

IBM SPSS Collaboration and Deployment
Services
8.5

*Content Repository URI Service
Developer's Guide*



Note

Before using this information and the product it supports, read the information in [“Notices” on page 93.](#)

Product Information

This edition applies to version 8, release 4, modification 0 of IBM® SPSS® Collaboration and Deployment Services and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2000, 2023.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 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.....	2
Web Service Description Language.....	3
Proxies.....	5
Chapter 2. Content Repository URI Service overview.....	7
Accessing the Content Repository URI Service	7
Calling Content Repository URI Service operations.....	7
Chapter 3. Repository Concepts.....	9
Resources.....	9
Folders.....	9
Files.....	9
Topics.....	10
Uniform Resource Identifiers.....	11
Specifiers.....	11
Custom properties.....	12
Defining custom properties.....	12
Assigning custom property values.....	12
Transfers.....	13
Chapter 4. Operation reference.....	15
The applyTransferPolicy operation.....	15
The cancelTransfer operation.....	16
The cascadePermissions operation.....	17
The copyResource operation.....	19
The createCustomProperty operation.....	20
The createOrAddResource operation.....	22
The createResource operation.....	23
The createResourcePropagate operation.....	25
The createUniqueSubmittedFolder operation.....	27
The deleteCustomProperty operation.....	27
The deleteResource operation.....	28
The disposeTransfer operation.....	29
The findAllLabels operation.....	30
The getActions operation.....	31
The getAllCustomProperties operation.....	34
The getAllLocks operation.....	35
The getAllVersions operation.....	38
The getBinaryContent operation.....	40
The getBulkResourceMetadata operation.....	40
The getChildren operation.....	42
The getChildrenOptions operation.....	44
The getFault operation.....	46
The getFile operation.....	47
The getResource operation.....	49
The getResourceSnapshot operation.....	51

The getResourceWithLock operation.....	52
The getTransferResults operation.....	53
The getTransferStatus operation.....	56
The getVersion operation.....	58
The getVersionLabels operation.....	59
The lockResource operation.....	60
The lockResources operation.....	61
The moveResource operation.....	62
The query operation.....	63
The removeLabel operation.....	64
The runCqlQuery operation.....	65
The setBulkResourceMetadata operation.....	66
The transferResource operation.....	67
The unlockResource operation.....	70
The unlockResources operation.....	71
The updateCustomProperty operation.....	72
The updateCustomPropertyValuesInBulk operation.....	73
The updateResource operation.....	75
The updateResources operation.....	77
Chapter 5. JAX-WS clients.....	79
Generating a JAX-WS client.....	79
Packaging a JAX-WS client.....	79
Configuring a JAX-WS client.....	79
SOAPHandler example.....	80
Exercising web services from JAX-WS clients.....	82
Chapter 6. Microsoft® .NET Framework-based clients.....	83
Adding a service reference.....	83
Service reference modifications.....	83
Configuring the web service endpoint.....	84
Configuring endpoint behaviors.....	85
Exercising the service.....	85
Single sign-on authentication.....	86
Chapter 7. Message header reference.....	87
Security headers.....	87
Security element.....	87
UsernameToken element.....	88
BinarySecurityToken and BinarySecuritySSToken elements.....	88
The client-accept-language element.....	89
HTTP headers.....	89
Appendix A. Deprecated features.....	91
Notices.....	93
Privacy policy considerations	94
Trademarks.....	94
Glossary.....	97
A.....	97
B.....	97
C.....	98
D.....	98
E.....	99
F.....	99
G.....	99

I.....	99
J.....	100
K.....	100
L.....	101
M.....	101
N.....	101
O.....	101
P.....	102
R.....	102
S.....	102
T.....	103
U.....	103
V.....	103
W.....	104
X.....	104
Index.....	105

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 WebSphere® or JBoss Application 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.

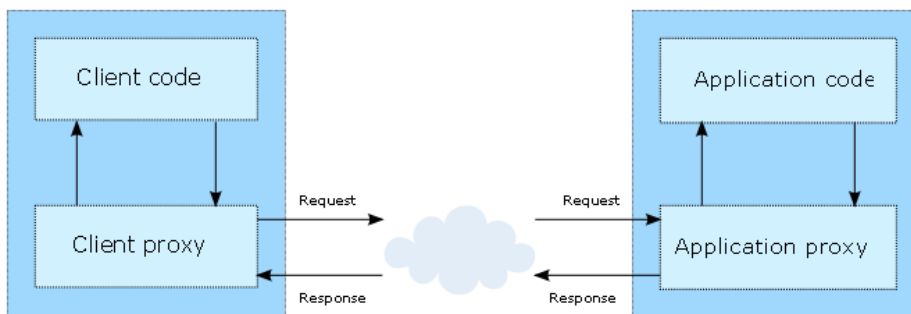


Figure 1. Web service architecture

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. See the topic [“Proxies” on page 5](#) for more information.

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.

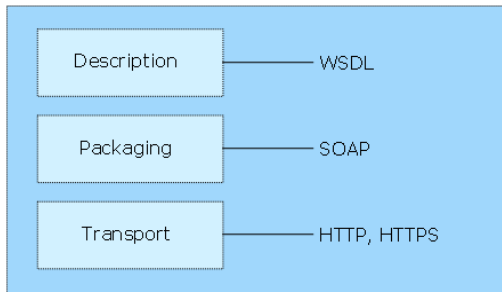


Figure 2. Web service protocol stack

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. See the topic [“Simple Object Access Protocol”](#) on page 2 for more information.

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. See the topic [“Web Service Description Language”](#) on page 3 for more information.

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:

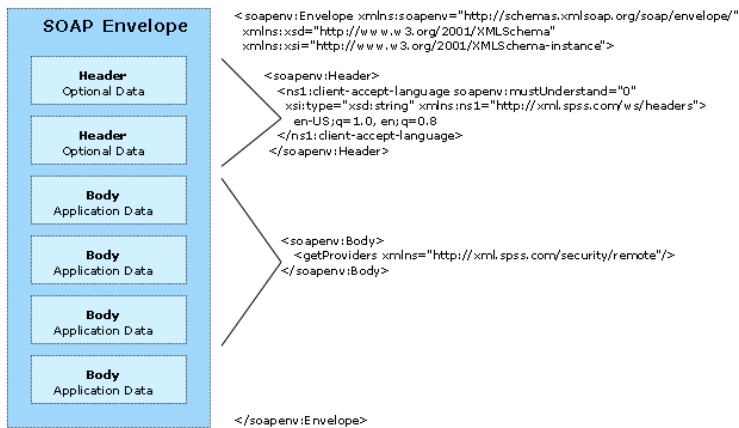


Figure 3. An example SOAP packet

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 target programming language.

In accordance with the [WSDL specification](#) 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. See the topic [“Types”](#) on page 3 for more information.
- **Messages.** Input and output definitions for the web service operations. See the topic [“Messages”](#) on page 4 for more information.
- **PortTypes.** Groups of operations offered by the web service. See the topic [“Port types”](#) on page 4 for more information.
- **Bindings.** Protocols and formats for the web service operations. See the topic [“Bindings”](#) on page 4 for more information.
- **Services.** Endpoints at which the web service functionality can be accessed. See the topic [“Services”](#) on page 5 for more information.

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 example 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>
```

```
</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 example 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. See the topic [“Types”](#) on page 3 for more information.

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 example 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. See the topic [“Messages”](#) on page 4 for more information.

Bindings

The binding element of a WSDL file binds the interface defined by the port type to transport and messaging protocols. Consider the following example 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 `wsdl:soap:binding` element defines HTTP as the transport protocol. The `getProviders` operation in the interface is 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 example service definition:

```
<wsdl:service name="ProviderInformationService">
  <wsdl:port binding="impl:ProviderInformationSoapBinding" name="ProviderInformation">
    <wsdl:soap: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. Content Repository URI Service overview

The IBM SPSS Collaboration and Deployment Services Repository provides the storage facilities for IBM SPSS Collaboration and Deployment Services. The repository stores objects in a hierarchical system similar to the folder/file structure used in operating systems. The objects themselves may exist in several different versions to accommodate and track changes to the object over time.

The Content Repository URI Service provides remote access to the repository for general storage and retrieval of content and meta-data. For example, a new object can be stored in the repository with particular name and keyword values. Retrieval of the object also returns the meta-data associated with it. The Content Repository URI Service offers functionality similar to that found in the Content Repository Service. The primary difference is the use of **uniform resource identifiers** to reference objects stored in the repository instead of identification specifiers.

Accessing the Content Repository URI Service

To access the functionality offered by the Content Repository URI 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>/<context-root>/cr-ws/services/ContentRepositoryURI
```

The value of *<host-name>* corresponds to the name or IP address of the machine on which IBM SPSS Collaboration and Deployment Services Repository is installed.

Note: An IPv6 address must be enclosed in square brackets, such as `[3ffe:2a00:100:7031::1]`.

