) adopted by the World Wide Web Consortium, information in the WSDL is organized into the following sections:

- **Types.** Content definitions for web service operation input and output. For more information, see the topic Types on p. 4.
- **Messages.** Input and output definitions for the web service operations. For more information, see the topic Messages on p. 5.
- **PortTypes.** Groups of operations offered by the web service. For more information, see the topic Port types on p. 5.
- **Bindings.** Protocols and formats for the web service operations. For more information, see the topic Bindings on p. 5.
- **Services.** Endpoints at which the web service functionality can be accessed. For more information, see the topic Services on p. 6.

Types

The types element of a WSDL file contains the data type definitions employed by messages processed by the web service. These definitions use XML to organize the information relevant to the type element being defined. Consider the following type definitions:

```
<wsdl:types>
 <schema targetNamespace="http://xml.spss.com/security/remote"
  xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="getProviders">
   <complexType />
  </element>
  <element name="getProvidersResponse">
   <complexType>
    <sequence>
     <element name="providerInfo[unbounded]" type="tns1:providerInfo" />
    </sequence>
   </complexType>
  </element>
 </schema>
</wsdl:types>
```

This section defines two elements, *getProviders* and *getProvidersResponse*. The former is an empty element. The latter contains a sequence of *providerInfo* child elements. These children are all of the *providerInfo* type, which is defined elsewhere.

In practice, the WSDL file typically references type element definitions found in an external XML schema. For instance, the following definition uses *security-remote.xsd* to define type elements.

```
<wsdl:types>
 <xs:schema>
  <xs:import namespace="http://xml.spss.com/security/remote"
   schemaLocation="security-remote.xsd"/>
 </xs:schema>
</wsdl:types>
```

Messages

The message elements of a WSDL file defines the input or output parameters for operations available in the web service. Each message can consist of one or more parts, with the parts similar to the parameters of a function call in a traditional programming language. Consider the following two message definitions:

```
<wsdl:message name="getProvidersResponse">
 <wsdl:part element="tns2:getProvidersResponse" name="parameters" />
</wsdl:message>
<wsdl:message name="getProvidersRequest">
 <wsdl:part element="tns2:getProviders" name="parameters" />
</wsdl:message>
```

The *getProvidersResponse* message contains a single part, corresponding to the *getProvidersResponse* element defined in the types section of the WSDL file. Similarly, the *getProvidersRequest* message also contains a single part, as defined by the *getProviders* element in the types section. For more information, see the topic Types on p. 4.

Port types

The portType element of a WSDL file defines the actual interface to the web service. A port type is simply a group of related operations and is comparable to a function library, module, or class in a traditional programming language. The definition specifies the parameters for the operations, as well as any values returned. The parameters and return values correspond to messages defined elsewhere in the WSDL file. Consider the following port type definition:

```
<wsdl:portType name="ProviderInformation">
 <wsdl:operation name="getProviders">
  <wsdl:input message="impl:getProvidersRequest" name="getProvidersRequest" />
  <wsdl:output message="impl:getProvidersResponse" name="getProvidersResponse" />
 </wsdl:operation>
</wsdl:portType>
```

The *ProviderInformation* port type consists of a single operation, *getProviders*. Input to this operation corresponds to the *getProvidersRequest* message. The operation returns information in the structure defined by the *getProvidersResponse* message. For more information, see the topic Messages on p. 5.

Bindings

The binding element of a WSDL file binds the interface defined by the port type to transport and messaging protocols. Consider the following binding definition:

```
<wsdl:binding name="ProviderInformationSoapBinding" type="impl:ProviderInformation">
 <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
 <wsdl:operation name="getProviders">
  <wsdlsoap:operation soapAction="" />
  <wsdl:input name="getProvidersRequest">
   <wsdlsoap:body namespace="http://xml.spss.com/security/remote" use="literal" />
```

```
  </wsdl:input>
  <wsdl:output name="getProvidersResponse">
   <wsdlsoap:body namespace="http://xml.spss.com/security" use="literal" />
  </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
```

In this case, the transport attribute of the wsdlsoap:binding element defines HTTP as the transport protocol. Both the *getProviders* and *getProvidersReponse* operations in the interface are bound to the SOAP messaging protocol.

Services

The service element of a WSDL file identifies the network location at which the service interface can be accessed. Consider the following service definition:

```
<wsdl:service name="ProviderInformationService">
 <wsdl:port binding="impl:ProviderInformationSoapBinding" name="ProviderInformation">
  <wsdlsoap:address location="http://pes_server:8080/security-ws/services/ProviderInformation" />
 </wsdl:port>
</wsdl:service>
```

In this example, the operations comprising the *ProviderInformation* port type can be accessed at:

http://pes_server:8080/security-ws/services/ProviderInformation

Proxies

Proxies serve as bridges between the client and the web service. A client-side proxy marshals the input objects into a standardized representation which is sent to the web service. A server-side proxy unmarshals the information into input objects for the service operations. The results of the operation are marshalled into standard representations and returned to the client. The client proxy unmarshals the response information into objects for any additional processing by the client.

Creating a proxy is the first step when developing a web service client; the proxy is the translation-unit between your application and the web service the application is using. Fortunately, many development environments include tools for automatically generating the client proxy from the web service WSDL file, allowing the client developer to focus on the client application code instead of transport and packaging protocols.

The proxy classes generated from a WSDL file depend on the tool used. For Java, the *wsdl2java* tool, which is part of the Apache Axis project, can be used. This tool produces a Java class for each type in the WSDL. Each port type results in a Java interface. A binding creates a stub class, and a WSDL service yields a service interface with a locator implementation. These generated classes and interfaces can be called directly from a client application written in Java to access the web service functionality.

An alternative Java proxy tool is *wsimport*, which is part of JAX-WS. The general structure of the generated classes is similar to that created by the Axis tool, but there are some differences. For example, instead of using arrays for input fields and returned items, the code generated from

the *wsimport* tool uses List collections. In addition, if an input type matches an output type for a method, the *wsimport* tool uses a Holder class for the parameter.

In contrast, on the .NET platform, the *wsdl.exe* tool is often used to generate a web service proxy. This tool creates a single source file in a specified language containing the proxy class. This class includes both synchronous and asynchronous methods for each operation defined in the WSDL. For example, the web service operation *getProviders* results in the methods *getProviders*, *getProvidersBegin*, and *getProvidersEnd*. The latter two can be used for asynchronous processing.

A variety of other tools exist for other programming languages. For details, consult the documentation for those tools. In each case, the tool creates native programming constructs that permit leveraging a web service regardless of the service implementation language.

Chapter
2

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

To access the functionality offered by the Process Management Service, create a client application using the proxy classes generated by your preferred web service tool. The endpoint for the service is:

http://<host-name>:<port-number>/process/services/ProcessManagement

The value of <*host-name*> corresponds to the machine on which IBM® SPSS® Collaboration and Deployment Services Repository is installed, with <*port-number*> indicating the port number on which it is running. To access the WSDL file for the service, append *?wsdl* to the service endpoint.

For example, if IBM SPSS Collaboration and Deployment Services Repository is running on port 80 of the machine *cads_server*, the WSDL file can be accessed using the path:

http://cads_server:80/process/services/ProcessManagement?wsdl

Calling Process Management Service operations

Clients access the operations offered by the web service using a stub for the service. The following is an example of how to acquire a stub in Java through Axis defined methods:

```
String context = "/process/services/ProcessManagement";
URL url = new URL("http", "cads_server", 80, context);
ProcessManagementService service = new ProcessManagementServiceLocator();
stub = service.getProcessManagement(url);
```

The service operations can be called directly from the stub, such as:

```
stub.submitJob(jobURI,false);
```

Chapter
3

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 Identifers

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:

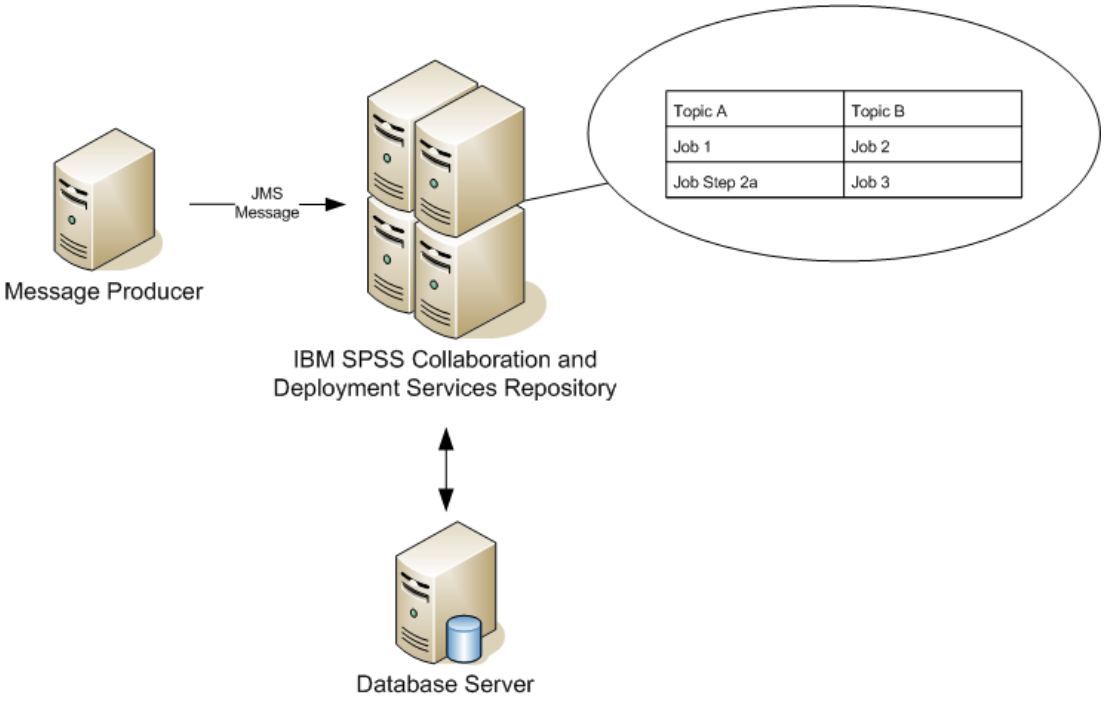
- **Once.** Executes a job a single time at a specified date/time.
- **Hourly.** Executes a job on an hourly basis. The recurrence pattern indicates the number of hours between executions. For example, set the interval to 1 to execute the job every hour. To execute the job every other hour, set the interval 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 job 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 job 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 3-1
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 associated jobs to execute. In the "IBM SPSS Collaboration and Deployment Services Messaging" 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.

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 (*http://java.sun.com/products/jms/docs.html*).

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 two types of filters: **text** and **selector**.

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, on the other hand, 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 (*http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt*). 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 desired. 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 3-1
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

Field Name	Description
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 3-2
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
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 3-3
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 3-4
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 specifers

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.
- **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, on the other hand, 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 11^{th} through the 20^{th} 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 “Example Execution Query Results” table illustrates the structure for a query returning two executions of the same job. Some columns are omitted in the interest of space.

Table 3-5
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 “Example Execution Query Results” 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
4

Operation reference

The cancelExecution operation

The cancelExecution operation cancels the processing of an execution.

Input fields

The following table lists the input fields for the cancelExecution operation.

Table 4-1
Fields for cancelExecution

Field	Type/Valid Values	Description
executionID	string	Identifier for the execution to be cancelled.

Java example

To cancel the execution of a job, supply the cancelExecution operation with a string corresponding to the identifier for the job being cancelled.

```
String [] executionID = new String[1];
executionID[0] = "0a0a4aac011937790000010d415272aa8ab4";
stub.cancelExecution(executionID);
```

SOAP request example

Client invocation of the cancelExecution operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
```

```
 <soapenv:Body>
  <cancelExecution xmlns="http://xml.spss.com/prms/remote">
   <executionID>0a0a4aac011937790000010d415272aa8ab4</executionID>
  </cancelExecution>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a cancelExecution operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <cancelExecutionResponse xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

The createMessageDrivenJob operation

Creates a new message-based trigger for a job in the system.

Input fields

The following table lists the input fields for the createMessageDrivenJob operation.

Table 4-2
Fields for createMessageDrivenJob

Field	Type/Valid Values	Description
jobLocationURI	string	URI of the job associated with the schedule.
messageDrivenJobIn	messageDrivenJobSpecifier	Message Drive Job parameters.

Return information

The following table identifies the information returned by the createMessageDrivenJob operation.

Table 4-3
Return Value

Type	Description
messageDrivenJobSpecifier	Details of a Message Drive Job Schedule.

Java example

Message-driven job creation requires the following two pieces of information:

- A string corresponding to the uri of the job associated with the trigger
- A MessageDrivenJobSpecifier object defining the parameters of the message-driven job

Typically, creating a message-driven job involves the following steps:

1. Create a MessageDrivenJob object.
2. Supply the setEnabled method with a boolean indicating whether or not the trigger is enabled.
3. Define the label for the job version associated with the trigger using the setLabel method.
4. Set the credentials for executing the message-driven job using the setCredentialID method.
5. Specify trigger filters using the the setMessageText and setMessageSelector methods.
6. Create a MessageDrivenJobSpecifier object using the job object.
7. Define a string indicating the URI for the job.
8. Supply the MessageDrivenJobSpecifier object and the URI string to the createMessageDrivenJob operation. The operation returns a MessageDrivenJobSpecifier object containing the message-driven job parameters, including its identifier.