The value of *<port-number>* indicates the port number on which the repository server is running. The *<context-root>* value specifies the custom context root, if any, configured for your repository server. If your system does not use a context root, omit this portion of the endpoint. 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* without a context root, the WSDL file can be accessed using the path:

```
http://cads_server:80/cr-ws/services/ContentRepositoryURI?wsdl
```

Calling Content Repository URI 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 = "/cr-ws/services/ContentRepositoryURI";
URL url = new URL("http", "cads_server", 80, context);
ContentRepositoryURIService service = new ContentRepositoryURIServiceLocator();
stub = service.getContentRepositoryURI(url);
```

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

```
stub.getAllVersions(uri);
```

The “[The cancelTransfer operation](#)” on page 16 contains examples of calling the operations available in the service. These examples are based on proxy code generated from the WSDL file using the Axis WSDL-to-Java tool. Proxy code generated using another tool may differ slightly from that shown here. For example, the Axis tool interprets the type `xs:dateTime` as the `java.util.Calendar` class while the Castor tool interprets it as a `java.util.Date`. As a result, the classes used in the examples may differ from an implementation based on another tool.

Chapter 3. Repository Concepts

Resources

A resource is an item stored within the repository, such as a folder, a file, or a topic. Any resource in the repository has a collection of metadata associated with it, including:

- **ID.** A unique identifier for the resource often used by web service operations to reference the repository resource being manipulated.
- **Version.** List of version identifiers for the resource. The specification of a version and an ID uniquely identifies a resource.
- **Creator.** The principal who created the resource.
- **Creation date.** The date and time the resource was created in the repository.
- **Modification Date.** The last date and time the resource was modified.
- **Title.** The resource name.
- **Description.** Text describing the resource. For non-English text, the language should be specified for proper processing.
- **Path.** The path to the resource in the repository hierarchy. The repository stores resources in hierarchies that depend on the resource type. As a result, the path must define which hierarchy type to use for the resource. Valid hierarchy types include folder, topic, configuration, server, credential, datasource, enterprise, and submitted.

In addition to this base meta-data, individual resource types may include their own custom meta-data. For example, files include content size.

Some resources are **controlled**, meaning they contain a meaningful access control list (ACL). The ACL defines the permissions for principals that can access the resource. For example, the resource may only allow read access to everyone but the principal who created it. That principal would have total control over the resource.

A resource can be **locked** by a user to prevent others from modifying the properties or content of any version of the resource. Other users can view a locked resource, but modifications are not allowed until the lock is removed. Locks do not expire, and can be removed only by the user who locked the resource or an administrator who has the appropriate action associated with their role.

The Content Repository URI Service includes operations for creating, retrieving, updating, copying, moving, and deleting resources within the repository.

Folders

Folder resources provide an organization mechanism for file resources based on storage location similar to the folder structure of Windows and the directory structure of Unix. The title of the resource in the folder hierarchy corresponds to the name of the folder. The resource path defines the location of the resource in the folder hierarchy. The contents of a folder are referred to as the **children** of the folder and may be either files or other folders.

Construction of a folder hierarchy using the Content Repository URI Service involves use of the `createResource` operation to define folders. Create a child for a folder by specifying the resource path when creating or updating the resource for the child.

Files

A file resource corresponds to a simple file stored within the IBM SPSS Collaboration and Deployment Services Repository, such as a IBM SPSS Modeler stream or a IBM SPSS Statistics syntax file. In addition

to the actual file content and the meta-data associated with general resources, a file resource includes the following information:

- **Author.** The author of the file.
- **MIME type.** Information indicating the media type contained within the file.
- **Size.** File size in bytes.
- **Language.** Language of the file content.
- **Topic list.** A list of topics associated with the file.
- **Keyword list.** A list of searchable keywords associated with the file.
- **Expiration date.** The date and time to remove the file from the repository.

Use the `getFile` operation to retrieve a file from the repository. If the file metadata only is needed, use the `getResource` operation.

Topics

Topics allow the definition of a classification system for the content stored in the IBM SPSS Collaboration and Deployment Services Repository, providing a hierarchical map to guide users to the resources they need. Topics function like a directory structure but differ from directories in that a single object can be listed under multiple topics.

For example, you might want to create a topic structure that mirrors your organization, with separate topics for marketing, finance, development, and so on. Users can then choose from the available topics when storing content. In addition, users can limit content searches to specific topics to accelerate the retrieval process. Because a given item can be listed under multiple topics, cross-indexing is also possible.

Alternatively, consider the following topic hierarchy, in which the model type is the basis for classifying resources:

```
Model
  association
    Apriori
    CARMA
    GRI
    Sequence
  clustering
    K-means
    Kohonen
    TwoStep
  decision tree
    C5
    CHAID
    C&RT
    QUEST
  neural network
  screening
    Anomaly detection
    Feature selection
  statistical
    factor PCA
    linear regression
    logistic regression
  text extraction
```

Any model in the repository could be assigned a topic in this hierarchy to assist in finding desired resources. For example, a user might want to find all association models that use a specific field. Alternatively, the search may be restricted to CARMA models only.

Topic resources provide an organization mechanism for file resources based on a keyword hierarchy. The title of the resource in the topic hierarchy denotes the name of the topic. The resource path defines the location of the resource in the topic hierarchy. Topics appearing under other topics in the topic hierarchy are referred to as subtopics, or **children** of the parent topic. For example, in the model type topic hierarchy, the *Anomaly Detection* and *Feature Selection* topics are children of the *screening* topic, which is itself a child of the *Model* topic.

Construction of a topic hierarchy using the Content Repository URI Service involves repeated use of the `createResource` operation to define topics and their parent-child relationships. Assign topics from the resulting hierarchy to files using the `updateResource` operation to modify the list of associated topics for the file.

Uniform Resource Identifiers

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

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

The URI has the following format:

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

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

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

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

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

Optional version fragments follow the object information. The fragments begin with a hash symbol (#), followed by a single letter indicating whether the fragment is a version label (l) or a version timestamp marker (m). The fragment ends with a period and the actual label or marker for the version. Replace any spaces in the label or marker with escape characters. For example, the URI:

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

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

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

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

Specifiers

To preserve polymorphism of both parameters and return values across the web service for clients implemented in varying programming languages, the service uses wrappers, or **specifiers**, for passing resources and resource identifiers.

Resource specifier. Wraps any resource, such as a file or topic. A resource specifier is a common parameter and return value for operations dealing with resources. For convenience, a client should include a method that creates a resource specifier from a resource, similar to the following Java example:

```
public static ResourceSpecifier getResourceSpecifier(Resource resource) {  
    ResourceSpecifier rs = new ResourceSpecifier();  
    rs.setResource(resource);  
}
```

```
    return is;
}
```

Identification specifier. Wraps both the identifier and version information for a resource, providing a way for the client to uniquely identify an object. This specifier also preserves polymorphism over operations involving the identifier information, which can be either a resource identifier (`ResourceID`) or a resource path (`ResourcePath`). For convenience, a client should include a method that creates an identification specifier from an identifier and a version specification, similar to the following Java example:

```
public static IdentificationSpecifier getIdentificationSpecifier(
    Identifier id, Version version) {
    IdentificationSpecifier is = new IdentificationSpecifier();
    is.setIdentifier(id);
    is.setVersion(version);
    return is;
}
```

Custom properties

The standard metadata for resources can be extended to include information from user-defined custom properties. For example, a *Reviewer* property could be assigned to files to denote the person responsible for reviewing the file before moving it into production. The default metadata for the resource does not include this property so a custom property must be defined. Working with custom properties involves the following two distinct approaches:

- Defining the properties
- Assigning values to the properties for specific objects

Defining custom properties

Custom property definition involves the specification of three criteria: the property type, the property label, and the property reach. The custom property type determines the type of information that can be stored within the property. Available property types include:

- `string`. Values for the custom property correspond to string data.
- `number`. The custom property values are numbers.
- `boolean`. Values for the custom property represent binary choices, such as *yes/no* or *true/false*.
- `single choice`. A value for this property type represents a single selection from a list of possible alternatives.
- `multiple select`. Values for the custom property correspond to a one or more selections from a list of possible alternatives.

The custom property type can be used by client interfaces to present appropriate controls for manipulating the property value. For example, if the server indicates that a property is a boolean, the client can use an option (radio) button to allow the user to modify the value. The property label provides text that can be used to describe the control.

The reach of a custom property specifies which types of repository objects include the property. Custom properties can be applied to all objects within the repository or restricted to certain types of objects, such as files, folders, or jobs.

The system includes operations for creating, retrieving, updating, and deleting custom properties.

Assigning custom property values

A newly created custom property is immediately available to all repository objects of the type defined by the property. For example, the custom property "Reviewer" may be defined for all jobs and files in the repository, but not for folders. Every job and file can be assigned a value for this property. Initially the property value will be a null or default value for each object, so assigning a value corresponds to updating the existing value. Updating a custom property value for an object requires the specification of both the new value for the property and an identifier for the object being assigned the value.

The system includes operations for retrieving and updating custom property values for objects within the repository.

Transfers

Transfers involve the exchange of resources between two repository instances or between a repository and a file system. Typically this involves the export of a folder from one repository and the subsequent import of that folder into another. For any successful transfer, the following two criteria must be defined:

- The source to transfer
- The target destination for the source being transferred

Both the source and target are specified using the path information for the resources. For example, if the source being transferred is the *Modeling* folder from repository A and the target path corresponds to the *Analysis* folder of repository B, the transfer creates a *Modeling* folder as a child of the *Analysis* folder in repository B.

The transfer definition also includes the specification of a policy for handling conflicts that occur if the target destination already contains a child having the same title as the resource being transferred. Possible conflict resolutions include the following:

- Not creating any new versions if a resource already exists in the target location
- Deleting existing target versions and replacing them with imported versions from the source
- Adding new versions to the target resource corresponding to the versions from the source

When adding new versions, a policy for handling label conflicts can also be defined. For example, suppose a version of the file *QuarterlyResults.sps* in the transfer source has the label *Production* and the transfer target contains a file with the same name in the same location. If conflicts are handled by appending new versions to the target and a version of the target file already has the label *Production*, a version label conflict occurs. A policy for resolving conflicts of this type determines whether the source or target version retains the conflicting label during a transfer.

The system includes operations for initiating, monitoring, and canceling resource transfers.

Chapter 4. Operation reference

The applyTransferPolicy operation

Applies specified policies to resolve conflicts arising during a resource transfer.

Input fields

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

Table 1. Fields for applyTransferPolicy		
Field	Type/Valid Values	Description
TransferIdentifier	TransferIdentifier	Unique identifier of the resource transfer.
CompositeTransferPolicy	CompositeTransferPolicy	Defines a set of the resource transfer policies.

Java example

To create and apply policies for resolving conflicts:

1. Create a CompositeTransferPolicy object to define the set of policies.
2. Create a policy object for each default policy. Modify the policy properties as needed. Add each policy object to the composite policy using the addTransferPolicy method. During conflict resolution, policies are applied in the order in which they were added to the composite policy so always add default policies first.
3. Create a policy object for each policy being applied to individual resources. Modify the policy properties as needed.
4. For each policy type, create a ResourceURI object and assign the resource URI for the resource to which the policy will be applied. Use the addResourceIdentifier method to add the URIs to the policy object.
5. Add each policy object to the composite policy using the addTransferPolicy method.
6. Supply the applyTransferPolicy operation with the identifier for the transfer and the overall policy object.

The following sample defines no change as the default policy. In addition, no change and append policies are used for specific resources.