The following sample creates a message-driven job for the job *Monthly Update*. The job runs when messages having a text body of *My Text* that meet the selector criterion are received.

```
MessageDrivenJob mdJob = new MessageDrivenJob();
mdJob.setScheduleEnabled(true);
mdJob.setLabel("LATEST");
mdJob.setCredentialID("0a0a4a356f526f1d0000011450a1669680af");
mdJob.setMessageText("My Text");
mdJob.setMessageSelector("JMSType = 'ETL' AND status = 'complete'");
MessageDrivenJobSpecifier mdjSpecIn = new MessageDrivenJobSpecifier(wSchedule);

String uri = "spsscr://pesServer/Monthly%20Update";
MessageDrivenJobSpecifier scheduleOut = stub.createMessageDrivenJob(uri, mdjSpecIn);
```

SOAP request example

Client invocation of the createMessageDrivenJob operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
```

```
      <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
      <wsse:Password xsi:type="xsd:string">password</wsse:Password>
    </wsse:UsernameToken>
   </wsse:Security>
   <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
    xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
  </soapenv:Header>
  <soapenv:Body>
   <createMessageDrivenJob xmlns="http://xml.spss.com/prms/remote">
    <jobLocationURI>spsscr://pesServer/Monthly%20Update</jobLocationURI>
    <ns2:messageDrivenJobIn xmlns:ns2="http://xml.spss.com/prms">
     <ns2:messageDrivenJob label="LATEST" credentialID="0a0a4a356f526f1d0000011450a1669680af"
      messageText="My Text" messageSelector="JMSType = 'ETL' AND status = 'complete'"/>
    </ns2:messageDrivenJobIn>
   </createMessageDrivenJob>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a createMessageDrivenJob operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <createMessageDrivenJobResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:messageDrivenJobOut xmlns:ns1="http://xml.spss.com/prms">
    <ns1:messageDrivenJob uuid="0a0a4a3519895570000000114a76f97678181" enabled="true"
     label="LATEST" messageText="My Text"/>
   </ns1:messageDrivenJobOut>
  </createMessageDrivenJobResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The createSchedule operation

Creates a new schedule in the system for a designated job.

Input fields

The following table lists the input fields for the createSchedule operation.

Table 4-4
Fields for createSchedule

Field	Type/Valid Values	Description
jobLocationURI	string	URI of the job associated with the schedule.
scheduleIn	scheduleSpecifier	Schedule parameters.

Return information

The following table identifies the information returned by the createSchedule operation.

Table 4-5
Return Value

Type	Description
scheduleSpecifier	Details of a schedule.

Java example

Schedule creation requires the following two pieces of information:

- A string corresponding to the uri of the job associated with the schedule
- A ScheduleSpecifier object defining the parameters of the schedule

Typically, creating a schedule involves the following steps:

1. Create a Schedule object corresponding to the desired recurrence pattern. Available Schedule objects include OnceSchedule, HourlySchedule, DailySchedule, WeeklySchedule, MonthlySchedule, and YearlySchedule.
2. Supply the setScheduleEnabled method with a boolean indicating whether or not the schedule is enabled.
3. Define the label for the job version associated with the schedule using the setScheduledLabel method.
4. Set the recurrence interval using the setInterval method.
5. Specify the initial date and time for the schedule using the setStartDateTime method.
6. Specify the time of day the job should run using the setTimeOfDay method.
7. Create a ScheduleSpecifier object using the schedule object.
8. Define a string indicating the URI for the job.
9. Supply the ScheduleSpecifier object and the URI string to the createSchedule operation. The operation returns a ScheduleSpecifier object containing the schedule parameters, including the schedule identifier and the next scheduled date and time for execution.

   The following sample creates a weekly schedule for the job *Report* that runs at 1:31PM every Sunday. The seven booleans in the WeeklySchedule constructor correspond to the seven days of the week beginning with Monday. The schedule runs on any day with a value of *true*.

```
WeeklySchedule wSchedule = new WeeklySchedule (false, false, false, false, false, false, true);
wSchedule.setScheduleEnabled(true);
wSchedule.setScheduledLabel("Production");
wSchedule.setInterval(BigInteger.valueOf(1));
Calendar startTime = new Calendar();
startTime.set(2006, 11, 10, 13, 31, 0);
wSchedule.setStartDateTime(startTime);
wSchedule.setTimeOfDay(Time.valueOf("13:31:00");
ScheduleSpecifier scheduleIn = new ScheduleSpecifier(wSchedule);

String uri = "spsscr://pes_server:80/Jobs/Report";
ScheduleSpecifier scheduleOut = stub.createSchedule(uri, scheduleIn);
```

SOAP request example

Client invocation of the createSchedule operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
  xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <createSchedule xmlns="http://xml.spss.com/prms/remote">
   <jobLocationURI>spsscr://pes_server:80/Jobs/Report</jobLocationURI>
   <scheduleIn xmlns="http://xml.spss.com/prms">
    <schedule scheduleEnabled="true" scheduledLabel="Production" interval="1"
     startDateTime="2006-11-10T13:31:31.000-06:00" timeOfDay="13:31:00"
     monday="false" tuesday="false" wednesday="false" thursday="false"
     friday="false" saturday="false" sunday="true" xsi:type="weeklySchedule"/>
   </scheduleIn>
  </createSchedule>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a createSchedule operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <createScheduleResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:scheduleOut xsi:type="scheduleSpecifier" xmlns:ns1="http://xml.spss.com/prms">
    <ns1:schedule uuid="0a0a4a35781b73930000010ed217fdaa81c1"
     nextScheduledDateTime="2006-11-13T13:31:00.000-06:00" scheduleEnabled="true"
     scheduledLabel="Production" interval="1" startDateTime="2006-11-10T13:31:31.000-06:00"
     timeOfDay="13:31:00" monday="false" tuesday="false" wednesday="false" thursday="false"
     friday="false" saturday="false" sunday="true" xsi:type="weeklySchedule"/>
   </ns1:scheduleOut>
  </createScheduleResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The deleteJobExecutions operation

Deletes one or more job executions from the repository.

Input fields

The following table lists the input fields for the deleteJobExecutions operation.

Table 4-6
Fields for deleteJobExecutions

Field	Type/Valid Values	Description
jobExecutionIDs	string[]	Execution IDs of the jobs to delete.

Java example

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.

```
String [] executionID = new String[1];
executionID[0] = "0a0a4aac011937790000010d415272aa8ab4";
stub.deleteJobExecutions(executionID);
```

SOAP request example

Client invocation of the deleteJobExecutions operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <deleteJobExecutions xmlns="http://xml.spss.com/prms/remote">
   <jobExecutionIDs>0a0a4aac011937790000010d415272aa8ab4</jobExecutionIDs>
  </deleteJobExecutions>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a deleteJobExecutions operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <deleteJobExecutionsResponse xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

The deleteJobTrigger operation

Deletes a message-based job trigger from the repository.

Input fields

The following table lists the input fields for the deleteJobTrigger operation.

Table 4-7
Fields for deleteJobTrigger

Field	Type/Valid Values	Description
objectID	string	Identifier for a schedule or messageDrivenJob to be deleted.

Java example

To remove a job trigger from the system, supply the deleteJobTrigger operation with a string corresponding to the identifier for the trigger to delete.

```
String objectID = new String;
objectID = "0A0A4A3519BC8C6900000114D228EF1F8039";
stub.deleteJobTrigger(objectID);
```

SOAP request example

Client invocation of the deleteJobTrigger operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <deleteJobTrigger xmlns="http://xml.spss.com/prms/remote">
   <objectID>0A0A4A3519BC8C6900000114D228EF1F8039</objectID>
  </deleteJobTrigger>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a deleteJobTrigger operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <deleteJobTriggerResponse xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

The deleteJobTriggers operation

Deletes one or more message-based job triggers from the repository.

Input fields

The following table lists the input fields for the deleteJobTriggers operation.

Table 4-8
Fields for deleteJobTriggers

Field	Type/Valid Values	Description
objectID	string[]	Identifiers for the schedules or messageDrivenJobs to be deleted.

Java example

To delete multiple job triggers, supply the deleteJobTriggers operation with array of strings corresponding to the identifiers for the triggers to delete.

```
String [] objectID = new String[2];
objectID[0] = "0A0A4A3578F9BA1100000114D5CED52780C4";
objectID[1] = "0A0A4A3578F9BA1100000114D5CED52780CC";
stub.deleteJobTriggers(objectID);
```

SOAP request example

Client invocation of the deleteJobTriggers operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <deleteJobTriggers xmlns="http://xml.spss.com/prms/remote">
   <objectID>0A0A4A3578F9BA1100000114D5CED52780C4</objectID>
   <objectID>0A0A4A3578F9BA1100000114D5CED52780CC</objectID>
  </deleteJobTriggers>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a deleteJobTriggers operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <deleteJobTriggersResponse xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

The deleteSchedule operation

Deletes a schedule from the repository.

Input fields

The following table lists the input fields for the deleteSchedule operation.

Table 4-9
Fields for deleteSchedule

Field	Type/Valid Values	Description
scheduleID	string	Identifier for a schedule to be deleted.

Return information

The following table identifies the information returned by the deleteSchedule operation.

Table 4-10
Return Value

Type	Description
string	

Java example

To remove a schedule from the system, supply the deleteSchedule operation with a string corresponding to the identifier for the schedule to delete.

```
String [] scheduleID = new String[1];
scheduleID[0] = "ac140fd900072ffb0000010cak390g73uvm30";
stub.deleteSchedule(scheduleID);
```

SOAP request example

Client invocation of the deleteSchedule operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
   <deleteSchedule xmlns="http://xml.spss.com/prms/remote">
    <scheduleID>ac140fd900072ffb0000010cak390g73uvm30</scheduleID>
   </deleteSchedule>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a deleteSchedule operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <deleteScheduleResponse xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

The deleteSchedules operation

Deletes one or more schedules from the repository.

Input fields

The following table lists the input fields for the deleteSchedules operation.

Table 4-11
Fields for deleteSchedules

Field	Type/Valid Values	Description
scheduleID	string[]	Identifiers for the schedules to be deleted.

Java example

To delete multiple schedules, supply the deleteSchedules operation with array of strings corresponding to the identifiers for the schedules to delete.

```
String [] scheduleID = new String[2];
scheduleID[0] = "ac140fd900072ffb0000010cak390g73uvm30";
scheduleID[1] = "bc740fd925072fgb0005518cei30c8700vnvg";
stub.deleteSchedules(scheduleID);
```

SOAP request example

Client invocation of the deleteSchedules operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
   <deleteSchedules xmlns="http://xml.spss.com/prms/remote">
    <scheduleID>ac140fd900072ffb0000010cak390g73uvm30</scheduleID>
    <scheduleID>bc740fd925072fgb0005518cei30c8700vnvg</scheduleID>
   </deleteSchedules>
 </soapenv:Body>
</soapenv:Envelope>
```

The finalizeRemoteWork operation

Input fields

The following table lists the input fields for the finalizeRemoteWork operation.

Table 4-12
Fields for finalizeRemoteWork

Field	Type/Valid Values	Description
remoteWorkCompletion	remoteWorkCompletion	Used internally.

The getCustomEventTypes operation

Returns all custom event types in the system.

Return information

The following table identifies the information returned by the getCustomEventTypes operation.

Table 4-13
Return Value

Type	Description
string[]	List of the customEvent types allowed in the system.

Java example

The following function checks whether or not a specified custom event type is supported. Using the service stub, the getCustomEventTypes operation returns an array of strings corresponding to the valid custom event types, from which a HashSet is created. If the specified custom type is in the set, the function returns *true*.

```
public boolean isSupportedCustomType(String customType)
  throws RemoteException, IOException, ServiceException {
  if (i_supportedCustomTypes == null) {
    String[] supportedTypes = stub.getCustomEventTypes();
    i_supportedCustomTypes = new HashSet(Arrays.asList(supportedTypes));
  }
  return i_supportedCustomTypes.contains(customType);
}
```

SOAP request example

Client invocation of the getCustomEventTypes operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
```

```
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
  xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <getCustomEventTypes xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getCustomEventTypes operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <getCustomEventTypesResponse xmlns="http://xml.spss.com/prms/remote">
   <customEventType>ProcessManagement.MessageDrivenJobstep</customEventType>
  </getCustomEventTypesResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The getExecutionDetails operation

Retrieves the metadata and results for a specified execution.

Input fields

The following table lists the input fields for the getExecutionDetails operation.

Table 4-14
Fields for getExecutionDetails

Field	Type/Valid Values	Description
executionID	string	The execution ID of a job, job step, or job iteration.
deliveryType	deliveryType	The format for delivering the log file for the execution. Valid values include: MIME (Multipurpose Internet Mail Extensions), DIME (Direct Internet Message Encapsulation), STRING, and NONE.

Return information

The following table identifies the information returned by the getExecutionDetails operation.

Table 4-15
Return Value

Type	Description
executionDetails	Records the execution instance of a Event. Every time an Event is executed, a new EventExecution is created in the repository.

Java example

Use the getExecutionDetails operation to return an executionDetails object containing information about an execution. Supply the operation with the ID for the execution and the format for log file delivery.

Accessor methods for the executionDetails object provide specific pieces of information for the execution. For example, the getEventName method returns the event name. Moreover, the getStartDateTime and getEndDateTime methods return the start and end times, respectively. The getExecutionSuccess and getExecutionState methods provide details about the success or failure of the execution.

```
String uri=
 "spsscr://pes_server:80/?id=0a0a4aac011937790000010d415272aa824a#m.1:2006-08-24%2013:16:21.069";
String executionID = stub.submitJob(uri, true);
ExecutionDetails exDetails = stub.getExecutionDetails(executionID,DeliveryType.STRING);

System.out.println("Event name: " + exDetails.getEventName());
Calendar start = exDetails.getStartDateTime();
System.out.println("Start time: " + start.getTime());
Calendar end = exDetails.getEndDateTime();
System.out.println("End time: " + end.getTime());
if (exDetails.getExecutionSuccess()) {
  System.out.println("Event succeeded.");
} else {
  System.out.println("Event failed.");
}
EventExecutionState state = exDetails.getExecutionState()
```

```
System.out.println("Execution state: " + state.toString());
```

SOAP request example

Client invocation of the getExecutionDetails operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <getExecutionDetails xmlns="http://xml.spss.com/prms/remote">
   <jobExecutionID>0a0a4aac011937790000010d415272aa8ab4</jobExecutionID>
   <deliveryType xmlns="http://xml.spss.com/prms">STRING</deliveryType>
  </getExecutionDetails>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getExecutionDetails operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <getExecutionDetailsResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:executionDetails uuid="0a0a4aac011937790000010d415272aa8ab4"
    executionState="ended" queuedDateTime="2006-08-25T10:01:15.780-05:00"
    startDateTime="2006-08-25T10:01:16.157-05:00"
    endDateTime="2006-08-25T10:02:04.047-05:00" completionCode="0"
    executionSuccess="true" eventUuid="0a0a4aac011937790000010d415272aa836a"
    eventName="testSPSS" notificationEnabled="true"
    xmlns:ns1="http://xml.spss.com/prms"/>
  </getExecutionDetailsResponse>
 </soapenv:Body>
```

```
</soapenv:Envelope>
```

The getJobParameters operation

Retrieves the variable definitions for a specified job.

Input fields

The following table lists the input fields for the getJobParameters operation.

Table 4-16
Fields for getJobParameters

Field	Type/Valid Values	Description
jobLocationURI	string	URI for a job.

Return information

The following table identifies the information returned by the getJobParameters operation.

Table 4-17
Return Value

Type	Description
jobParameters	The parameters for this job

Java example

To access the variable definitions for a job, supply the getJobParameters operation with a string denoting the uniform resource identifier for the job. The operation returns an array of JobParameter objects containing the characteristics for each job variable. Use the getName, getDescription, and getDefault methods to extract specific settings.

```
String uri = "spsscr:///Jobs/My%20Job#m.4:2009-03-17%2008:53:42.482";
JobParameter[] parameterArray = stub.getJobParameters(uri);

System.out.println("PARAMETER NAME\tDESCRIPTION\tDEFAULT\n");
for (int i = 0; i < parameterArray.length; i++) {
  System.out.println(parameterArray[i].getName() + "\t" +
    parameterArray[i].getDescription() + "\t" +
    parameterArray[i].getDefault());
}
```

SOAP request example

Client invocation of the getJobParameters operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
  <soapenv:Body>
    <getJobParameters xmlns="http://xml.spss.com/prms/remote">
      <jobLocationURI>SPSSCR:///Jobs/My%20Job#m.4:2009-03-17%2008:53:42.482</jobLocationURI>
    </getJobParameters>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getJobParameters operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getJobParametersResponse xmlns="http://xml.spss.com/prms/remote">
      <parms>
        <parameter name="command" default="dir" description="Command to run"
          xmlns="http://xml.spss.com/prms"/>
        <parameter name="options" default="c:" description="Options for the command"
          xmlns="http://xml.spss.com/prms"/>
      </parms>
    </getJobParametersResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The getJobStepChildExecutions operation

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.

Input fields

The following table lists the input fields for the getJobStepChildExecutions operation.

Table 4-18
Fields for getJobStepChildExecutions

Field	Type/Valid Values	Description
jobStepExecutionID	string	The execution ID of a job step.
deliveryType	deliveryType	The format for delivering the log file for the execution. Valid values include: MIME (Multipurpose Internet Mail Extensions), DIME (Direct Internet Message Encapsulation), STRING, and NONE.

Return information

The following table identifies the information returned by the getJobStepChildExecutions operation.

Table 4-19
Return Value

Type	Description
jobStepChildExecutions	The event executions for the children of the job step.

Java example

Use the getJobStepChildExecutions operation to return a JobStepChildExecutions object from which an array of ExecutionDetails objects containing information about child executions for job steps can be obtained. Supply the operation with the ID for the job step execution and the format for log file delivery.

Accessor methods for the ExecutionDetails objects provide specific pieces of information for the step executions. For example, the getEventName method returns the event name. Moreover, the getStartDateTime and getEndDateTime methods return the start and end times, respectively. The getExecutionSuccess and getExecutionState methods provide details about the success or failure of the step execution.

```
String stepExeID = new String;
stepExeID = "0a0a4a357ce2d48300000110597d9efb80db";
JobStepChildExecutions childExecutions = stub.getJobStepChildExecutions(stepExeID);
ExecutionDetails[] exDetails = childExecutions.getIterations();
for (int j = 0; j < exDetails.length; j++) {
  System.out.println("Event name: " + exDetails[j].getEventName());
  Calendar start = exDetails[j].getStartDateTime();
  System.out.println("Start time: " + start.getTime());
  Calendar end = exDetails[j].getEndDateTime();
  System.out.println("End time: " + end.getTime());
  if (exDetails[j].getExecutionSuccess()) {
    System.out.println("Event succeeded.");
  } else {
    System.out.println("Event failed.");
  }
  EventExecutionState state = exDetails[j].getExecutionState()
  System.out.println("Execution state: " + state.toString());
```

}

SOAP request example

Client invocation of the getJobStepChildExecutions operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <getJobStepChildExecutions xmlns="http://xml.spss.com/prms/remote">
   <jobStepExecutionID>0a0a4a357ce2d48300000110597d9efb80db</jobStepExecutionID>
   <deliveryType xmlns="http://xml.spss.com/prms">STRING</deliveryType>
  </getJobStepChildExecutions>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getJobStepChildExecutions operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <getJobStepChildExecutionsResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:jobStepChildExecutions xmlns:ns1="http://xml.spss.com/prms">
    <ns1:iterations uuid="0a0a4a357ce2d48300000110597d9efb80dc" executionState="ended"
     queuedDateTime="2007-01-25T08:45:46.390-06:00"
     startDateTime="2007-01-25T08:45:46.467-06:00" endDateTime="2007-01-25T08:45:53.110-06:00"
     completionCode="0" executionSuccess="true"
     eventUuid="0a0a4a355a576f48000001104fe9b57482b4" eventName="iterStep2.rptdesign_step"
     notificationEnabled="true" executionWarning="false" hasIterations="false">
     <ns1:artifactLocation>
      spsscr://pes_server:80/?id=0a0a4a355a576f48000001104fe9b5748049#m.2:2007-01-25%2008:45:50.148
     </ns1:artifactLocation>
```

```
      </ns1:iterations>
      <ns1:iterations uuid="0a0a4a357ce2d48300000110597d9efb80dd" executionState="ended"
       queuedDateTime="2007-01-25T08:45:46.403-06:00"
       startDateTime="2007-01-25T08:45:46.467-06:00" endDateTime="2007-01-25T08:45:52.937-06:00"
       completionCode="0" executionSuccess="true"
       eventUuid="0a0a4a355a576f48000001104fe9b57482b4" eventName="iterStep2.rptdesign_step"
       notificationEnabled="true" executionWarning="false" hasIterations="false">
       <ns1:artifactLocation>
        spsscr://pes_server:80/?id=0a0a4a355a576f48000001104fe9b5748048#m.3:2007-01-25%2008:45:50.023
       </ns1:artifactLocation>
      </ns1:iterations>
     </ns1:jobStepChildExecutions>
    </getJobStepChildExecutionsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The getJobStepExecutions operation

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

Input fields

The following table lists the input fields for the getJobStepExecutions operation.

Table 4-20
Fields for getJobStepExecutions

Field	Type/Valid Values	Description
jobExecutionID	string	The execution ID of a job.
deliveryType	deliveryType	The format for delivering the log file for the execution. Valid values include: MIME (Multipurpose Internet Mail Extensions), DIME (Direct Internet Message Encapsulation), STRING, and NONE.

Return information

The following table identifies the information returned by the getJobStepExecutions operation.

Table 4-21
Return Value

Type	Description
executionDetails[]	Records the execution instance of a Event. Every time an Event is executed, a new EventExecution is created in the repository.

Java example

Use the getJobStepExecutions operation to return an array of executionDetails objects containing information about executions for job steps. Supply the operation with the ID for the job execution and the format for log file delivery.

Accessor methods for the executionDetails objects provide specific pieces of information for the step executions. For example, the getEventName method returns the event name. Moreover, the getStartDateTime and getEndDateTime methods return the start and end times, respectively. The getExecutionSuccess and getExecutionState methods provide details about the success or failure of the step execution.

```
String [] executionID = new String[1];
executionID[0] = "0a0a4aac011937790000010d415272aa8ab4";
ExecutionDetails[] exDetails = stub.getJobStepExecutions(executionID);
for (int j = 0; j < exDetails.length; j++) {
  System.out.println("Event name: " + exDetails[j].getEventName());
  Calendar start = exDetails[j].getStartDateTime();
  System.out.println("Start time: " + start.getTime());
  Calendar end = exDetails[j].getEndDateTime();
  System.out.println("End time: " + end.getTime());
  if (exDetails[j].getExecutionSuccess()) {
    System.out.println("Event succeeded.");
  } else {
    System.out.println("Event failed.");
  }
  EventExecutionState state = exDetails[j].getExecutionState()
  System.out.println("Execution state: " + state.toString());
}
```

SOAP request example

Client invocation of the getJobStepExecutions operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
```

```
 <soapenv:Body>
  <getJobStepExecutions xmlns="http://xml.spss.com/prms/remote">
   <jobExecutionID>0a0a4aac011937790000010d415272aa8ab4</jobExecutionID>
   <deliveryType xmlns="http://xml.spss.com/prms">STRING</deliveryType>
  </getJobStepExecutions>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getJobStepExecutions operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <getJobStepExecutionsResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:executionDetails uuid="0a0a4aac011937790000010d415272aa8ab5"
    executionState="ended" queuedDateTime="2006-08-25T10:01:15.877-05:00"
    startDateTime="2006-08-25T10:01:17.593-05:00"
    endDateTime="2006-08-25T10:02:02.813-05:00" completionCode="0"
    executionSuccess="true" eventUuid="0a0a4aac011937790000010d415272aa836c"
    eventName="tree_model.sps_step" notificationEnabled="true"
    xmlns:ns1="http://xml.spss.com/prms">
    <ns1:artifactLocation>
     spsscr://pes_server:80/?id=0a0a4aac011937790000010d415272aa848e#m.4:2006-08-25%2010:02:02.468
    </ns1:artifactLocation>
    <ns1:log>
     <ns1:logAsString>—PRMS —
     spssb -f syntax.sps -type html  -out &amp;quot;tree_model.html&amp;quot; -LC_ALL en_US -language en
     The process terminated with the exit code 0.
     Adding artifact spsscr:/Jobs/tree_model.zip.
     —STDOUT —
     —STDERR —
     </ns1:logAsString>
    </ns1:log>
   </ns1:executionDetails>
  </getJobStepExecutionsResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The getMessageDrivenJob operation

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

Input fields

The following table lists the input fields for the getMessageDrivenJob operation.

Table 4-22
Fields for getMessageDrivenJob

Field	Type/Valid Values	Description
messageDrivenJobID	string	Identifier for a messageDrivenJob.

Return information

The following table identifies the information returned by the getMessageDrivenJob operation.

Table 4-23
Return Value

Type	Description
messageDrivenJobSpecifier	Details of a Message Drive Job Schedule.

Java example

To retrieve a message driven job from the system, supply the getMessageDrivenJob operation with a string corresponding to the identifier for the job. The operation returns a MessageDrivenJobSpecifier object, from which specific details relating to the job can be accessed.

The following sample uses the getMessageDrivenJob method to return the MessageDrivenJob object from the MessageDrivenJobSpecifier. The getLabel method returns the label for the job. If the message driven job is enabled, as determined by the getEnabled method, the getCredentialID and getMessageText methods return the associated credentials identifier and filter text respectively.

```
String [] mdJobID = new String[1];
mdJobID[0] = "0a0a4a3519895570000000114a76f97678181";
MessageDrivenJobSpecifier mdJobSpec = stub.getMessageDrivenJob(mdJobID);
MessageDrivenJob mdJob = mdJobSpec.getMessageDrivenJob()
System.out.println("Message driven job label: " + mdJob.getLabel());
if (mdJob.getEnabled()) {
  System.out.println("Credentials: " + mdJob.getCredentialID());
  System.out.println("Message text: " + mdJob.getMessageText());
}
System.out.println("Reserved: " + mdJobSpec.getReserved());
```

SOAP request example

Client invocation of the getMessageDrivenJob operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
```

```
    <wsse:UsernameToken>
      <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
      <wsse:Password xsi:type="xsd:string">password</wsse:Password>
    </wsse:UsernameToken>
   </wsse:Security>
   <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
    xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
  </soapenv:Header>
  <soapenv:Body>
   <getMessageDrivenJob xmlns="http://xml.spss.com/prms/remote">
    <messageDrivenJobID>0a0a4a3519895570000000114a76f97678181</messageDrivenJobID>
   </getMessageDrivenJob>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getMessageDrivenJob operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <getMessageDrivenJobResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:messageDrivenJobOut xmlns:ns1="http://xml.spss.com/prms">
    <ns1:messageDrivenJob uuid="0a0a4a3519895570000000114a76f97678181" enabled="true"
     label="LATEST" credentialID="0a0a4a356f526f1d0000011450a1669680af" messageText="My Text"/>
   </ns1:messageDrivenJobOut>
  </getMessageDrivenJobResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The getMessageDrivenJobs operation

Returns information about the message-based schedules for a specified job.

Input fields

The following table lists the input fields for the getMessageDrivenJobs operation.

Table 4-24
Fields for getMessageDrivenJobs

Field	Type/Valid Values	Description
jobLocationURI	string	URI for a job.