```
CompositeTransferPolicy compositeTransferPolicy = new CompositeTransferPolicy();
// our default is "no change", this will apply to all the resources
NoChangeImportPolicy noChangeImportPolicy = new NoChangeImportPolicy();
compositeTransferPolicy.addTransferPolicy(noChangeImportPolicy);

noChangeImportPolicy.setInvalidVersions(true);
ResourceURI resourceURI = new ResourceURI();
resourceURI.setValue("spsscr:///id=0a0a4a356c24e80b0000011585ab4a0b854e");
noChangeImportPolicy.addResourceIdentifier(resourceURI);
compositeTransferPolicy.addTransferPolicy(noChangeImportPolicy);

AppendImportPolicy appendImportPolicy = new AppendImportPolicy();
appendImportPolicy.setVersionLabelPolicy(VersionLabelPolicy.SOURCE);
resourceURI.setValue("spsscr:///id=0a0a4a356c24e80b0000011585ab4a0b8562");
appendImportPolicy.addResourceIdentifier(resourceURI);
compositeTransferPolicy.addTransferPolicy(appendImportPolicy);

stub.applyTransferPolicy(transferId, compositeTransferPolicy);
```

Note that the resource URIs in this sample do not include version information. As a result, the policies are applied at the resource level.

To resume importing using the applied policies, use the `setExecute` method to set execution to `true` for the composite policy.

SOAP request example

Client invocation of the `applyTransferPolicy` 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>
    <applyTransferPolicy xmlns="http://xml.spss.com/repository/remote">
      <ns2:TransferIdentifier engineVersion="2.0" xmlns:ns2="http://xml.spss.com/repository"
        >603g8f7e81</ns2:TransferIdentifier>
      <ns3:CompositeTransferPolicy execute="true" xmlns:ns3="http://xml.spss.com/repository">
        <ns3:TransferPolicy invalidVersions="true" xsi:type="NoChangeImportPolicy"/>
        <ns3:TransferPolicy xsi:type="NoChangeImportPolicy">
          <ns3:resourceIdentifier value="spsscr:///id=0a0a4a356c24e80b0000011585ab4a0b854e"
            xsi:type="ResourceURI"/>
        </ns3:TransferPolicy>
        <ns3:TransferPolicy versionLabelPolicy="source" xsi:type="AppendImportPolicy">
          <ns3:resourceIdentifier value="spsscr:///id=0a0a4a356c24e80b0000011585ab4a0b8562"
            xsi:type="ResourceURI"/>
        </ns3:TransferPolicy>
      </ns3:CompositeTransferPolicy>
    </applyTransferPolicy>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `applyTransferPolicy` 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>
    <applyTransferPolicyResponse xmlns="http://xml.spss.com/repository/remote"/>
  </soapenv:Body>
</soapenv:Envelope>
```

The cancelTransfer operation

Cancels a resource transfer.

Input fields

The following table lists the input fields for the `cancelTransfer` operation.

Table 2. Fields for cancelTransfer		
Field	Type/Valid Values	Description
TransferIdentifier	TransferIdentifier	Unique identifier of the resource transfer.

Java example

The following sample cancels the transfer with identifier *transferId*.

```
stub.cancelTransfer(transferId);
```

SOAP request example

Client invocation of the `cancelTransfer` 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>
    <cancelTransfer xmlns="http://xml.spss.com/repository/remote">
      <TransferIdentifier xmlns="http://xml.spss.com/repository">5d9485bag0</TransferIdentifier>
    </cancelTransfer>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `cancelTransfer` 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>
    <cancelTransferResponse xmlns="http://xml.spss.com/repository/remote"/>
  </soapenv:Body>
</soapenv:Envelope>
```

The `cascadePermissions` operation

Allows a client to set permissions on a resource and cascade the permission to all children. Permissions in the file system are not intrinsically inherited from parent resources. This does not mean that a child file of a folder will not have the same permissions as the parent folder, and indeed this will occur by default. However, the children of a resource can be updated with custom permissions. As a result, it may be necessary when changing permissions on a folder to either cascade the entire new access control list or the new permission down to lower levels.

Input fields

The following table lists the input fields for the `cascadePermissions` operation.

Table 3. Fields for <code>cascadePermissions</code>		
Field	Type/Valid Values	Description
parentURI	string	URI for the resource whose permissions are being cascaded.
Options	Options	Options to cascade permission deltas as well as set permission on the parent.
AccessControlList	AccessControlList	Defines entire scope of permission on a controlled resource for all principals that have access.

Table 3. Fields for cascadePermissions (continued)		
Field	Type/Valid Values	Description
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Java example

Cascading permissions involves the following steps:

1. Define the URI for the parent object whose permissions are to be cascaded.
2. Use an Options object to define the cascading behavior.
3. Specify the permissions to cascade using an AccessControlList object, which contains a list of AccessControlEntry objects. An AccessControlEntry object pairs a principal with a permission.
4. Define the owner of the AccessControlList object.
5. Supply the cascadePermissions operation with the URI, cascade options, and access control list.

The following sample cascades the permission changes to the file *E01_FindDuplicates.str*.

```
String uri = "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Understanding/E01_FindDuplicates.str";

Options opt = new Options();
opt.setCascadeDelta(true);
opt.setSetParent(true);

AccessControlList acl = new AccessControlList();
AccessControlEntry[] ace = new AccessControlEntry[3];
ace[0].setPermission(PermissionType.READ);
Principal principal = new Principal();
principal.setID("//gNative//$$security/everyoneGroup");
principal.setDisplayName("-- everyone --");
principal.setName("$$security/everyoneGroup");
principal.setIsGroup(true);
ace[0].setPrincipal(principal);
ace[1].setPermission(PermissionType.READ);
principal.setID("//uNative//validUser");
principal.setDisplayName("validUser");
principal.setName("validUser");
principal.setIsGroup(false);
ace[1].setPrincipal(principal);
ace[2].setPermission(PermissionType.WRITE);
principal.setID("//uNative//validUser");
principal.setDisplayName("validUser");
principal.setName("validUser");
principal.setIsGroup(false);
ace[2].setPrincipal(principal);
acl.setAccessControlEntry(ace);
principal.setID("//uNative//admin");
principal.setDisplayName("admin");
principal.setName("admin");
principal.setIsGroup(false);
acl.setOwner(principal);

stub.cascadePermissions(uri, opt, acl, null);
```

SOAP request example

Client invocation of the cascadePermissions 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>
    <cascadePermissions xmlns="http://xml.spss.com/repository/remote">
      <ns2:ParentIdentificationSpecifier xsi:type="IdentificationSpecifier"
        xmlns:ns2="http://xml.spss.com/repository">
        <ns2:identifier value="0a0a4a35d98ee53f0000010ea9597eda80fa" xsi:type="ResourceID"/>
      </ns2:ParentIdentificationSpecifier>
      <ns3:Options cascadedelta="true" setparent="true" xmlns:ns3="http://xml.spss.com/repository"/>
      <ns4:AccessControllist xmlns:ns4="http://xml.spss.com/repository">
        <ns4:AccessControlEntry Permission="READ">
          <ns4:Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
            Name="$$security/everyoneGroup" IsGroup="true"/>
        </ns4:AccessControlEntry>
        <ns4:AccessControlEntry Permission="READ">
          <ns4:Principal ID="//uNative//validUser" DisplayName="validUser" Name="validUser"
            IsGroup="false"/>
        </ns4:AccessControlEntry>
        <ns4:AccessControlEntry Permission="WRITE">
          <ns4:Principal ID="//uNative//validUser" DisplayName="validUser" Name="validUser"
            IsGroup="false"/>
        </ns4:AccessControlEntry>
        <ns4:owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
      </ns4:AccessControllist>
    </cascadePermissions>
  </soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a cascadePermissions 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>
    <cascadePermissionsResponse xmlns="http://xml.spss.com/repository/remote"/>
  </soapenv:Body>
</soapenv:Envelope>

```

The copyResource operation

Copies a resource to a specified parent. The resource version can be specified in the URI. If there is an attempt to copy the resource to a topic or a file parent, an exception is thrown. The operation returns the URI of the newly created object.