Return information

The following table identifies the information returned by the getMessageDrivenJobs operation.

Table 4-25
Return Value

Type	Description
messageDrivenJobList	A list of messageDrivenJobs.

The getSchedule operation

Returns information about the schedule corresponding to a specified identifier.

Input fields

The following table lists the input fields for the getSchedule operation.

Table 4-26
Fields for getSchedule

Field	Type/Valid Values	Description
scheduleID	string	Identifier for a schedule.

Return information

The following table identifies the information returned by the getSchedule operation.

Table 4-27
Return Value

Type	Description
scheduleSpecifier	Details of a schedule.

Java example

To retrieve a schedule from the system, supply the getSchedule operation with a string corresponding to the identifier for the schedule. The operation returns a ScheduleSpecifier object, from which specific details relating to the schedule can be accessed.

The following sample uses the getSchedule method to return the Schedule object from the ScheduleSpecifier. The getScheduledLabel method returns the label for the schedule. If the schedule is enabled, as determined by the getScheduleEnabled method, the getNextScheduledDateTime method returns the next scheduled execution time as a Calendar object.

```
String [] scheduleID = new String[1];
scheduleID[0] = "ac140fd900072ffb0000010cak390g73uvm30";
ScheduleSpecifier scheduleSpec = stub.getSchedule(scheduleID);
Schedule sched = scheduleSpec.getSchedule()
System.out.println("Schedule label: " + sched.getScheduledLabel());
if (sched.getScheduleEnabled()) {
  Calendar nextTime = sched.getNextScheduledDateTime();
```

```
  System.out.println("Next execution time: " + nextTime.getTime());
}
System.out.println("Reserved: " + scheduleSpec.getReserved());
```

SOAP request example

Client invocation of the getSchedule operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
   <getSchedule xmlns="http://xml.spss.com/prms/remote">
    <scheduleID>0a0a4a35781b73930000010ed217fdaa8173</scheduleID>
   </getSchedule>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getSchedule operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <getScheduleResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:scheduleOut xsi:type="scheduleSpecifier" xmlns:ns1="http://xml.spss.com/prms">
    <ns1:schedule uuid="0a0a4a35781b73930000010ed217fdaa8173"
     nextScheduledDateTime="2006-11-13T13:31:00.000-06:00" scheduleEnabled="true"
     scheduledLabel="LATEST" interval="1" startDateTime="2006-11-10T13:31:31.000-06:00"
     timeOfDay="13:31:00" monday="true" tuesday="false" wednesday="false" thursday="false"
     friday="false" saturday="false" sunday="false" xsi:type="weeklySchedule"/>
   </ns1:scheduleOut>
  </getScheduleResponse>
```

```
 </soapenv:Body>
</soapenv:Envelope>
```

The getSchedules operation

Retrieves all schedules associated with a specified job.

Input fields

The following table lists the input fields for the getSchedules operation.

Table 4-28
Fields for getSchedules

Field	Type/Valid Values	Description
jobLocationURI	string	URI for a job.

Return information

The following table identifies the information returned by the getSchedules operation.

Table 4-29
Return Value

Type	Description
scheduleList	A list of schedules.

Java example

To retrieve all schedules associated with a job, supply the getSchedules operation with the URI for the job. Use the getSchedule method for the returned ScheduleList object to access individual schedules.

The following sample loops over all schedules associated with a job, reporting the next scheduled execution time for each. The getScheduledLabel method returns the label for a schedule. If a schedule is enabled, as determined by the getScheduleEnabled method, the getNextScheduledDateTime method returns the next scheduled execution time as a Calendar object.

```
String uri=
  "spsscr://localhost/?id=ac140fd900072ffb0000010c5f86c39b84f7#m.2:2006-07-12%2008:20:14.625";
ScheduleList schedList = stub.getSchedules(uri);
System.out.println("Reserved: " + schedList.getReserved());
Schedule[] schedArray = schedList.getSchedule();
for (int j = 0; j < schedArray.length; j++) {
   System.out.println("Schedule label: " + schedArray[j].getScheduledLabel());
   if (schedArray[j].getScheduleEnabled()) {
      Calendar nextTime = schedArray[j].getNextScheduledDateTime();
   System.out.println("Next execution time: " + nextTime.getTime());
   }
}
```

SOAP request example

Client invocation of the getSchedules operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">admin</wsse:Username>
    <wsse:Password xsi:type="xsd:string">spss</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
    <getSchedules xmlns="http://xml.spss.com/prms/remote">
      <jobLocationURI>spsscr://pes_server:80/Jobs/Analysis</jobLocationURI>
    </getSchedules>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getSchedules operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <getSchedulesResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:scheduleList xmlns:ns1="http://xml.spss.com/prms">
    <ns1:schedule uuid="0a0a4a35781b73930000010ed217fdaa8173"
     nextScheduledDateTime="2006-11-13T13:31:00.000-06:00" scheduleEnabled="true"
     scheduledLabel="LATEST" interval="1" startDateTime="2006-11-10T13:31:31.000-06:00"
     timeOfDay="13:31:00" monday="true" tuesday="false" wednesday="false" thursday="false"
     friday="false" saturday="false" sunday="false" xsi:type="weeklySchedule"/>
    <ns1:schedule uuid="0a0a4a35781b73930000010ed217fdaa8175"
     nextScheduledDateTime="2006-12-01T13:31:00.000-06:00" scheduleEnabled="true"
     scheduledLabel="LATEST" interval="1" startDateTime="2006-11-10T13:31:49.000-06:00"
     timeOfDay="13:31:00" dayOfMonth="1" xsi:type="monthlySchedule"/>
   </ns1:scheduleList>
```

```
  </getSchedulesResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The getVersion operation

Returns the version number of the service.

Return information

The following table identifies the information returned by the getVersion operation.

Table 4-30
Return Value

Type	Description
string	The version number of the web service.

Java example

To access the version number of the service, call the getVersion operation from the service stub.

```
System.out.println("Service Version =  " + stub.getVersion());
```

SOAP request example

Client invocation of the getVersion operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
  xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <getVersion xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getVersion operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <getVersionResponse xmlns="http://xml.spss.com/prms/remote">
   <version>4.20.000</version>
  </getVersionResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The getWorkTypes operation

Retrieves a list of the types of work that can be included in an event cluster. Examples of work types include:

- IBM® SPSS® Modeler streams
- IBM® SPSS® Statistics syntax
- SAS syntax

Return information

The following table identifies the information returned by the getWorkTypes operation.

Table 4-31
Return Value

Type	Description
string[]	List of the work types allowed in the system.

Java example

The following function checks whether or not a specified work type is supported. Using the service stub, the getWorkTypes operation returns an array of strings corresponding to the valid work types, from which a HashSet is created. If the specified work type is in the set, the function returns *true*.

```
public boolean isSupportedWorkType(String workType)
  throws RemoteException, IOException, ServiceException {
  if (i_supportedWorkTypes == null) {
    String[] supportedTypes = stub.getWorkTypes();
    i_supportedWorkTypes = new HashSet(Arrays.asList(supportedTypes));
  }
  return i_supportedWorkTypes.contains(workType);
}
```

SOAP request example

Client invocation of the getWorkTypes operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <getWorkTypes xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getWorkTypes operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <getWorkTypesResponse xmlns="http://xml.spss.com/prms/remote">
   <workType>Clementine.ClementineStreamWork</workType>
   <workType>ModelManagement.ExecutableContentWork</workType>
   <workType>ModelManagement.SASSyntaxWork</workType>
   <workType>ProcessManagement.RunJavaClass</workType>
   <workType>ProcessManagement.CleanupExecutions</workType>
   <workType>ProcessManagement.WindowsCommandWork</workType>
   <workType>ModelManagement.SPSSSyntaxWork</workType>
   <workType>Reporting.BIRTReportWork</workType>
   <workType>Scenario.ScenarioWork</workType>
  </getWorkTypesResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The handleMessageDomainChanged operation

Restarts message listeners for existing subscribers to a message domain, allowing the subscribers to recognize updated message domain settings. If this operation is not used, any existing subscribers will continue to use the previous domain settings.

Input fields

The following table lists the input fields for the handleMessageDomainChanged operation.

Table 4-32
Fields for handleMessageDomainChanged

Field	Type/Valid Values	Description
msgDomainID	string	Identifier for the message Domain that changed.

Java example

To restart listeners for subscribers, supply the handleMessageDomainChanged operation with a string corresponding to the identifier for the message domain that has been modified.

```
String domainID = "0a0a4a3578f9ba1100000114d5ced52780c8";
stub.handleMessageDomainChanged(domainID);
```

SOAP request example

Client invocation of the handleMessageDomainChanged operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
  xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <handleMessageDomainChanged xmlns="http://xml.spss.com/prms/remote">
   <msgDomainID xmlns="http://xml.spss.com/prms">
    0a0a4a3578f9ba1100000114d5ced52780c8
   </msgDomainID>
  </handleMessageDomainChanged>
 </soapenv:Body>
```

```
</soapenv:Envelope>
```

SOAP response example

The server responds to a handleMessageDomainChanged operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <handleMessageDomainChangedResponse xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

The queryAllSchedules operation

Returns all time-based and message-based schedules in the system meeting specified criteria. The information returned by the operation includes the fields identified in the following table.

Table 4-33
Return Fields

Field Name	Description
job.objid	Identifier for the job
jobPath	Path and file name for the job
fileversion.marker	Version marker associated with this scheduled job
usesfiledep.label	Version label specified on the schedule
schedule.objid	Identifier of the schedule
messagedriven.objid	Identifier of the message-driven job
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	Credential name used to run the job

The *trigger.summary* and the *next.trigger* fields will have mixed data (timestamps and strings) if the system includes both time-based and message-based schedules. The client application calling the operation needs to check each type to determine how to handle the information.

Input fields

The following table lists the input fields for the queryAllSchedules operation.

Table 4-34
Fields for queryAllSchedules

Field	Type/Valid Values	Description
jobTriggerCriterion	jobTriggerCriterion	Definition of the query parameters.

Return information

The following table identifies the information returned by the queryAllSchedules operation.

Table 4-35
Return Value

Type	Description
pageResult	Results of a search request to the search2.5 mechanism. This may either be an initial search based on any criterion, or a subsequent search for another page.

Java example

Querying the repository for all schedules typically involves the following steps:

1. Create a JobTriggerCriterion object.
2. Create a ReturnSpecifier object and define the return settings.
3. Create a PageSelector object and define the selector settings.
4. Assign the ReturnSpecifier and PageSelector objects to the JobTriggerCriterion object.
5. Supply the JobTriggerCriterion object to the queryAllSchedules operation.

The following sample retrieves all schedules in the system, sending the returned fields and their values for each schedule to the standard output.

```
JobTriggerCriterion jobTriggerCrit = new JobTriggerCriterion();

ReturnSpecifier retSpec = new ReturnSpecifier();
retSpec.setSortOrder(ReturnSpecifierSortOrder.descending);
BigInteger pSize = new BigInteger("11");
retSpec.setPageSize(pSize);
jobTriggerCrit.setReturnSpecifier(retSpec);

PageSelector pageSel = new PageSelector();
JobTriggerFilter filter = new JobTriggerFilter();
filter.setJobLocationURI("SPSSCR://pes_server/Monthly%20Status");
pageSel.setFilter(filter);
jobTriggerCrit.setPageSelector(pageSel);

PageResult qResult = stub.queryAllSchedules(jobTriggerCrit);

ColumnType[] resultColumn = qResult.getColumn();
Row[] resultRow = qrResult.getRow();
```

```
Cell[] resultCell = new Cell[resultColumn.length];
for (int i = 0; i < resultRow.length; i++) {
  System.out.println("Hit " + i);
  resultCell = resultRow[i].getCell();
  for (int j = 0; j < resultCell.length; j++) {
    System.out.println(resultColumn[j].getFieldName() + "=" +
      resultCell[j].getValue().getDisplay());
  }
  System.out.println("\n");
}
```

SOAP request example

Client invocation of the queryAllSchedules operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
   soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username>validUser</wsse:Username>
    <wsse:Password>password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
   soapenv:mustUnderstand="0" xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0,
  en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <queryAllSchedules xmlns="http://xml.spss.com/prms/remote">
   <jobTriggerCriterion xmlns="http://xml.spss.com/prms">
    <returnSpecifier sortOrder="descending" pageSize="14"/>
    <pageSelector>
     <filter xsi:type="jobTriggerFilter">
      <jobLocationURI>SPSSCR://pes_server/Jobs/Results</jobLocationURI>
     </filter>
    </pageSelector>
   </jobTriggerCriterion>
  </queryAllSchedules>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a queryAllSchedules operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <queryAllSchedulesResponse xmlns="http://xml.spss.com/prms/remote">
   <pageResult hitCount="2" pageSize="14" pageNumber="1" clientKey="446cb951:116153f80a6:-7df9"
    sortColumn="job.objid" sortOrder="ascending" xmlns="http://www.spss.com/pes/pager">
    <column display="job.objid" fieldName="job.objid" colType="string"/>
    <column display="jobPath" fieldName="jobPath" colType="string"/>
    <column display="fileversion.marker" fieldName="fileversion.marker" colType="string"/>
    <column display="usesfiledep.label" fieldName="usesfiledep.label" colType="string"/>
    <column display="schedule.objid" fieldName="schedule.objid" colType="string"/>
    <column display="messagedriven.objid" fieldName="messagedriven.objid" colType="string"/>
    <column display="trigger.summary" fieldName="trigger.summary" colType="string"/>
    <column display="next.trigger" fieldName="next.trigger" colType="string"/>
    <column display="lastrun.startDateTime" fieldName="lastrun.startDateTime" colType="stamp"/>
    <column display="lastrun.status" fieldName="lastrun.status" colType="string"/>
    <column display="credentials.name" fieldName="credentials.name" colType="string"/>
    <row rowNumber="1">
     <cell>
      <value>
       <display xmlns="">0A0A4A352DEF327800000115CDABFC0A84EA</display>
      </value>
     </cell>
     <cell><value><display xmlns="">/Jobs/Results</display></value></cell>
     <cell><value><display xmlns="">0:2007-10-23 12:14:01.71</display></value></cell>
     <cell><value><display xmlns="">LATEST</display></value></cell>
     <cell><value><display xmlns=""/><rawString xmlns=""/></value></cell>
     <cell>
      <value>
       <display xmlns="">0A0A4A35446CB95100000116153F80A681B6</display>
      </value>
     </cell>
     <cell><value><display xmlns="">Trigger</display></value></cell>
     <cell><value><display xmlns=""/><rawString xmlns=""/></value>
     </cell>
     <cell>
      <value>
       <display xmlns=""/><rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
      </value>
     </cell>
     <cell><value><display xmlns="">Not Available</display></value></cell>
     <cell><value><display xmlns="">kk-admin</display></value></cell>
    </row>
    <row rowNumber="2">
     <cell>
      <value>
       <display xmlns="">0A0A4A352DEF327800000115CDABFC0A84EA</display>
      </value>
     </cell>
     <cell><value><display xmlns="">/Jobs/Results</display></value></cell>
```