Input fields

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

Table 4. Fields for copyResource		
Field	Type/Valid Values	Description
targetParentURI	string	URI for the destination of the copy.
sourceURI	string	URI for the resource being copied.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

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

Table 5. Return Value	
Type	Description
string	URI for the new resource copy.

Java example

The following sample copies the file *P01_AgeCalculations.str* to the folder *Streams* at the root of the repository.

```
String targetURI = "spsscr://pes_server:80/Streams";
String sourceURI =
    "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str";
String newURI = stub.copyResource(targetURI, sourceURI, null);
```

SOAP request example

Client invocation of the `copyResource` 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>
    <copyResource xmlns="http://xml.spss.com/repository/remote">
      <targetParentURI>spsscr://pes_server:80/Streams</targetParentURI>
      <sourceURI>
        spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str</sourceURI>
      </copyResource>
    </soapenv:Body>
  </soapenv:Envelope>
```

SOAP response example

The server responds to a `copyResource` 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>
    <copyResourceResponse xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr:///?id=0a0a4a359cb71b550000010e75728dd183f5#m.0:2006-10-23%2014:07:37.814</uri>
    </copyResourceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The createCustomProperty operation

Creates a custom property that can be assigned values for resources.

Input fields

The following table lists the input fields for the `createCustomProperty` operation.

Table 6. Fields for createCustomProperty		
Field	Type/Valid Values	Description
CustomProperty	CustomProperty	Metadata describing a custom property defined by a customer to apply to resources.

Return information

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

Table 7. Return Value	
Type	Description
string	The identifier of the created custom property.

Java example

The following sample creates a custom property having the label *Reviewed*. The AppliesTo object defines the reach of the property while the Constraint object defines its type.

```
CustomProperty cp = new CustomProperty();
cp.setLabel("Reviewed");

AppliesTo at = new AppliesTo();
at.setFolderApplicable(false);
at.setJobApplicable(true);
cp.setAppliesTo(at);

Constraint cnst = new Constraint();
Freeform ff = new Freeform();
ff.setType(CustomPropertyValue.B00LEAN);
cnst.setFreeform(ff);
cp.setConstraint(cnst);

String id = stub.createCustomProperty(cp);
```

The AppliesTo object includes three methods for defining the property reach, setFolderApplicable, setJobApplicable, and setFileApplicable. The first two accept boolean arguments indicating whether or not the property applies to folders and jobs. The setFileApplicable method limits a custom property to specific file types defined by a FileApplicable object corresponding to the MIME types of the files. In the absence of such a specification, the custom property being defined applies to all files.

To define the Constraint object, create a Freeform object and set its type to the type for the property using setType. Use setFreeform to assign the type to the Constraint object.

After assigning the reach and type to the property using setAppliesTo and setConstraint, add the custom property to the system using the createCustomProperty operation for the stub.

SOAP request example

Client invocation of the createCustomProperty 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">Native/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>
    <createCustomProperty xmlns="http://xml.spss.com/repository/remote">
      <CustomProperty label="Reviewed" xmlns="http://xml.spss.com/repository">
        <appliesTo>
          <fileApplicable/>
        </appliesTo>
      </CustomProperty>
    </createCustomProperty>
  </soapenv:Body>
</soapenv:Envelope>
```

```

        <jobApplicable>true</jobApplicable>
        <folderApplicable>>false</folderApplicable>
      </appliesTo>
      <constraint>
        <freeform type="boolean"/>
      </constraint>
    </CustomProperty>
  </createCustomProperty>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a createCustomProperty 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>
    <createCustomPropertyResponse xmlns="http://xml.spss.com/repository/remote">
      <identifier>0a0a4a3567b4a2cc0000010fed740baa800b</identifier>
    </createCustomPropertyResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

The createOrAddResource operation

Creates a resource in the repository. If the resource already exists, this operation adds a new version of it. The resource may be a folder, topic, or file.

Input fields

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

Table 8. Fields for createOrAddResource		
Field	Type/Valid Values	Description
parentURI	string	URI for the parent of the resource being created.
ResourceSpecifier	ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.
PropagateVersionSpecifier	VersionSpecifier	This Type is used to specify a particular version of an object, either by Marker or Label. It is distinguished from a Version Type by the fact that Version is used to return information about an Object's version, and may contain multiple labels.

Return information

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

Table 9. Return Value	
Type	Description
string	URI for the new resource.

Java example

Creating a resource either as a new version of an existing resource or a new item typically involves the following steps:

1. Create a string indicating the URI for the parent of the resource being created.
2. Create a `ResourceSpecifier` object to wrap the resource object being created.
3. Create a resource object corresponding to the item being created: `Folder` for folders, `Topic` for topics, and `File` for files.
4. Define the properties for the resource object, such as its title.
5. Add the resource object to the `ResourceSpecifier` object.
6. Create a `VersionSpecifier` object to identify the version from which the metadata should be propagated if the resource already exists. Use either the label or marker for the version.
7. Add the resource object to the repository using the `createOrAddResource` operation.

The following code adds a file to the *Images* folder. If the file already exists, a new version is created, propagating the metadata from the *PRODUCTION* version.

```
String parentURI = "spsscr://pes_server:80/Images";
ResourceSpecifier specRes = new ResourceSpecifier();
File f = new File();
Title t = new Title();
t.setValue("property.svg");
f.setTitle(t);
MimeType mt = new MimeType();
mt.setValue("image/svg+xml");
f.setMimeType(mt);
Author a = new Author();
a.setValue("sbennett");
f.setAuthor(a);
URL("file:///c:/property.svg");
URLDataSource uds = new URLDataSource(url);
DataHandler dh = new DataHandler(uds);
AttachmentPart at = new AttachmentPart(dh);
BinaryContent bc = new BinaryContent();
Attachment att = new Attachment();
att.setHref(at.getContentId());
bc.setAttachment(att);
f.setBinaryContent(bc);
((org.apache.axis.client.Stub)stub).addAttachment(at);
specRes.setResource(f);
VersionSpecifier prop = new VersionSpecifier();
prop.setLabel("PRODUCTION");
String uri = stub.createOrAddResource(parentURI, specRes, null, prop);
```

The createResource operation

Creates a resource in the repository. The resource may be a folder, topic, file, or new version of an existing file.

Input fields

The following table lists the input fields for the `createResource` operation.

Table 10. Fields for createResource		
Field	Type/Valid Values	Description
parentURI	string	URI for the parent of the resource being created.

Table 10. Fields for createResource (continued)		
Field	Type/Valid Values	Description
ResourceSpecifier	ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

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

Table 11. Return Value	
Type	Description
string	URI for the new resource.

Java example

Creating a resource in the repository typically involves the following steps:

1. Create a string indicating the URI for the parent of the resource being created.
2. Create a resource object corresponding to the item being created: Folder for folders, Topic for topics, and File for files.
3. Define the properties for the resource object, such as its title.
4. Add the resource object to the repository using createResource.

The following code adds the folder *Statistics files* to the root level of the repository.

```
String parentURI = "spsscr://pes_server:80";
ResourceSpecifier specifier = new ResourceSpecifier();
Folder folder = new Folder();
Title title = new Title();
title.setValue("Statistics files");
folder.setTitle(title);
specifier.setResource(folder);
String newURI = stub.createResource(parentURI, specifier, null);
```

SOAP request example

Client invocation of the createResource 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>
```

```

<createResource xmlns="http://xml.spss.com/repository/remote">
  <parentURI>spsscr://pes_server:80</parentURI>
  <ns3:ResourceSpecifier xmlns:ns3="http://xml.spss.com/repository">
    <ns3:Resource xsi:type="Folder">
      <ns3:Title value="Statistics files"/>
    </ns3:Resource>
  </ns3:ResourceSpecifier>
</createResource>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a createResource 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>
    <createResourceResponse xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr:///id=0a0a4a3526737cd10000010ec7edde5680d4</uri>
    </createResourceResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

The createResourcePropagate operation

Creates a resource in the repository, propagating the metadata from a specified resource version to the new resource. The resource may be a folder, topic, file, or new version of an existing file.

Input fields

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

Table 12. Fields for createResourcePropagate		
Field	Type/Valid Values	Description
parentURI	string	URI for the parent of the resource being created.
ResourceSpecifier	ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.
PropagateVersionSpecifier	VersionSpecifier	This Type is used to specify a particular version of an object, either by Marker or Label. It is distinguished from a Version Type by the fact that Version is used to return information about an Object's version, and may contain multiple labels.

Return information

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

Table 13. Return Value	
Type	Description
string	URI for the new resource.

Java example

Propagating properties to a newly created resource typically involves the following steps:

1. Create a string indicating the URI for the parent of the resource being created.
2. Create a `ResourceSpecifier` object to wrap the resource object being created.
3. Create a resource object corresponding to the item being created: `Folder` for folders, `Topic` for topics, and `File` for files.
4. Define the properties for the resource object, such as its title.
5. Add the resource object to the `ResourceSpecifier` object.
6. Create a `VersionSpecifier` object to identify the version from which the metadata should be propagated. Use either the label or marker for the version.
7. Add the resource object to the repository using the `createResourcePropagate` operation.

The following code creates a new version of a file in the *Images* folder, propagating the metadata from the *PRODUCTION* version.

```
String parentURI = "spsscr://pes_server:80/Images";
ResourceSpecifier specRes = new ResourceSpecifier();
File f = new File();
Title t = new Title();
t.setValue("property.svg");
f.setTitle(t);
MimeType mt = new MimeType();
mt.setValue("image/svg+xml");
f.setMimeType(mt);
Author a = new Author();
a.setValue("sbennett");
f.setAuthor(a);
URL("file:///c:/property.svg");
URLDataSource uds = new URLDataSource(url);
DataHandler dh = new DataHandler(uds);
AttachmentPart at = new AttachmentPart(dh);
BinaryContent bc = new BinaryContent();
Attachment att = new Attachment();
att.setHref(at.getContentId());
bc.setAttachment(att);
f.setBinaryContent(bc);
((org.apache.axis.client.Stub)stub).addAttachment(at);
specRes.setResource(f);
VersionSpecifier prop = new VersionSpecifier();
prop.setLabel("PRODUCTION");
String uri = stub.createResourcePropagate(parentURI, specRes, null, prop);
```

SOAP request example

Client invocation of the `createResourcePropagate` 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">Native/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>
    <createResourcePropagate xmlns="http://xml.spss.com/repository/remote">
      <parentURI>spsscr://pes_server:80/Images</parentURI>
      <ResourceSpecifier xmlns="http://xml.spss.com/repository">
        <Resource xsi:type="File" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <Title value="property.svg"/>
          <Author value="sbennett"/>
          <MimeType value="image/svg+xml"/>
          <BinaryContent>
            <Attachment href="B0245F71E6C368E83D2B99EE9645C5DF"/>
          </BinaryContent>
        </Resource>
      </ResourceSpecifier>
    </createResourcePropagate>
  </soapenv:Body>
</soapenv:Envelope>
```

```

        </BinaryContent>
      </Resource>
    </ResourceSpecifier>
    <PropagateVersionSpecifier label="PRODUCTION" xmlns="http://xml.spss.com/repository"/>
  </createResourcePropagate>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a createResourcePropagate 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>
    <createResourcePropagateResponse xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr:///id=0a0a4aac00072ffb00000106f3f7b05b3a03</uri>
    </createResourcePropagateResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

The createUniqueSubmittedFolder operation

For internal use only. Consumers of the web service should not use this operation.

Input fields

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

Table 14. Fields for createUniqueSubmittedFolder		
Field	Type/Valid Values	Description
workName	string	The name which is passed from client side.

Return information

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

Table 15. Return Value	
Type	Description
IdentificationSpecifier	Wrapper for identification information(Id, path, version label, version marker).Used to define service interface parameters and return values and preserve polymorphism of wrapped objects.

The deleteCustomProperty operation

Deletes a custom property from the repository.

Input fields

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

Table 16. Fields for deleteCustomProperty		
Field	Type/Valid Values	Description
identifier	string	Identifier for the custom property.

Java example

The following function accepts a string corresponding to the identifier for the custom property to be deleted. The deleteCustomProperty operation uses this information to delete the property from the repository.

```
public void deleteCustomProperty(String customPropertyID)
    throws IOException, ServiceException, RepositoryException {
    ContentRepository repository = getContentRepository();
    repository.deleteCustomProperty(customPropertyID);
}
```

SOAP request example

Client invocation of the deleteCustomProperty 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">Native/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>
    <deleteCustomProperty xmlns="http://xml.spss.com/repository/remote">
      <identifier>0a0a4aac00072ffb00000106f3f7b05b348f</identifier>
    </deleteCustomProperty>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a deleteCustomProperty 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>
    <deleteCustomPropertyResponse xmlns="http://xml.spss.com/repository/remote"/>
  </soapenv:Body>
</soapenv:Envelope>
```

The deleteResource operation

Deletes a specified resource from the repository. To delete a specific version of the resource, include that information in the URI. If there is not a version specified, the latest version is deleted.

Input fields

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

Table 17. Fields for deleteResource		
Field	Type/Valid Values	Description
uri	string	URI for the resource to delete.

Table 17. Fields for deleteResource (continued)

Field	Type/Valid Values	Description
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Java example

The following example deletes the latest version of the file *P05_Median.str* from the repository.

```
String uri = "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P05_Median.str";
stub.deleteResource(uri, null);
```

SOAP request example

Client invocation of the deleteResource 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>
    <deleteResource xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P05_Median.str</uri>
    </deleteResource>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a deleteResource 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>
    <deleteResourceResponse xmlns="http://xml.spss.com/repository/remote"/>
  </soapenv:Body>
</soapenv:Envelope>
```

The disposeTransfer operation

Frees system resources allocated on the server for a resource transfer.

Input fields

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

Table 18. Fields for disposeTransfer		
Field	Type/Valid Values	Description
TransferIdentifier	TransferIdentifier	Unique identifier of the resource transfer.

Java example

The following sample frees server resources allocated to the transfer with identifier *transferId*.

```
stub.disposeTransfer(transferId);
```

SOAP request example

Client invocation of the `disposeTransfer` 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>
    <disposeTransfer xmlns="http://xml.spss.com/repository/remote">
      <TransferIdentifier xmlns="http://xml.spss.com/repository">5d947f5e7f</TransferIdentifier>
    </disposeTransfer>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `disposeTransfer` 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>
    <disposeTransferResponse xmlns="http://xml.spss.com/repository/remote"/>
  </soapenv:Body>
</soapenv:Envelope>
```

The findAllLabels operation

Queries the repository for all labels currently assigned to objects in the hierarchical content repository. This includes labels in use for files and jobs.

Return information

The following table identifies the information returned by the `findAllLabels` operation.

Table 19. Return Value	
Type	Description
string[]	Labels in use in the repository.

Java example

The following function returns all labels as an array of strings.

```
String[] labels = stub.findAllLabels();
System.out.println("Labels in use:");
for (int i = 0; i < labels.length; i++) {
    System.out.println(labels[i]);
}
```

SOAP request example

Client invocation of the `findAllLabels` 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">Native/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>
    <findAllLabels xmlns="http://xml.spss.com/repository/remote"/>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `findAllLabels` 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>
    <findAllLabelsResponse xmlns="http://xml.spss.com/repository/remote">
      <label>Modeler Output</label>
      <label>Production</label>
    </findAllLabelsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The `getActions` operation

Retrieves the list of actions, if any, that are defined for a resource.

Input fields

The following table lists the input fields for the `getActions` operation.

Table 20. Fields for <code>getActions</code>		
Field	Type/Valid Values	Description
<code>uri</code>	string	URI for the resource.
<code>ResourceRetrievalOptions</code>	<code>ResourceRetrievalOptions</code>	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

The following table identifies the information returned by the `getActions` operation.

Table 21. Return Value	
Type	Description
ActionList	List of action specifications.

Java example

The following example returns an `ActionList` object containing an array of `ActionSpecificationType` objects for the file *P01_AgeCalculations.str*. For each `ActionSpecificationType` object, the `getAction` method returns the action and the `getSpecification` returns the actual specification for the action.

```
String uri = "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str";
ActionList alist = stub.getActions(uri, null);
ActionSpecificationType[] aSpecType = alist.getActionSpecification();
for (int i = 0; i < aSpecType.length; i++) {
    System.out.println("Action: " + aSpecType[i].getAction());
    System.out.println("Specification: " + aSpecType[i].getSpecification());
    System.out.println("");
}
```

SOAP request example

Client invocation of the `getActions` 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>
    <getActions xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str</uri>
    </getActions>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getActions` 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>
    <getActionsResponse xmlns="http://xml.spss.com/repository/remote">
      <ActionList>
        <ActionSpecification>
          <specification>&lt;?xml version="1.0" encoding="UTF-8" >&lt;job
            javaClass="com.spss.work.ClementineExecutor"
            repositoryUrl="http://pes_server:80/cr-ws/services/ContentRepository"
            repositoryId="0a0a4a359cb71b550000010e75728dd18211"
            contentType="application/x-vnd.spss-clementine-stream"
            name="General"
            objectType="Clementine.General"
            choice="Clementine.General.ClementineServer"
            name="Modeler server"
            description="Choose a preconfigured Modeler server connection setting"
            itemDisplay="No Modeler servers have been configured"
            value=""
            choice="Clementine.General.Credentials"
            name="Credentials"
            description="Select
```