```
      <cell><value><display xmlns="">0:2007-10-23 12:14:01.71</display></value></cell>
      <cell><value><display xmlns="">Test</display></value></cell>
      <cell>
       <value>
        <display xmlns="">0A0A4A35446CB95100000116153F80A681C7</display>
       </value>
      </cell>
      <cell><value><display xmlns=""/><rawString xmlns=""/></value></cell>
      <cell><value><display xmlns="">Weekly</display></value></cell>
      <cell>
       <value>
        <display xmlns="">2007-11-12T14:48:00-0600</display>
        <rawStamp xmlns="">2007-11-12T14:48:00.000-06:00</rawStamp>
       </value>
      </cell>
      <cell>
       <value>
        <display xmlns=""/><rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
       </value>
      </cell>
      <cell><value><display xmlns="">Not Available</display></value></cell>
      <cell><value><display xmlns="">kk</display></value></cell>
     </row>
     <navigator>
      <page display="1" selector="1" current="true" xmlns=""/>
     </navigator>
    </pageResult>
   </queryAllSchedulesResponse> </soapenv:Body>
</soapenv:Envelope>
```

The queryExecutions operation

Returns all executions in the system meeting specified criteria.

Input fields

The following table lists the input fields for the queryExecutions operation.

Table 4-36
Fields for queryExecutions

Field	Type/Valid Values	Description
executionCriterion	executionCriterion	Definition of the query parameters.

Return information

The following table identifies the information returned by the queryExecutions operation.

Table 4-37
Return Value

Type	Description
pageResult	Results of a search request to the search2.5 mechanism. This may either be an initial search based on any criterion, or a subsequent search for another page.

Java example

Querying the repository for executions typically involves the following steps:

1. Create an ExecutionCriterion object.
2. Create a ReturnSpecifier object and define the return settings.
3. Create a PageSelector object and define the selector settings.
4. Assign the ReturnSpecifier and PageSelector objects to the ExecutionCriterion object.
5. Supply the ExecutionCriterion object to the queryExecutions operation.

The following sample retrieves all executions in the system, sending the returned fields and their values for each execution to the standard output.

```
ExecutionCriterion execCrit = new ExecutionCriterion();
PageResult qResult = stub.queryExecutions(execCrit);
ColumnType[] resultColumn = qResult.getColumn();
Row[] resultRow = qrResult.getRow();
Cell[] resultCell = new Cell[resultColumn.length];
for (int i = 0; i < resultRow.length; i++) {
  System.out.println("Hit " + i);
  resultCell = resultRow[i].getCell();
  for (int j = 0; j < resultCell.length; j++) {
    System.out.println(resultColumn[j].getFieldName() + "=" +
      resultCell[j].getValue().getDisplay());
  }
  System.out.println("\n");
}
```

SOAP request example

Client invocation of the queryExecutions operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
```

```
      <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
      <wsse:Password xsi:type="xsd:string">password</wsse:Password>
    </wsse:UsernameToken>
   </wsse:Security>
   <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
    xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
  </soapenv:Header>
  <soapenv:Body>
   <queryExecutions xmlns="http://xml.spss.com/prms/remote">
    <executionCriterion xmlns="http://xml.spss.com/prms"/>
   </queryExecutions>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a queryExecutions operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <queryExecutionsResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:pageResult hitCount="3" pageSize="255" pageNumber="1"
    clientKey="29f5f2b0:10ed2672a37:-7df4" sortColumn="file.objid" sortOrder="ascending"
    xmlns:ns1="http://www.spss.com/pes/pager">
    <ns1:column display="file.objid" fieldName="file.objid" colType="string"/>
    <ns1:column display="file.path" fieldName="file.path" colType="string"/>
    <ns1:column display="fileversion.marker" fieldName="fileversion.marker" colType="string"/>
    <ns1:column display="fileversion.label" fieldName="fileversion.label" colType="string"/>
    <ns1:column display="eventexe.objId" fieldName="eventexe.objId" colType="string"/>
    <ns1:column display="eventexe.state" fieldName="eventexe.state" colType="string"/>
    <ns1:column display="eventexe.completionCode" fieldName="eventexe.completionCode"
     colType="string"/>
    <ns1:column display="eventexe.startDateTime" fieldName="eventexe.startDateTime"
     colType="string"/>
    <ns1:column display="eventexe.endDateTime" fieldName="eventexe.endDateTime" colType="string"/>
    <ns1:column display="eventexe.queuedDateTime" fieldName="eventexe.queuedDateTime"
     colType="string"/>
    <ns1:row rowNumber="1">
     <ns1:cell>
      <ns1:value><display xmlns="">0A0B32E0CFC420760000010EC3B57CAF81C3</display>
       <rawString xmlns=""/></ns1:value>
     </ns1:cell>
     <ns1:cell>
      <ns1:value><display xmlns="">/Sales Data Analysis</display>
       <rawString xmlns=""/></ns1:value>
     </ns1:cell>
     <ns1:cell>
```

```
  <ns1:value><display xmlns="">1:2006-11-10 12:37:08.385</display>
    <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0A0B32E029F5F2B00000010ED2672A37814D</display>
    <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">4</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 12:37:16.927</display>
    <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 12:40:55.007</display>
    <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 12:37:16.44</display>
    <rawString xmlns=""/></ns1:value>
 </ns1:cell>
</ns1:row>
<ns1:row rowNumber="2">
 <ns1:cell>
  <ns1:value><display xmlns="">0A0B32E0CFC420760000010EC3B57CAF81C3</display>
    <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">/Sales Data Analysis</display>
    <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2:2006-11-10 13:11:04.257</display>
    <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0A0B32E029F5F2B00000010ED2672A3781A3</display>
    <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">4</display><rawString xmlns=""/>
  </ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2</display><rawString xmlns=""/>
  </ns1:value>
```

```
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 13:11:09.947</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 13:13:46.707</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 13:11:09.853</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
</ns1:row>
<ns1:row rowNumber="3">
 <ns1:cell>
  <ns1:value><display xmlns="">0A0B32E0CFC420760000010EC3B57CAF81C3</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">/Sales Data Analysis</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">3:2006-11-10 13:17:44.698</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">LATEST</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0A0B32E029F5F2B00000010ED2672A3781F5</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">4</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 13:17:51.463</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 13:21:19.277</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 13:17:51.37</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
```

```
      </ns1:row>
      <ns1:navigator>
        <page display="1" selector="1" current="true" xmlns=""/>
      </ns1:navigator>
    </ns1:pageResult>
  </queryExecutionsResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The queryJobTriggers operation

Returns all schedules in the system meeting specified criteria.

Input fields

The following table lists the input fields for the queryJobTriggers operation.

Table 4-38
Fields for queryJobTriggers

Field	Type/Valid Values	Description
jobTriggerCriterion	jobTriggerCriterion	Definition of the query parameters.

Return information

The following table identifies the information returned by the queryJobTriggers operation.

Table 4-39
Return Value

Type	Description
pageResult	Results of a search request to the search2.5 mechanism. This may either be an initial search based on any criterion, or a subsequent search for another page.

Java example

Querying the repository for job triggers typically involves the following steps:

1. Create a JobTriggerCriterion object.
2. Create a ReturnSpecifier object and define the return settings.
3. Create a PageSelector object and define the selector settings.
4. Assign the ReturnSpecifier and PageSelector objects to the JobTriggerCriterion object.
5. Supply the JobTriggerCriterion object to the queryJobTriggers operation.

The following sample retrieves all job triggers in the system for a specified file, sending the returned fields and their values for each schedule to the standard output.

```
JobTriggerCriterion jobTriggerCrit = new JobTriggerCriterion();

ReturnSpecifier retSpec = new ReturnSpecifier();
retSpec.setSortOrder(ReturnSpecifierSortOrder.descending);
BigInteger pSize = new BigInteger("11");
retSpec.setPageSize(pSize);
jobTriggerCrit.setReturnSpecifier(retSpec);

PageSelector pageSel = new PageSelector();
JobTriggerFilter filter = new JobTriggerFilter();
filter.setJobLocationURI("SPSSCR://pes_server/Monthly%20Status");
pageSel.setFilter(filter);
jobTriggerCrit.setPageSelector(pageSel);

PageResult qResult = stub.queryJobTriggers(jobTriggerCrit);

ColumnType[] resultColumn = qResult.getColumn();
Row[] resultRow = qrResult.getRow();
Cell[] resultCell = new Cell[resultColumn.length];
for (int i = 0; i < resultRow.length; i++) {
  System.out.println("Hit " + i);
  resultCell = resultRow[i].getCell();
  for (int j = 0; j < resultCell.length; j++) {
    System.out.println(resultColumn[j].getFieldName() + "=" +
      resultCell[j].getValue().getDisplay());
  }
  System.out.println("\n");
}
```

SOAP request example

Client invocation of the queryJobTriggers operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <queryJobTriggers xmlns="http://xml.spss.com/prms/remote">
   <ns2:jobTriggerCriterion xmlns:ns2="http://xml.spss.com/prms">
```

```
    <ns2:returnSpecifier sortOrder="descending" pageSize="11"/>
    <ns2:pageSelector>
     <ns2:filter xsi:type="jobTriggerFilter">
      <ns2:jobLocationURI>SPSSCR://pes_server/Monthly%20Status</ns2:jobLocationURI>
     </ns2:filter>
    </ns2:pageSelector>
   </ns2:jobTriggerCriterion>
  </queryJobTriggers>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a queryJobTriggers operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <queryJobTriggersResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:pageResult hitCount="2" pageSize="11" pageNumber="1"
    clientKey="19bc8c69:114d228ef1f:-7f9a" sortColumn="job.objid" sortOrder="ascending"
    xmlns:ns1="http://www.spss.com/pes/pager">
    <ns1:column display="job.objid" fieldName="job.objid" colType="string"/>
    <ns1:column display="jobPath" fieldName="jobPath" colType="string"/>
    <ns1:column display="fileversion.marker" fieldName="fileversion.marker" colType="string"/>
    <ns1:column display="usesfiledep.label" fieldName="usesfiledep.label" colType="string"/>
    <ns1:column display="scheduleObjID" fieldName="scheduleObjID" colType="string"/>
    <ns1:column display="schedule.frequency" fieldName="schedule.frequency" colType="number"/>
    <ns1:column display="schedule.interval" fieldName="schedule.interval" colType="number"/>
    <ns1:column display="schedule.month" fieldName="schedule.month" colType="number"/>
    <ns1:column display="schedule.dayOfMonth" fieldName="schedule.dayOfMonth" colType="number"/>
    <ns1:column display="schedule.daysOfWeek" fieldName="schedule.daysOfWeek" colType="number"/>
    <ns1:column display="schedule.timeOfDay" fieldName="schedule.timeOfDay" colType="stamp"/>
    <ns1:column display="schedule.scheduleStartDate" fieldName="schedule.scheduleStartDate"
     colType="stamp"/>
    <ns1:column display="schedule.scheduleEndDate" fieldName="schedule.scheduleEndDate"
     colType="stamp"/>
    <ns1:column display="schedule.nextScheduledTime" fieldName="schedule.nextScheduledTime"
     colType="stamp"/>
    <ns1:column display="schedule.scheduleEnabled" fieldName="schedule.scheduleEnabled"
     colType="string"/>
    <ns1:column display="scheduleEventExecution.state" fieldName="scheduleEventExecution.state"
     colType="number"/>
    <ns1:column display="scheduleEventExecution.executionSuccess"
     fieldName="scheduleEventExecution.executionSuccess" colType="string"/>
    <ns1:column display="scheduleEventExecution.startDateTime"
     fieldName="scheduleEventExecution.startDateTime" colType="stamp"/>
    <ns1:column display="scheduleEventExecution.endDateTime"
```

```
 fieldName="scheduleEventExecution.endDateTime" colType="stamp"/>
<ns1:column display="scheduleEventExecution.queuedDateTime"
 fieldName="scheduleEventExecution.queuedDateTime" colType="stamp"/>
<ns1:column display="credentials.name" fieldName="credentials.name" colType="string"/>
<ns1:column display="msgDrivenJob.objid" fieldName="msgDrivenJob.objid" colType="string"/>
<ns1:column display="msgDrivenJob.messageSelector "
 fieldName="msgDrivenJob.messageSelector " colType="string"/>
<ns1:column display="msgDrivenJob.messageText " fieldName="msgDrivenJob.messageText "
 colType="string"/>
<ns1:column display="msgEventExecution.state " fieldName="msgEventExecution.state "
 colType="number"/>
<ns1:column display="msgEventExecution.executionSuccess "
 fieldName="msgEventExecution.executionSuccess " colType="string"/>
<ns1:column display="msgEventExecution.startDateTime "
 fieldName="msgEventExecution.startDateTime " colType="stamp"/>
<ns1:column display="msgEventExecution.endDateTime "
 fieldName="msgEventExecution.endDateTime " colType="stamp"/>
<ns1:column display="msgEventExecution.queuedDateTime "
 fieldName="msgEventExecution.queuedDateTime " colType="stamp"/>
<ns1:column display="messageDomain.name " fieldName="messageDomain.name " colType="string"/>
<ns1:column display="messageDomain.destinationName "
 fieldName="messageDomain.destinationName " colType="string"/>
<ns1:column display="messageDomain.namingService " fieldName="messageDomain.namingService "
 colType="string"/>
<ns1:column display="messageDomain.namingFactory " fieldName="messageDomain.namingFactory "
 colType="string"/>
<ns1:row rowNumber="1">
 <ns1:cell>
  <ns1:value><display xmlns="">0A0A4A3519BC8C6900000114D228EF1F8023</display></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">/Monthly Status</display></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0:2007-09-04 15:36:33.548</display></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">LATEST</display></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
 </ns1:cell>
 <ns1:cell>
```

```
  <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
  </ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
  </ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
  </ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
  </ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
  </ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
  </ns1:value>
</ns1:cell>
<ns1:cell>
  <ns1:value>
```

```
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
   </ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">validUser</display></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">0A0A4A3519BC8C6900000114D228EF1F8039</display></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
   </ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
   </ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
   </ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">MyDomain</display></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">testTopic</display></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">jnp://localhost:1099</display></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">org.jnp.interfaces.NamingContextFactory</display></ns1:value>
  </ns1:cell>
```

```
</ns1:row>
<ns1:row rowNumber="2">
 <ns1:cell>
  <ns1:value><display xmlns="">0A0A4A3519BC8C6900000114D228EF1F8023</display></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">/Monthly Status</display></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0:2007-09-04 15:36:33.548</display></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">LATEST</display></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0A0A4A3519BC8C6900000114D228EF1F803D</display></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">3</display><rawNumber xmlns="">3</rawNumber></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">1</display><rawNumber xmlns="">1</rawNumber></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">1</display><rawNumber xmlns="">1</rawNumber></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value>
   <display xmlns="">1970-01-01T15:38:00-0600</display>
   <rawStamp xmlns="">1970-01-01T15:38:00.000-06:00</rawStamp>
  </ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value>
   <display xmlns="">2007-09-04T15:38:34-0500</display>
   <rawStamp xmlns="">2007-09-04T15:38:34.000-05:00</rawStamp>
  </ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value>
   <display xmlns=""/>
   <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
  </ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value>
```

```
    <display xmlns="">2007-10-01T15:38:00-0500</display>
    <rawStamp xmlns="">2007-10-01T15:38:00.000-05:00</rawStamp>
   </ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">true</display></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
   </ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
   </ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value>
    <display xmlns=""/>
    <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
   </ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">validUser</display></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns="">0</display><rawNumber xmlns="">0</rawNumber></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
  </ns1:cell>
  <ns1:cell>
   <ns1:value>
    <display xmlns=""/>
```

```
        <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
       </ns1:value>
      </ns1:cell>
      <ns1:cell>
       <ns1:value>
        <display xmlns=""/>
        <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
       </ns1:value>
      </ns1:cell>
      <ns1:cell>
       <ns1:value>
        <display xmlns=""/>
        <rawStamp xmlns="">1969-12-31T18:00:00.000-06:00</rawStamp>
       </ns1:value>
      </ns1:cell>
      <ns1:cell>
       <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
      </ns1:cell>
      <ns1:cell>
       <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
      </ns1:cell>
      <ns1:cell>
       <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
      </ns1:cell>
      <ns1:cell>
       <ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value>
      </ns1:cell>
     </ns1:row>
     <ns1:navigator>
      <page display="1" selector="1" current="true" xmlns=""/>
     </ns1:navigator>
    </ns1:pageResult>
   </queryJobTriggersResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The querySchedules operation

Returns all schedules in the system meeting specified criteria.

Input fields

The following table lists the input fields for the querySchedules operation.

Table 4-40
Fields for querySchedules

Field	Type/Valid Values	Description
scheduleCriterion	scheduleCriterion	Definition of the query parameters.

Return information

The following table identifies the information returned by the querySchedules operation.

Table 4-41
Return Value

Type	Description
pageResult	Results of a search request to the search2.5 mechanism. This may either be an initial search based on any criterion, or a subsequent search for another page.

Java example

Querying the repository for schedules typically involves the following steps:

1. Create a ScheduleCriterion object.
2. Create a ReturnSpecifier object and define the return settings.
3. Create a PageSelector object and define the selector settings.
4. Assign the ReturnSpecifier and PageSelector objects to the ScheduleCriterion object.
5. Supply the ScheduleCriterion object to the querySchedules operation.

The following sample retrieves all schedules in the system, sending the returned fields and their values for each schedule to the standard output.

```
ScheduleCriterion schedCrit = new ScheduleCriterion();
PageResult qResult = stub.querySchedules(schedCrit);
ColumnType[] resultColumn = qResult.getColumn();
Row[] resultRow = qrResult.getRow();
Cell[] resultCell = new Cell[resultColumn.length];
for (int i = 0; i < resultRow.length; i++) {
  System.out.println("Hit " + i);
  resultCell = resultRow[i].getCell();
  for (int j = 0; j < resultCell.length; j++) {
    System.out.println(resultColumn[j].getFieldName() + "=" +
      resultCell[j].getValue().getDisplay());
  }
  System.out.println("\n");
}
```

SOAP request example

Client invocation of the querySchedules operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <querySchedules xmlns="http://xml.spss.com/prms/remote">
   <scheduleCriterion xmlns="http://xml.spss.com/prms"/>
  </querySchedules>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a querySchedules operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <querySchedulesResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:pageResult hitCount="3" pageSize="255" pageNumber="1"
    clientKey="781b7393:10ed217fdaa:-7e80" sortColumn="file.objid" sortOrder="ascending"
    xmlns:ns1="http://www.spss.com/pes/pager">
    <ns1:column display="file.objid" fieldName="file.objid" colType="string"/>
    <ns1:column display="file.path" fieldName="file.path" colType="string"/>
    <ns1:column display="fileversion.marker" fieldName="fileversion.marker" colType="string"/>
    <ns1:column display="usesfiledep.label" fieldName="usesfiledep.label" colType="string"/>
    <ns1:column display="schedule.frequency" fieldName="schedule.frequency" colType="string"/>
    <ns1:column display="schedule.interval" fieldName="schedule.interval" colType="string"/>
    <ns1:column display="schedule.month" fieldName="schedule.month" colType="string"/>
    <ns1:column display="schedule.dayOfMonth" fieldName="schedule.dayOfMonth" colType="string"/>
    <ns1:column display="schedule.daysOfWeek" fieldName="schedule.daysOfWeek" colType="string"/>
    <ns1:column display="schedule.timeOfDay" fieldName="schedule.timeOfDay" colType="string"/>
    <ns1:column display="schedule.scheduleStartDate" fieldName="schedule.scheduleStartDate"
     colType="string"/>
    <ns1:column display="schedule.scheduleEndDate" fieldName="schedule.scheduleEndDate"
     colType="string"/>
    <ns1:column display="schedule.nextScheduleTime" fieldName="schedule.nextScheduleTime"
     colType="string"/>
    <ns1:column display="schedule.scheduleEnabled" fieldName="schedule.scheduleEnabled"
     colType="string"/>
    <ns1:column display="eventexe.state" fieldName="eventexe.state" colType="string"/>
```

```
<ns1:column display="eventexe.completionCode" fieldName="eventexe.completionCode"
 colType="string"/>
<ns1:column display="eventexe.startDateTime" fieldName="eventexe.startDateTime"
 colType="string"/>
<ns1:column display="eventexe.endDateTime" fieldName="eventexe.endDateTime" colType="string"/>
<ns1:column display="eventexe.queuedDateTime" fieldName="eventexe.queuedDateTime"
 colType="string"/>
<ns1:row rowNumber="1">
 <ns1:cell>
  <ns1:value><display xmlns="">0A0A4A35057D4A5E0000010ECCFED0BA81B6</display>
   <rawString xmlns=""/>
  </ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">/Jobs/Analysis</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0:2006-11-09 15:07:31.52</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">LATEST</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">1</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">4</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">1970-01-01 13:31:00.0</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 13:31:31.0</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-13 13:31:00.0</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
```

```
  <ns1:value><display xmlns="">1</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
</ns1:row>
<ns1:row rowNumber="2">
 <ns1:cell>
  <ns1:value><display xmlns="">0A0A4A35057D4A5E0000010ECCFED0BA81B6</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">/Jobs/Analysis</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0:2006-11-09 15:07:31.52</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">LATEST</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">3</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">1</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">1</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">1970-01-01 13:31:00.0</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-10 13:31:49.0</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-12-01 13:31:00.0</display>
   <rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
```

```
    <ns1:value><display xmlns="">1</display><rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
   <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
   <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
   <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
   <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
  </ns1:row>
  <ns1:row rowNumber="3">
   <ns1:cell>
    <ns1:value><display xmlns="">0A0A4A35781B73930000010ED217FDAA8177</display>
     <rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">/Jobs/Report</display><rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">0:2006-11-10 13:32:09.731</display>
     <rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">Production</display><rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">1</display><rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">1</display><rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">0</display><rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">0</display><rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">0</display><rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">1970-01-01 13:32:00.0</display>
     <rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">2006-11-10 13:32:57.0</display>
     <rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
   <ns1:cell>
    <ns1:value><display xmlns="">2006-11-11 13:32:00.0</display>
     <rawString xmlns=""/></ns1:value>
   </ns1:cell>
   <ns1:cell>
```

```
      <ns1:value><display xmlns="">1</display><rawString xmlns=""/></ns1:value>
     </ns1:cell>
     <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
     <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
     <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
     <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
     <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
    </ns1:row>
    <ns1:navigator>
     <page display="1" selector="1" current="true" xmlns=""/>
    </ns1:navigator>
   </ns1:pageResult>
  </querySchedulesResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The querySubmittedExecutions operation

Returns all submitted executions in the system meeting specified criteria.

Input fields

The following table lists the input fields for the querySubmittedExecutions operation.

Table 4-42
Fields for querySubmittedExecutions

Field	Type/Valid Values	Description
submittedExecutionCriterion	submittedExecutionCriterion	Definition of the query parameters.

Return information

The following table identifies the information returned by the querySubmittedExecutions operation.

Table 4-43
Return Value

Type	Description
pageResult	Results of a search request to the search2.5 mechanism. This may either be an initial search based on any criterion, or a subsequent search for another page.

Java example

Querying the repository for executions typically involves the following steps:

1. Create a SubmittedExecutionCriterion object.
2. Create a ReturnSpecifier object and define the return settings.

3. Create a PageSelector object and define the selector settings.
4. Assign the ReturnSpecifier and PageSelector objects to the SubmittedExecutionCriterion object.
5. Supply the SubmittedExecutionCriterion object to the querySubmittedExecutions operation.

The following sample retrieves all submitted executions with the label *Production*, sending the returned fields and their values for each execution to the standard output. Results are sorted by the completion code for the execution.

```
SubmittedExecutionCriterion execCrit = new SubmittedExecutionCriterion();
ReturnSpecifier retSpec = new ReturnSpecifier();
retSpec.setSortColumn("eventexe.completionCode");
execCrit.setReturnSpecifier(retSpec);

PageSelector pageSel = new PageSelector();
SubmittedExecutionFilter filter = new SubmittedExecutionFilter();
filter.setVersionLabel("Production");
pageSel.setFilter(filter);
execCrit.setPageSelector(pageSel);