```

the credentials to use for this job
step</description><<item display="No execution
credentials have been configured" value="A Repository
Resource ID"><<value><A Repository Resource
ID</value></default><A Repository Resource
ID</default></choice></choice
objectType="Clementine.General.Version"
name="Stream
Version"><<description>Choose the version of
this stream to execute</description><<item
display="LATEST"
value="LATEST"></item
display="Production"
value="Production"></item
display="Test"
value="Test"></item><value>LATEST</default>LATEST
</default></choice></node></node
name="Repository"
objectType="Repository.General"></choice
objectType="Repository.General.RepositoryServer"
name="Repository
Server"><<description>Choose a preconfigured
Repository server connection setting</description></item
display="No Content Repository Servers have been
configured"
value=""></item><value></default></default>
</choice></choice
objectType="Repository.General.Credentials"
name="Repository
Credentials"><<description>Select the
credentials to use for this job step</description></item
display="No Content Repository Credentials have been
configured" value=""></item
display="No Content Repository Credentials have been
configured"
value=""></item><value></default></default>
</choice></node></node
name="Results"
objectType="Clementine.Results"></node
name="RowSets"
objectType="Clementine.Results.RowSets"></output
objectType="Clementine.Results.table"
name="Customer Ages" needsFolder=">false"
systemType="any"
default="spsscr://ModelerStreamLibrary/Data%20Preparation/Customer%20Ages"
value="spsscr://ModelerStreamLibrary/Data%20Preparation/Customer%20Ages"><
</description><A
path to a folder in the repository</description></node
name="Output parameters"
objectType="Clementine.Results.RowSets.OutputParameters"></choice
objectType="Clementine.Results.RowSets.OutputParameters.OutputFormat"
name="Output
format"><<description>Specify the format for
the file to store</description></item
display="Modeler native output format"
value="cou"></item
display="Formatted"
value="tab"></item display="Data
(comma delimited)"
value="dat"></item display="HTML
document"
value="html"></item><value>dat</default>dat</default>
</choice></repositoryObject
name="This attr is required by the schema, but shouldn't
be" hidden="true"
value="0a0a4a359cb71b550000010e75728dd1832f"></simple
name="Clementine.Processor.Type"
hidden="true"
type="freeform"></item><value>table</simple></simple
name="Clementine.Processor.Name"
hidden="true"
type="freeform"></item><value>table</simple></node>
</ns1:AccessControlList
poe="CMOR"
xmlns:ns1="http://xml.spss.com/repository"></ns1:AccessControlEntry
Permission="READ"></ns1:Principal
ID="gNative//$$security/everyoneGroup"
DisplayName="-- everyone --"
Name="$$security/everyoneGroup"
IsGroup="true"></ns1:AccessControlEntry></ns1:AccessControlEntry
Permission="READ"></ns1:Principal
ID="uNative//admin"
DisplayName="admin" Name="admin"
IsGroup=">false"></ns1:AccessControlEntry></ns1:owner
ID="uNative//admin"
DisplayName="admin" Name="admin"
IsGroup=">false"></ns1:AccessControlList></output></output
objectType="Clementine.Results.table"
name="Joining Age" needsFolder=">false"
systemType="any"
default="spsscr://ModelerStreamLibrary/Data%20Preparation/Joining%20Age"
value="spsscr://ModelerStreamLibrary/Data%20Preparation/Joining%20Age"><
</description><A
path to a folder in the repository</description></node
name="Output parameters"
objectType="Clementine.Results.RowSets.OutputParameters"></choice
objectType="Clementine.Results.RowSets.OutputParameters.OutputFormat"
name="Output
format"><<description>Specify the format for
the file to store</description></item
display="Modeler native output format"
value="cou"></item
display="Formatted"
value="tab"></item display="Data
(comma delimited)"
value="dat"></item display="HTML
document"
value="html"></item><value>dat</default>dat</default>
</choice></repositoryObject

```

```

name=&quot;This attr is required by the schema, but shouldn't
be&quot; hidden=&quot;true&quot;
value=&quot;0a0a4a359cb71b55000010e75728dd1832e&quot;/&gt;&lt;simple
name=&quot;Clementine.Processor.Type&quot;
hidden=&quot;true&quot;
type=&quot;freeform&quot;&gt;&lt;value&gt;table&lt;/value&gt;&lt;/simple&gt;&lt;simple
name=&quot;Clementine.Processor.Name&quot;
hidden=&quot;true&quot;
type=&quot;freeform&quot;&gt;&lt;value&gt;table&lt;/value&gt;&lt;/simple&gt;&lt;/node&gt;
&lt;ns2:AccessControlList
poe=&quot;CMOR&quot;
xmlns:ns2=&quot;http://xml.spss.com/repository&quot;&gt;&lt;ns2:AccessControlEntry
Permission=&quot;READ&quot;&gt;&lt;ns2:Principal
ID=&quot;gNative//$$security/everyoneGroup&quot;
DisplayName=&quot;-- everyone --&quot;
Name=&quot;$$security/everyoneGroup&quot;
IsGroup=&quot;true&quot;&gt;&lt;/ns2:AccessControlEntry&gt;&lt;ns2:AccessControlEntry
Permission=&quot;READ&quot;&gt;&lt;ns2:Principal
ID=&quot;uNative//admin&quot;
DisplayName=&quot;admin&quot; Name=&quot;admin&quot;
IsGroup=&quot;false&quot;&gt;&lt;/ns2:AccessControlEntry&gt;&lt;ns2:owner
ID=&quot;uNative//admin&quot;
DisplayName=&quot;admin&quot; Name=&quot;admin&quot;
IsGroup=&quot;false&quot;&gt;&lt;/ns2:AccessControlList&gt;&lt;/output&gt;&lt;/node&gt;
&lt;/node&gt;&lt;node
name=&quot;Parameters&quot;
objectType=&quot;Clementine.Parameters&quot;/&gt;&lt;/scheduleParameters&gt;&lt;/job&gt;</specification>
<action>execute</action>
</ActionSpecification>
</ActionList>
</getActionsResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The getAllCustomProperties operation

Retrieves all custom properties currently defined in the repository.

Return information

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

Table 22. Return Value	
Type	Description
CustomProperty[]	Metadata describing a custom property defined by a customer to apply to resources.

Java example

The following function simply uses the stub for the service to return an array containing the complete set of available custom properties. Accessor methods for the returned CustomProperty objects can be used to return the properties for any specific property, such as its label.

```

CustomProperty[] customProperties = stub.getAllCustomProperties();
for (int i = 0; i < customProperties.length; i++) {
    System.out.println("Property label: " + customProperties[i].getLabel());
    System.out.println("Identifier: " + customProperties[i].getIdentifier());
    System.out.println();
}

```

SOAP request example

Client invocation of the getAllCustomProperties 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">Native/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>

```

```
<getAllCustomProperties xmlns="http://xml.spss.com/repository/remote"/>
</soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getAllCustomProperties` 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>
    <getAllCustomPropertiesResponse xmlns="http://xml.spss.com/repository/remote">
      <CustomProperty label="Reviewed?" identifier="0a0a4aac00072ffb000001094fa9f57ed7e8"
        xmlns="http://xml.spss.com/repository">
        <appliesTo>
          <fileApplicable/>
          <jobApplicable>false</jobApplicable>
          <folderApplicable>false</folderApplicable>
        </appliesTo>
        <constraint>
          <freeform type="boolean"/>
        </constraint>
      </CustomProperty>
      <CustomProperty label="Reviewer" identifier="0a0a4aac00072ffb000001094fa9f57ed7e9"
        xmlns="http://xml.spss.com/repository">
        <appliesTo>
          <fileApplicable/>
          <jobApplicable>false</jobApplicable>
          <folderApplicable>false</folderApplicable>
        </appliesTo>
        <constraint>
          <select multipleSelect="false">
            <selectionValue>Andy</selectionValue>
            <selectionValue>Nate</selectionValue>
            <selectionValue>Cory</selectionValue>
            <selectionValue>Rick</selectionValue>
          </select>
        </constraint>
      </CustomProperty>
    </getAllCustomPropertiesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The `getAllLocks` operation

Returns all currently locked resources. For maximum flexibility, the structure of the information returned follows the [webrowset.xsd](#) schema.

Return information

The following table identifies the information returned by the `getAllLocks` operation.

Table 23. Return Value	
Type	Description
RowSetContent	Allows the user to transport a row set either as an out-of-band attachment or as in-band string

Java example

The following sample uses the stub for the service to return a list of all active locks currently in the system. The `getContent` method returns a string containing the `WebRowSet` structure.

```
RowSetContent rowset = stub.getAllLocks();
System.out.println(rowset.getContent());
```

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

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

36 IBM SPSS Collaboration and Deployment Services: Content Repository URI Service Developer's Guide

```

<catalog-name></catalog-name>
<column-type>12</column-type>
<column-type-name>nvarchar</column-type-name>
<column-definition><column-definition>
<column-index>3</column-index>
<auto-increment>false</auto-increment>
<case-sensitive>false</case-sensitive>
<currency>false</currency>
<nullable>1</nullable>
<signed>false</signed>
<searchable>true</searchable>
<column-display-size>512</column-display-size>
<column-label>marker</column-label>
<column-name>marker</column-name>
<schema-name></schema-name>
<column-precision>512</column-precision>
<column-scale>0</column-scale>
<table-name>SPSSCMOR_OBJECTVERSION</table-name>
<catalog-name></catalog-name>
<column-type>12</column-type>
<column-type-name>nvarchar</column-type-name>
<column-definition><column-definition>
<column-index>4</column-index>
<auto-increment>false</auto-increment>
<case-sensitive>false</case-sensitive>
<currency>false</currency>
<nullable>1</nullable>
<signed>false</signed>
<searchable>true</searchable>
<column-display-size>1024</column-display-size>
<column-label>fileVersionLabeled</column-label>
<column-name>fileVersionLabeled</column-name>
<schema-name></schema-name>
<column-precision>1024</column-precision>
<column-scale>0</column-scale>
<table-name></table-name>
<catalog-name></catalog-name>
<column-type>12</column-type>
<column-type-name>nvarchar</column-type-name>
<column-definition><column-definition>
<column-index>5</column-index>
<auto-increment>false</auto-increment>
<case-sensitive>false</case-sensitive>
<currency>false</currency>
<nullable>1</nullable>
<signed>false</signed>
<searchable>true</searchable>
<column-display-size>128</column-display-size>
<column-label>owner</column-label>
<column-name>owner</column-name>
<schema-name></schema-name>
<column-precision>128</column-precision>
<column-scale>0</column-scale>
<table-name>SPSSCMOR_LOCK</table-name>
<catalog-name></catalog-name>
<column-type>12</column-type>
<column-type-name>nvarchar</column-type-name>
<column-definition><column-definition>
<column-index>6</column-index>
<auto-increment>false</auto-increment>
<case-sensitive>false</case-sensitive>
<currency>false</currency>
<nullable>1</nullable>
<signed>false</signed>
<searchable>true</searchable>
<column-display-size>23</column-display-size>
<column-label>creationDate</column-label>
<column-name>creationDate</column-name>
<schema-name></schema-name>
<column-precision>23</column-precision>
<column-scale>3</column-scale>
<table-name>SPSSCMOR_LOCK</table-name>
<catalog-name></catalog-name>
<column-type>93</column-type>
<column-type-name>datetime</column-type-name>
<column-definition><column-definition>
<currentRow><columnValue></columnValue>
<columnValue>/Jobs/Results</columnValue>
<columnValue>0:2007-10-23 12:14:01.71</columnValue>
<columnValue>LATEST, Test</columnValue>
<columnValue>admin</columnValue>
<columnValue>1193409797727</columnValue>
<currentRow><columnValue></columnValue>
<columnValue></columnValue>
<columnValue>/ModelerStreamLibrary/Data
Preparation/P01_AgeCalculations.str</columnValue>
<columnValue>0:2007-10-23 12:09:19.132</columnValue>
<columnValue>LATEST, Production</columnValue>
<columnValue>admin</columnValue>
<columnValue>1193410484000</columnValue>
<currentRow><columnValue></columnValue>
<columnValue></columnValue>
<columnValue>/ModelerStreamLibrary/Data
Preparation/P02_BasketTransformation.str</columnValue>
<columnValue>0:2007-10-23 12:09:22.841</columnValue>
<columnValue>LATEST, Production</columnValue>
<columnValue>admin</columnValue>
<columnValue>1193410529473</columnValue>
<currentRow><columnValue></columnValue>
</RowSetContent>
</getAllLocksResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The getAllVersions operation

Retrieves metadata information for all versions of a specified resource.

The information returned by this operation corresponds to resource versions that are visible for the credentials used in the web service call. Visibility depends on both label security and expiration settings.

- Expired versions are visible only to the resource owner and administrators.
- If the credentials are associated with the *Show All Versions* action or correspond to the owner of the resource, all versions of the file are visible. However, all of the labels may not be visible. Users with these credentials can also see and use the *LATEST* label on the resource.
- If the credentials are not associated with the *Show All Versions* action or do not correspond to the owner of the resource, only labeled resource versions are visible. Users with these credentials can see the *LATEST* version only if they have *Show Latest* action.

Input fields

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

Table 24. Fields for getAllVersions		
Field	Type/Valid Values	Description
uri	string	URI for the resource. Any version information in the URI is ignored.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

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

Table 25. Return Value	
Type	Description
ResourceSpecifier[]	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Java example

The following example returns an array of resource specifiers corresponding to all versions of the file *P01_AgeCalculations.str*. To access the actual resources, use the `getResource` method for the resource specifiers.

```
String uri = "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str";
ResourceSpecifier[] versions = stub.getAllVersions(uri, null);
for (int i = 0; i < versions.length; i++) {
    Resource resourceVersion = versions[i].getResource();
    Version version = resourceVersion.getVersion();
    System.out.println("Version marker: " + version.getMarker());
    System.out.println("Version label: " + version.getLabel());
    Calendar creationValue = resourceVersion.getCreationDate().getValue();
    System.out.println("Creation Date: " + creationValue.toString());
    System.out.println("Created By: " + resourceVersion.getCreatedBy().getValue());
    System.out.println();
}
```

SOAP request example

Client invocation of the `getAllVersions` 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>
    <getAllVersions xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str</uri>
    </getAllVersions>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getAllVersions` 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>
    <getAllVersionsResponse xmlns="http://xml.spss.com/repository/remote">
      <ResourceSpecifier>
        <Resource canWrite="false" canDelete="false" canModifyPermissions="false" xsi:type="File">
          <ResourceID value="0a0a4a359cb71b550000010e75728dd18211"/>
          <Version marker="0:2006-10-23 12:33:07.919">
            <label>Test</label>
          </Version>
          <CreationDate value="2006-10-23T12:33:07.920-05:00"/>
          <ModificationDate value="2006-10-23T12:42:31.240-05:00"/>
          <Title value="P01_AgeCalculations.str"/>
          <Description value="Modeler Stream Library: P1_AgeCalculations.str" language="en"/>
          <ResourcePath value="/ModelerStreamLibrary/Data Preparation/P01_AgeCalculations.str"
            hierarchyType="folder"/>
          <CreatedBy value="admin"/>
          <ObjectCreationDate value="2006-10-23T12:33:07.607-05:00"/>
          <ObjectLastModifiedBy value="admin"/>
          <ObjectLastModifiedDate value="2006-10-23T12:33:08.310-05:00"/>
          <AccessControllist poe="CMOR">
            <AccessControlEntry Permission="READ">
              <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
                Name="$$security/everyoneGroup" IsGroup="true"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="READ">
              <Principal ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
            </AccessControlEntry>
            <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
          </AccessControllist>
          <Author value="admin"/>
          <MimeType value="application/x-vnd.spss-clementine-stream"/>
          <ContentSize value="5003"/>
        </Resource>
      </ResourceSpecifier>
      <ResourceSpecifier>
        <Resource canWrite="false" canDelete="false" canModifyPermissions="false" xsi:type="File">
          <ResourceID value="0a0a4a359cb71b550000010e75728dd18211"/>
          <Version marker="1:2006-10-23 12:34:39.804">
            <label>Production</label>
          </Version>
          <CreationDate value="2006-10-23T12:34:39.803-05:00"/>
          <ModificationDate value="2006-10-23T12:42:32.147-05:00"/>
          <Title value="P01_AgeCalculations.str"/>
          <Description value="Modeler Stream Library: P1_AgeCalculations.str" language="en"/>
          <ResourcePath value="/ModelerStreamLibrary/Data Preparation/P01_AgeCalculations.str"
            hierarchyType="folder"/>
          <CreatedBy value="admin"/>
          <ObjectCreationDate value="2006-10-23T12:33:07.607-05:00"/>
          <ObjectLastModifiedBy value="admin"/>
          <ObjectLastModifiedDate value="2006-10-23T12:33:08.310-05:00"/>
          <AccessControllist poe="CMOR">
            <AccessControlEntry Permission="READ">
              <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
                Name="$$security/everyoneGroup" IsGroup="true"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="READ">
              <Principal ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
            </AccessControlEntry>
            <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
          </AccessControllist>
          <Author value="admin"/>
          <MimeType value="application/x-vnd.spss-clementine-stream"/>
          <ContentSize value="4965"/>
        </Resource>
      </ResourceSpecifier>
    </getAllVersionsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

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

The getBinaryContent operation

This operation is for internal use only.

Input fields

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

Table 26. Fields for getBinaryContent		
Field	Type/Valid Values	Description
uri	string	

The getBulkResourceMetadata operation

Retrieves an array of resources for a given array of uniform resource identifiers. The information returned may be limited to a select set of metadata.

The information returned by this operation corresponds to resource versions that are visible for the credentials used in the web service call. Visibility depends on both label security and expiration settings.

- Expired versions are visible only to the resource owner and administrators.
- If the credentials are associated with the *Show All Versions* action or correspond to the owner of the resource, all versions of the file are visible. However, all of the labels may not be visible. Users with these credentials can also see and use the *LATEST* label on the resource.
- If the credentials are not associated with the *Show All Versions* action or do not correspond to the owner of the resource, only labeled resource versions are visible. Users with these credentials can see the *LATEST* version only if they have *Show Latest* action.

Input fields

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

Table 27. Fields for getBulkResourceMetadata		
Field	Type/Valid Values	Description
uri	string[]	Array of resource URIs.
SelectedMetadata	SelectedMetadata	Provides a way to select specific metadata for retrieval or update
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

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

Table 28. Return Value	
Type	Description
ResourceSpecifier[]	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Java example

The following sample returns an array of `ResourceSpecifier` objects indicating who created each file in the *Data Understanding* folder.