PageResult qResult = stub.querySubmittedExecutions(execCrit);
ColumnType[] resultColumn = qResult.getColumn();
Row[] resultRow = qrResult.getRow();
Cell[] resultCell = new Cell[resultColumn.length];
for (int i = 0; i < resultRow.length; i++) {
  System.out.println("Hit " + i);
  resultCell = resultRow[i].getCell();
  for (int j = 0; j < resultCell.length; j++) {
    System.out.println(resultColumn[j].getFieldName() + "=" +
      resultCell[j].getValue().getDisplay());
  }
  System.out.println("\n");
}
```

SOAP request example

Client invocation of the querySubmittedExecutions operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
```

```
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <querySubmittedExecutions xmlns="http://xml.spss.com/prms/remote">
   <submittedExecutionCriterion xmlns="http://xml.spss.com/prms">
    <returnSpecifier sortColumn="eventexe.completionCode"/>
    <pageSelector>
     <submittedExecutionFilter>
      <versionLabel>Production</versionLabel>
     </submittedExecutionFilter>
    </pageSelector>
   </submittedExecutionCriterion>
  </querySubmittedExecutions>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a querySubmittedExecutions operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <querySubmittedExecutionsResponse xmlns="http://xml.spss.com/prms/remote">
   <ns1:pageResult hitCount="9" pageSize="2" pageNumber="1"
    clientKey="50f84fe9:10f33316d48:-7671" sortColumn="eventexe.completionCode"
    sortOrder="ascending" xmlns:ns1="http://www.spss.com/pes/pager">
    <ns1:column display="file.objid" fieldName="file.objid" colType="string"/>
    <ns1:column display="file.name" fieldName="file.name" colType="string"/>
    <ns1:column display="fileversion.marker" fieldName="fileversion.marker" colType="string"/>
    <ns1:column display="fileversion.expirationDate" fieldName="fileversion.expirationDate"
     colType="string"/>
    <ns1:column display="eventexe.objId" fieldName="eventexe.objId" colType="string"/>
    <ns1:column display="eventexe.state" fieldName="eventexe.state" colType="string"/>
    <ns1:column display="eventexe.startDateTime" fieldName="eventexe.startDateTime"
     colType="string"/>
    <ns1:column display="eventexe.endDateTime" fieldName="eventexe.endDateTime" colType="string"/>
    <ns1:column display="eventexe.queuedDateTime" fieldName="eventexe.queuedDateTime"
     colType="string"/>
    <ns1:column display="eventexe.completionCode" fieldName="eventexe.completionCode"
     colType="string"/>
    <ns1:row rowNumber="1">
     <ns1:cell>
      <ns1:value>
       <display xmlns="">0A58C34649AB87F00000010EFAFA4C3D89CC</display><rawString xmlns=""/>
      </ns1:value>
     </ns1:cell>
     <ns1:cell>
```

```
  <ns1:value><display xmlns="">Employee.dbq.html</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0:2006-11-20 10:35:49.342</display><rawString xmlns=""/>
  </ns1:value>
 </ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell>
  <ns1:value>
   <display xmlns="">0A58C34649AB87F00000010EFAFA4C3D89C5</display><rawString xmlns=""/>
  </ns1:value>
 </ns1:cell>
 <ns1:cell><ns1:value><display xmlns="">4</display><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-20 10:35:46.967</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-20 10:35:50.153</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-20 10:35:46.953</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell><ns1:value><display xmlns="">0</display><rawString xmlns=""/></ns1:value></ns1:cell>
</ns1:row>
<ns1:row rowNumber="2">
 <ns1:cell>
  <ns1:value>
   <display xmlns="">0A58C34649AB87F00000010EFAFA4C3D89DD</display><rawString xmlns=""/>
  </ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">Employee.dbq.html</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">0:2006-11-20 10:36:38.699</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell><ns1:value><display xmlns=""/><rawString xmlns=""/></ns1:value></ns1:cell>
 <ns1:cell>
  <ns1:value>
   <display xmlns="">0A58C34649AB87F00000010EFAFA4C3D89D5</display><rawString xmlns=""/>
  </ns1:value>
 </ns1:cell>
 <ns1:cell><ns1:value><display xmlns="">4</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-20 10:36:35.95</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-20 10:36:39.76</display><rawString xmlns=""/></ns1:value>
 </ns1:cell>
 <ns1:cell>
  <ns1:value><display xmlns="">2006-11-20 10:36:35.933</display><rawString xmlns=""/></ns1:value>
```

```
      </ns1:cell>
      <ns1:cell><ns1:value><display xmlns="">0</display><rawString xmlns=""/></ns1:value></ns1:cell>
     </ns1:row>
     <ns1:navigator>
      <next display="pager/navNext" selector="3" current="false" xmlns=""/>
      <page display="1" selector="1" current="true" xmlns=""/>
      <page display="2" selector="3" current="false" xmlns=""/>
      <page display="3" selector="5" current="false" xmlns=""/>
      <page display="4" selector="7" current="false" xmlns=""/>
      <page display="5" selector="9" current="false" xmlns=""/>
     </ns1:navigator>
    </ns1:pageResult>
   </querySubmittedExecutionsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The rerunFailedJobStepIterations operation

Submits only the failed job step iterations for execution.

Input fields

The following table lists the input fields for the rerunFailedJobStepIterations operation.

Table 4-44
Fields for rerunFailedJobStepIterations

Field	Type/Valid Values	Description
oldJobStepExecutionID	string	Identifier for a job step execution.

Return information

The following table identifies the information returned by the rerunFailedJobStepIterations operation.

Table 4-45
Return Value

Type	Description
string	Identifier for the job step execution resulting from re-running the failed steps.

The submitJob operation

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.

Input fields

The following table lists the input fields for the submitJob operation.

Table 4-46
Fields for submitJob

Field	Type/Valid Values	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.

Return information

The following table identifies the information returned by the submitJob operation.

Table 4-47
Return Value

Type	Description
string	Identifier for the job execution.

Java example

Submitting a job for execution requires the specification of two parameters. The first is a string corresponding to the URI for the job. The second is a boolean indicating whether or not any notifications defined for the job should be sent as a result of the submission. The submitJob operation returns the identifier for the execution of the specified job, from which specific result details can be obtained. For more information, see the topic The getExecutionDetails operation on p. 34.

```
String uri=
  "spsscr://pes_server:80/?id=0a0a4aac011937790000010d415272aa824a#m.1:2006-08-24%2013:16:21.069";
String executionID = stub.submitJob(uri, true);
ExecutionDetails exDetails = stub.getExecutionDetails(executionID,DeliveryType.STRING);
```

SOAP request example

Client invocation of the submitJob operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
```

```
 </soapenv:Header>
 <soapenv:Body>
  <submitJob xmlns="http://xml.spss.com/prms/remote">
   <jobLocationURI>
    spsscr://pes_server:80/?id=0a0a4aac011937790000010d415272aa824a#m.1:2006-08-24%2013:16:21.069
   </jobLocationURI>
   <notificationEnabled>true</notificationEnabled>
  </submitJob>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a submitJob operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <submitJobResponse xmlns="http://xml.spss.com/prms/remote">
   <jobExecutionID>0a0a4aac011937790000010d415272aa8ab4</jobExecutionID>
  </submitJobResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The submitJobWithOptions operation

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

Input fields

The following table lists the input fields for the submitJobWithOptions operation.

Table 4-48
Fields for submitJobWithOptions

Field	Type/Valid Values	Description
jobLocationURI	string	URI for a job.
jobOptions	jobOptions	Options for running a job.

Return information

The following table identifies the information returned by the submitJobWithOptions operation.

Table 4-49
Return Value

Type	Description
string	Identifier for the job execution.

Java example

To specify options when submitting a job for execution:

1. Create a JobOptions object.
2. Supply the setNotificationEnabled method with a boolean indicating whether or not notifications defined for the job should be submitted as a result of submitting the job.
3. Create an array of JobParameterValue objects. Use the setName and setValue methods to assign the name and value for each parameter value in the job.
4. Create a JobParameterValues object. Use the setParameterValue method to assign the parameter value array to the object.
5. Assign the parameter values to the options object using the setParameterValues method.
6. Supply the submitJobWithOptions operation with a string denoting the uniform resource identifier for the job and the options object. The operation returns the identifier for the execution of the specified job, from which specific result details can be obtained. For more information, see the topic The getExecutionDetails operation on p. 34.

The following sample specifies the values for two job variables when submitting a job.

```
String uri = "spsscr:///Jobs/My%20Job#m.4:2009-03-17%2008:53:42.482";

JobOptions options = new JobOptions();
options.setNotificationEnabled(true);

JobParameterValue[] valArray = new JobParameterValue[2];
valArray[0].setName("command");
valArray[0].setValue("dir");
valArray[1].setName("options");
valArray[1].setValue("c:");
JobParameterValues pValues = new JobParameterValues();
pValues.setParameterValue(valArray);
options.setParameterValues(pValues);

String executionID = stub.submitJobWithOptions(uri, options);

ExecutionDetails exDetails = stub.getExecutionDetails(executionID,DeliveryType.STRING);
```

SOAP request example

Client invocation of the submitJobWithOptions operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
   xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
  <soapenv:Body>
    <submitJobWithOptions xmlns="http://xml.spss.com/prms/remote">
      <jobLocationURI>SPSSCR:///Jobs/My%20Job#m.4:2009-03-17%2008:53:42.482</jobLocationURI>
      <jobOptions xmlns="http://xml.spss.com/prms">
        <parameterValues>
          <parameterValue name="command" value="dir"/>
          <parameterValue name="options" value="c:"/>
        </parameterValues>
        <notificationEnabled>true</notificationEnabled>
      </jobOptions>
    </submitJobWithOptions>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a submitJobWithOptions operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <submitJobWithOptionsResponse xmlns="http://xml.spss.com/prms/remote">
      <jobExecutionID>0a010a0706890875000001201009f4f583fd</jobExecutionID>
    </submitJobWithOptionsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The submitWork operation

Submits a designated piece of work for processing, generating an execution from which the results can be obtained.

Input fields

The following table lists the input fields for the submitWork operation.

Table 4-50
Fields for submitWork

Field	Type/Valid Values	Description
workSpec	workSpec	Defines a specification for submitting work.

Return information

The following table identifies the information returned by the submitWork operation.

Table 4-51
Return Value

Type	Description
string	Identifier for the job step execution.

Java example

Submitting work for execution requires the creation of a work specifiction. The specification defines the following two parameters:

- The identifier for the work to submit
- A name for the work, used to organize the work in the submitted hierarchy

The following sample begins by constructing a Work object for an SPSS syntax file. The setUuid method defines the identifier for the syntax file. The WorkSpec constructor accepts this work object and a string indicating the work name, in this case *Results*. The submitWork operation returns the identifier for the execution of the work, from which specific result details can be obtained. For more information, see the topic The getExecutionDetails operation on p. 34.

```
Work work = new Work("ModelManagement.SPSSSyntaxWork");
Id id = new Id("0a0a4aac011937790000010d415272aa81e8");
work.setUuid(id);
WorkSpec wSpec = new WorkSpec(work, "Results");
String executionID = stub.submitWork(wSpec);
```

SOAP request example

Client invocation of the submitWork operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
```

```
      <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
      <wsse:Password xsi:type="xsd:string">password</wsse:Password>
    </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
    xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <submitWork xmlns="http://xml.spss.com/prms/remote">
   <workSpec xmlns="http://xml.spss.com/prms">
    <work uuid="0a0a4aac011937790000010d415272aa81e8" workType="ModelManagement.SPSSSyntaxWork"/>
    <workName>Results</workName>
   </workSpec>
  </submitWork>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a submitWork operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <submitWorkResponse xmlns="http://xml.spss.com/prms/remote">
   <jobStepExecutionID>0a0a4aac011937790000010d415272aa8d0a</jobStepExecutionID>
  </submitWorkResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The updateMessageDrivenJob operation

Updates an existing message-driven job with new characteristics.

Input fields

The following table lists the input fields for the updateMessageDrivenJob operation.

Table 4-52
Fields for updateMessageDrivenJob

Field	Type/Valid Values	Description
messageDrivenJobIn	messageDrivenJobSpecifier	Message Drive Job parameters.

Java example

Updating a message driven job typically involves the following steps:

1. Obtain a MessageDrivenJobSpecifier object for the job to be updated by supplying the getMessageDrivenJob operation with the message driven job identifier.
2. Retrieve the MessageDrivenJob object.
3. Update the parameters as desired.
4. Update the MessageDrivenJobSpecifier object with the new settings.
5. Supply the updateMessageDrivenJob operation with the revised specifier object.

The following sample changes the filter text used for an existing message driven job.

```
String id = "0A0A4A3578F9BA1100000114D5CED52780E6";
MessageDrivenJobSpecifier mdjSpec = stub.getMessageDrivenJob(id);
MessageDrivenJob mdJob = mdjSpec.getMessageDrivenJob();
mdJob.setMessageText("ETL complete");
mdjSpec.setMessageDrivenJob(mdJob);
stub.updateMessageDrivenJob(mdjSpec);
```

SOAP request example

Client invocation of the updateMessageDrivenJob operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
  xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <updateMessageDrivenJob xmlns="http://xml.spss.com/prms/remote">
   <messageDrivenJobIn xmlns="http://xml.spss.com/prms">
    <messageDrivenJob uuid="0a0a4a3578f9ba1100000114d5ced52780e6" enabled="true" label="LATEST"
     credentialID="0a0a4a3519bc8c6900000114d228ef1f8033"
     messageDomainID="0a0a4a3578f9ba1100000114d5ced52780c8" messageText="ETL complete"
     durableSubscription="false"/>
   </messageDrivenJobIn>
  </updateMessageDrivenJob>
```

```
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a updateMessageDrivenJob operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <updateMessageDrivenJobResponse xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

The updateSchedule operation

Updates an existing schedule with new characteristics.

Input fields

The following table lists the input fields for the updateSchedule operation.

Table 4-53
Fields for updateSchedule

Field	Type/Valid Values	Description
scheduleIn	scheduleSpecifier	Schedule parameters.

Java example

Updating a schedule typically involves the following steps:

1. Obtain a ScheduleSpecifier object for the schedule to be updated by supplying the getSchedule operation with the schedule identifier.
2. Retrieve the Schedule object.
3. Update the schedule parameters as desired.
4. Update the ScheduleSpecifier object with the new schedule settings.
5. Supply the updateSchedule operation with the revised ScheduleSpecifier object.

The following sample changes the day a weekly schedule runs from Sunday to Monday.

```
String id = "0a0a4a35781b73930000010ed217fdaa81c1";
ScheduleSpecifier schedSpec = stub.getSchedule(id);
Schedule schedule = schedSpec.getSchedule();
```

```
schedule.setMonday(true);
schedule.setSunday(false);
schedSpec.setSchedule(schedule);
stub.updateSchedule(schedSpec);
```

SOAP request example

Client invocation of the updateSchedule operation generates a SOAP request message that is sent to the server for processing. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="0"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken>
    <wsse:Username xsi:type="xsd:string">validUser</wsse:Username>
    <wsse:Password xsi:type="xsd:string">password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
  <ns1:client-accept-language soapenv:mustUnderstand="0" xsi:type="xsd:string"
  xmlns:ns1="http://xml.spss.com/ws/headers">en-US;q=1.0, en;q=0.8</ns1:client-accept-language>
 </soapenv:Header>
 <soapenv:Body>
  <updateSchedule xmlns="http://xml.spss.com/prms/remote">
   <scheduleIn xmlns="http://xml.spss.com/prms">
    <schedule uuid="0a0a4a35781b73930000010ed217fdaa81c1"
     scheduleEnabled="true" scheduledLabel="Production" interval="1"
     startDateTime="2006-11-10T13:31:31.000-06:00" timeOfDay="13:31:00"
     monday="true" tuesday="false" wednesday="false" thursday="false"
     friday="false" saturday="false" sunday="false" xsi:type="weeklySchedule"/>
   </scheduleIn>
  </updateSchedule>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a updateSchedule operation call by sending a SOAP response message containing the results. An example of such a message follows.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <updateScheduleResponse xmlns="http://xml.spss.com/prms/remote"/>
 </soapenv:Body>
</soapenv:Envelope>
```

Appendix

A

Microsoft® .NET Framework-based clients

In order to use the web services from a Microsoft Windows Communication Foundation (WCF) client, you will need Visual Studio 2008 or later. The discussion here assumes the use of Visual Studio 2008. In general, the process for accessing IBM® SPSS® Collaboration and Deployment Services web services involves the following steps:

1. Add a Service Reference. For more information, see the topic Adding a service reference on p. 92.
2. Configure the web service endpoint. For more information, see the topic Configuring the web service endpoint on p. 94.
3. Programmatically configure the necessary endpoint behaviors. For more information, see the topic Configuring endpoint behaviors on p. 95.
4. Exercise the web service. For more information, see the topic Exercising the service on p. 95.

Note that the IBM SPSS Collaboration and Deployment Services single sign-on implementation is not compatible with Microsoft .NET web services, or the WCF. As a result, single sign-on is not available from these clients.

Adding a service reference

The first step in using a WCF client to access IBM® SPSS® Collaboration and Deployment Services web services is to make the service available to the Visual Studio project by adding it as a Service Reference.

1. In Visual Studio, right-click the folder's *References* folder and select Add Service Reference.
2. Type the URL of the service WSDL location in the Address field, and click Go. The value corresponds to the service endpoint appended with *?wsdl*.
3. Specify the desired namespace in the Namespace field.
4. Click OK.

Visual Studio adds a new service reference to the Service Reference directory for the project. The name of the reference corresponds to the specified namespace.

Service reference modifications

Due to known compatibility issues between Microsoft tooling and some WSDL files, you need to manually modify some service references before they can be used successfully. For information about the specific issues, see articles 891386 (*http://support.microsoft.com/kb/891386*) and 326790 (*http://support.microsoft.com/kb/326790*) on the Microsoft Support site.

To modify a service reference:

1. In Visual Studio, select the project and click Show All Files from the Project menu.
2. Expand the service reference that needs to be modified.
3. Expand the *Reference.svcmap* node.
4. Open the *Reference.cs* file.
5. Make the required modifications.
6. Save the file.

For the Content Repository Service , Content Repository URI Service, and Process Management Service, you need to make the following changes to the RowType class:

- private value[][] cellField should be changed to private value[] cellField
- public value[][] cell should be changed to public value[] cell

For the Scoring Service, you need to make the following changes:

- in the returnedDPDOutputTable class, private returnedDPDOutputValue[][] returnedDPDOutputrowField should be changed to private returnedDPDOutputValue[] returnedDPDOutputrowField
- in the returnedDPDOutputTable class, private returnedDPDOutputValue[][] returnedDPDOutputRow should be changed to private returnedDPDOutputValue[] returnedDPDOutputRow
- in the returnedRequestInputTable class, private returnedRequestInputValue[][] returnedRequestInputRow should be changed to private returnedRequestInputValue[] returnedRequestInputRow
- in the returnedRequestInputTable class, private returnedRequestInputValue[][] returnedRequestInputRowField should be changed to private returnedRequestInputValue[] returnedRequestInputRowField
- in the requestInputTable class, private input1[][] requestInputRowField should be changed to private input1[] requestInputRowField
- in the requestInputTable class, private input1[][] requestInputRow should be changed to private input1[] requestInputRow

For the PevServices Service, you need to make the following changes:

- in the avTableConflict class, private avColumnMeta[][] avColumnConflictField should be changed to private avColumnMeta[] avColumnConflictField

- in the avTableConflict class, private avColumnMeta[][] avColumnConflict should be changed to private avColumnMeta[] avColumnConflict
- in the evTableConflict class, private evColumnMeta[][] evColumnConflictField should be changed to private evColumnMeta[] evColumnConflictField
- in the evTableConflict class, private evColumnMeta[][] evColumnConflict should be changed to private evColumnMeta[] evColumnConflict

Configuring the web service endpoint

In WCF, you can configure a service endpoint either declaratively using an *app.config* file, or programmatically using the WCF APIs. The following steps describe the creation of a basic configuration within an *app.config* file.

1. In Visual Studio, double-click the *app.config* file for the application (or *web.config* for a web-application).
2. Find the system.serviceModel element. Create it if it does not already exist.
3. Find the client element. Create it if it does not already exist.
4. Create a new endpoint element as a child of the client element.
5. Specify the appropriate service endpoint URL as the value of the *address* attribute.
6. Specify *basicHttpBinding* as the value of the *binding* attribute.
7. Specify the appropriate service contract as the value of the *contract* attribute. The service contract is the value of the service reference namespace appended with the service name.
8. Optionally specify a value for the *name* attribute that identifies a name for the endpoint configuration. If the *name* is blank, the configuration is used as the default for the service.

The resulting *app.config* file should be similar to the following example:

```
<system.serviceModel>
  <client>
    <endpoint
      address="http://cads_server:8080/cr-ws/services/ContentRepository"
      binding="basicHttpBinding"
      bindingConfiguration=""
      contract="IBM.SPSS.ContentRepository"
      name=""/>
  </client>
</system.serviceModel>
```

Configuring endpoint behaviors

The following two issues complicate the use of IBM® SPSS® Collaboration and Deployment Servicesweb services by WCF clients:

- WCF does not allow the username and password to be transmitted over HTTP
- WCF does not correctly understand the SOAP Fault format returned by the services

To address these problems, a sample Visual Studio project is available that contains classes adding endpoint behaviors that resolve both issues. The IBM SPSS Collaboration and Deployment Services installation media includes this project.

To use these classes, ensure that the *IBM.SPSS.WCF.Utilities* project containing these classes has been compiled and added as a reference to the Visual Studio project that exercises the web services. When constructing a new service client instance, ensure that the behaviors are added as follows:

```
ContentRepositoryClient serviceClient = new ContentRepositoryClient();
serviceClient.Endpoint.Behaviors.Add(
  new ApplyClientInspectorsBehavior(
  new HeaderInjectionMessageInspector(
    new UsernameTokenSecurityHeader("admin", "Abcdefg1")
  ),
  new SOAPFaultFormatMessageInspector())
);
```

This adds two message inspectors to the behaviors for the endpoint. The first allows message headers to be injected, permitting a UsernameToken security header containing the username and password to be transmitted over HTTP. The second message inspector intercepts SOAP Faults, ensuring that they are formatted for proper WCF processing.

Exercising the service

After adding the service reference to the project, configuring the endpoint, and adding the necessary endpoint behaviors, the WCF-based web service client is ready. Add the .NET source code to the project to exercise the web service as needed.

There may be instances in which the .NET client proxies are generated incorrectly, leading to unexpected missing results at runtime. If a web service call returns no results when results are expected, the generated .NET types associated with the request and response should be examined. Specifically, members of the types may have two .NET attributes assigned. The first, MessageBodyMemberAttribute, will often include the proper namespace for the member type. The second, XmlElementAttribute, should have the same namespace as MessageBodyMemberAttribute. If this is not the case, add the namespace to XmlElementAttribute. Moreover, the addition of XML serialization attributes, such as System.XML.Serialization.XmlElementAttribute, may be necessary to correctly name the expected namespace or element. For example, the following generated client code would need to be modified:

```
public partial class getUsersResponse {
  System.ServiceModel.MessageBodyMemberAttribute(Namespace =
    "http://xml.spss.com/pes/userPref/remote", Order = 0)]
```

```
  public IBM.SPSS.ManagerUserPref.usersResponse usersResponse;
```

The corrected code is as follows:

```
public partial class getUsersResponse {
  [System.ServiceModel.MessageBodyMemberAttribute(Namespace =
    "http://xml.spss.com/pes/userPref/remote", Order = 0)]
  [System.Xml.Serialization.XmlElementAttribute(ElementName="usersRequestResponse")]
  public IBM.SPSS.ManagerUserPref.usersResponse usersResponse;
```

Appendix

B

Notices

This information was developed for products and services offered worldwide.

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, Armonk, NY 10504-1785, U.S.A.

For license inquiries regarding double-byte character set (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., 1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242-8502 Japan.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES 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 states 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply 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 Software Group, Attention: Licensing, 233 S. Wacker Dr., Chicago, IL 60606, USA.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All 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 the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, and SPSS are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at *http://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 of Sun Microsystems, Inc. in the United States, other countries, or both.

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


Java

Index

app.config files
  WCF clients, 94

bindings
  in WSDL files, 5
body elements
  in SOAP messages, 3

cancelExecution operation, 20
child columns
  in page results, 19
columns
  in page results, 19
Content Repository service
  WCF clients, 93
Content Repository URI service
  WCF clients, 93
createMessageDrivenJob operation, 21
createSchedule operation, 23

daily schedules, 11
deleteJobExecutions operation, 26
deleteJobTrigger operation, 27
deleteJobTriggers operation, 29
deleteSchedule operation, 30
deleteSchedules operation, 31
domains, 12
durable subscriptions, 13

event executions, 15
event types
  retrieving, 33
events
  event executions, 15
executions, 15
  canceling, 20
  deleting, 26
  querying, 58, 77
  retrieving, 34, 38, 41

field names
  in queries, 15
filters
  in queries, 18
finalizeRemoteWork operation, 33

getCustomEventTypes operation, 33
getDefault method
  for JobParameter objects, 37
getDescription method
  for JobParameter objects, 37
getExecutionDetails operation, 34
getJobParameters operation, 37
getJobStepChildExecutions operation, 38
getJobStepExecutions operation, 41
getMessageDrivenJob operation, 43
getMessageDrivenJobs operation, 45
getName method
  for JobParameter objects, 37
getSchedule operation, 46
getSchedules operation, 48
getVersion operation, 50
getWorkTypes operation, 51

handleMessageDomainChanged operation, 53
header elements
  in SOAP messages, 3
Holder classes
  in JAX-WS, 6
hourly schedules, 11
HTTP, 2
HTTPS, 2

Java proxies, 6
JAX-WS, 6
JMS, 11
  message structure, 13
JobOptions objects, 84
JobParameter objects, 37
JobParameterValue objects, 84
JobParameterValues objects, 84
jobs, 9
  submitting, 81, 83
  variables, 10, 37, 83

legal notices, 97
List collections
  in JAX-WS, 6

message domains, 12, 53
message selectors, 15
MessageBodyMemberAttribute
  for WCF clients, 95
messages
  in WSDL files, 5
monthly schedules, 11

navigators
  in page results, 19
.NET framework, 92
.NET proxies, 7

page requests
  in queries, 18
page selectors
  in queries, 18
PevServices service
  WCF clients, 93

port types
  in WSDL files, 5
Process Management service, 8
  service endpoint, 8
  stubs, 8
  WCF clients, 93
protocols
  in web services, 2
proxies, 6
  Java, 6
  .NET, 7

queries, 15
  filters, 18
  page requests, 18
  page selectors, 18
  return specifiers, 18
queryAllSchedules operation, 54
queryExecutions operation, 58
queryJobTriggers operation, 63
querySchedules operation, 71
querySubmittedExecutions operation, 77

rerunFailedJobStepIterations operation, 81
return specifiers, 18
rows
  in page results, 19

schedules, 10
  creating, 21, 23
  deleting, 27, 29–31
  message-based, 11, 21, 27, 29, 43, 45, 53, 87
  querying, 54, 63, 71
  retrieving, 43, 45–46, 48
  time-based, 11
  updating, 87, 89
Scoring service
  WCF clients, 93
selectors
  for JMS messages, 15
service endpoints
  Process Management service, 8
services
  in WSDL files, 6
setName method
  for JobParameterValue objects, 84
setNotificationEnabled method
  for JobOptions objects, 84
setParameterValue method
  for JobParameterValues objects, 84
setParameterValues method
  for JobOptions objects, 84
setValue method
  for JobParameterValue objects, 84
single sign-on
  WCF clients, 92
SOAP, 2–3
sorting
  in queries, 18
stubs
  Process Management service, 8
submitJob operation, 81
submitJobWithOptions operation, 83
submitting
  jobs, 81, 83
  work, 85
submitWork operation, 85
subscriptions
  durable, 13

trademarks, 98
types
  in WSDL files, 4

uniform resource identifiers, 9
updateMessageDrivenJob operation, 87
updateSchedule operation, 89
URIs
  syntax, 9

versions
  in URIs, 10
Visual Studio, 92

WCF clients, 92, 95
  endpoint behaviors, 95
  endpoint configuration, 94
  limitations, 92
  service reference, 92–93
  single sign-on, 92
web services
  introduction to web services, 1
  protocol stack, 2
  system architecture, 1
  what are web services?, 1
web.config files
  WCF clients, 94
weekly schedules, 11
Windows Communication Foundation, 92
work
  submitting, 85
  types, 51
WSDL files, 2–3
  accessing, 8
  bindings, 5
  messages, 5
  port types, 5
  services, 6
  types, 4
wsdl.exe, 7
wsdl2java, 6

wsimport, 6

XmlElementAttribute
  for WCF clients, 95

yearly schedules, 11