```
String[] uri = {
    "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Understanding/E01_FindDuplicates.str",
    "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Understanding/E02_ExploreDataQuality.str",
    "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Understanding/E03_VisualQuery.str"
};
SelectedMetadata sm = new SelectedMetadata();
sm.setMetadataBase(new CreatedBy());
ResourceSpecifier[] resourceSpec = stub.getBulkResourceMetadata(uri, sm, null);
for (int i = 0; i < resourceSpec.length; i++) {
    Resource resource = resourceSpec[i].getResource();
    System.out.println("Title: " + resource.getTitle().getValue());
    System.out.println("Created by: " + resource.getCreatedBy().getValue());
    System.out.println("");
}
```

SOAP request example

Client invocation of the `getBulkResourceMetadata` 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>
    <getBulkResourceMetadata xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr://pes_server:80/ModelerStreamLibrary/Data%20Understanding/E01_FindDuplicates.str</uri>
      <uri>spsscr://pes_server:80/ModelerStreamLibrary/Data%20Understanding/E02_ExploreDataQuality.str</uri>
      <uri>spsscr://pes_server:80/ModelerStreamLibrary/Data%20Understanding/E03_VisualQuery.str</uri>
      <SelectedMetadata xmlns="http://xml.spss.com/repository">
        <CreatedBy/>
      </SelectedMetadata>
      <ResourceRetrievalOptions xmlns="http://xml.spss.com/repository"/>
    </getBulkResourceMetadata>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getBulkResourceMetadata` 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>
    <getBulkResourceMetadataResponse xmlns="http://xml.spss.com/repository/remote">
      <ResourceSpecifier xmlns="http://xml.spss.com/repository">
        <Resource canWrite="true" canDelete="false" canModifyPermissions="false" xsi:type="File">
          <ResourceID value="0a0a4a35d98ee53f00000010ea9597eda812a"/>
          <Title value="E01_FindDuplicates.str"/>
          <ResourcePath value="/ModelerStreamLibrary/Data Understanding/E01_FindDuplicates.str"
            hierarchyType="folder"/>
          <CreatedBy value="admin"/>
          <AccessControllist poe="CMOR">
            <AccessControlEntry Permission="READ">
              <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
                Name="$$security/everyoneGroup" IsGroup="true"/>
            </AccessControlEntry>
          </AccessControllist>
        </Resource>
      </ResourceSpecifier>
    </getBulkResourceMetadataResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

```

        <AccessControlEntry Permission="READ">
          <Principal ID="//uNative//validUser" DisplayName="validUser" Name="validUser"
            IsGroup="false"/>
        </AccessControlEntry>
        <AccessControlEntry Permission="WRITE">
          <Principal ID="//uNative//validUser" DisplayName="validUser" Name="validUser"
            IsGroup="false"/>
        </AccessControlEntry>
        <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
      </AccessControllist>
    </Resource>
  </ResourceSpecifier>
  <ResourceSpecifier xmlns="http://xml.spss.com/repository">
    <Resource canWrite="true" canDelete="false" canModifyPermissions="false" xsi:type="File">
      <ResourceID value="0a0a4a35d98ee53f0000010ea9597eda80fd"/>
      <Title value="E02_ExploreDataQuality.str"/>
      <ResourcePath
        value="/ModelerStreamLibrary/Data Understanding/E02_ExploreDataQuality.str"
        hierarchyType="folder"/>
      <CreatedBy value="admin"/>
      <AccessControllist poe="CMOR">
        <AccessControlEntry Permission="READ">
          <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
            Name="$$security/everyoneGroup" IsGroup="true"/>
        </AccessControlEntry>
        <AccessControlEntry Permission="READ">
          <Principal ID="//uNative//validUser" DisplayName="validUser" Name="validUser"
            IsGroup="false"/>
        </AccessControlEntry>
        <AccessControlEntry Permission="WRITE">
          <Principal ID="//uNative//validUser" DisplayName="validUser" Name="validUser"
            IsGroup="false"/>
        </AccessControlEntry>
        <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
      </AccessControllist>
    </Resource>
  </ResourceSpecifier>
  <ResourceSpecifier xmlns="http://xml.spss.com/repository">
    <Resource canWrite="true" canDelete="false" canModifyPermissions="false" xsi:type="File">
      <ResourceID value="0a0a4a35d98ee53f0000010ea9597eda810a"/>
      <Title value="E03_VisualQuery.str"/>
      <ResourcePath value="/ModelerStreamLibrary/Data Understanding/E03_VisualQuery.str"
        hierarchyType="folder"/>
      <CreatedBy value="admin"/>
      <AccessControllist poe="CMOR">
        <AccessControlEntry Permission="READ">
          <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
            Name="$$security/everyoneGroup" IsGroup="true"/>
        </AccessControlEntry>
        <AccessControlEntry Permission="READ">
          <Principal ID="//uNative//validUser" DisplayName="validUser" Name="validUser"
            IsGroup="false"/>
        </AccessControlEntry>
        <AccessControlEntry Permission="WRITE">
          <Principal ID="//uNative//validUser" DisplayName="validUser" Name="validUser"
            IsGroup="false"/>
        </AccessControlEntry>
        <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
      </AccessControllist>
    </Resource>
  </ResourceSpecifier>
</getBulkResourceMetadataResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The getChildren operation

Retrieves the immediate children of a specified parent. In addition, the operation allows the client to specify which metadata should be returned. Any metadata not requested that is not part of a collection will have a null value, so it should not attempt to be retrieved without checking for null or monitoring for a `NullPointerException`.

Input fields

The following table lists the input fields for the `getChildren` operation.

Table 29. Fields for <code>getChildren</code>		
Field	Type/Valid Values	Description
parentURI	string	URI for the parent of the children to retrieve.
SelectedMetadata	SelectedMetadata	Provides a way to select specific metadata for retrieval or update

Table 29. Fields for <i>getChildren</i> (continued)		
Field	Type/Valid Values	Description
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

The following table identifies the information returned by the *getChildren* operation.

Table 30. Return Value	
Type	Description
ResourceSpecifier[]	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Java example

The following example retrieves an array of all children of the folder *Data Understanding*, which would consist of both subfolders and files. The only optional metadata returned is the title of the child. It is not necessary to set values for the metadata, only to instantiate the objects. The *ResourceRetrievalOptions* object indicates that expired children should be included in the returned set.

```
String uri="spsscr://pes_server:80/ModelerStreamLibrary/Data%20Understanding";
SelectedMetadata sm = new SelectedMetadata();
sm.setMetadataBase(new Title());
ResourceRetrievalOptions options = new ResourceRetrievalOptions();
options.setShowExpired(true);
ResourceSpecifier[] resources = stub.getChildren(uri, sm, options);
for (int i = 0; i < resources.length; i++) {
    Resource resource = resources[i].getResource();
    System.out.println(resource.getTitle().getValue());
}
```

SOAP request example

Client invocation of the *getChildren* 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>
    <getChildren xmlns="http://xml.spss.com/repository/remote">
      <parentURI>spsscr://pes_server:80/ModelerStreamLibrary/Data%20Understanding</parentURI>
      <SelectedMetadata xmlns="http://xml.spss.com/repository">
        <Title/>
      </SelectedMetadata>
      <ResourceRetrievalOptions showExpired="true" xmlns="http://xml.spss.com/repository"/>
    </getChildren>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getChildren` 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>
    <getChildrenResponse xmlns="http://xml.spss.com/repository/remote">
      <ResourceSpecifier xmlns="http://xml.spss.com/repository">
        <Resource canWrite="false" canDelete="false" canModifyPermissions="false" xsi:type="File">
          <ResourceID value="0a0a4a35d98ee53f0000010ea9597eda812a"/>
          <Title value="E01_FindDuplicates.str"/>
          <ResourcePath value="/ModelerStreamLibrary/Data Understanding/E01_FindDuplicates.str"
            hierarchyType="folder"/>
          <AccessControllist poe="CMOR">
            <AccessControlEntry Permission="READ">
              <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
                Name="$$security/everyoneGroup" IsGroup="true"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="READ">
              <Principal ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
            </AccessControlEntry>
            <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
          </AccessControllist>
        </Resource>
      </ResourceSpecifier>
      <ResourceSpecifier xmlns="http://xml.spss.com/repository">
        <Resource canWrite="false" canDelete="false" canModifyPermissions="false" xsi:type="File">
          <ResourceID value="0a0a4a35d98ee53f0000010ea9597eda810a"/>
          <Title value="E02_ExploreDataQuality.str"/>
          <ResourcePath value="/ModelerStreamLibrary/Data Understanding/E02_ExploreDataQuality.str"
            hierarchyType="folder"/>
          <AccessControllist poe="CMOR">
            <AccessControlEntry Permission="READ">
              <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
                Name="$$security/everyoneGroup" IsGroup="true"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="READ">
              <Principal ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
            </AccessControlEntry>
            <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
          </AccessControllist>
        </Resource>
      </ResourceSpecifier>
      <ResourceSpecifier xmlns="http://xml.spss.com/repository">
        <Resource canWrite="false" canDelete="false" canModifyPermissions="false" xsi:type="File">
          <ResourceID value="0a0a4a35d98ee53f0000010ea9597eda810a"/>
          <Title value="E03_VisualQuery.str"/>
          <ResourcePath value="/ModelerStreamLibrary/Data Understanding/E03_VisualQuery.str"
            hierarchyType="folder"/>
          <AccessControllist poe="CMOR">
            <AccessControlEntry Permission="READ">
              <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
                Name="$$security/everyoneGroup" IsGroup="true"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="READ">
              <Principal ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
            </AccessControlEntry>
            <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
          </AccessControllist>
        </Resource>
      </ResourceSpecifier>
    </getChildrenResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The `getChildrenOptions` operation

Retrieves the immediate children of a specified parent that have a specific label or set of labels applied.

The information returned by this operation corresponds to resource versions that are visible for the credentials used in the web service call. Visibility depends on both label security and expiration settings.

- Expired versions are visible only to the resource owner and administrators.
- If the credentials are associated with the *Show All Versions* action or correspond to the owner of the resource, all versions of the file are visible. However, all of the labels may not be visible. Users with these credentials can also see and use the *LATEST* label on the resource.
- If the credentials are not associated with the *Show All Versions* action or do not correspond to the owner of the resource, only labeled resource versions are visible. Users with these credentials can see the *LATEST* version only if they have *Show Latest* action.

Input fields

The following table lists the input fields for the `getChildrenOptions` operation.

Table 31. Fields for <i>getChildrenOptions</i>		
Field	Type/Valid Values	Description
parentURI	string	URI for the parent of the children to retrieve.
ChildrenSpecification	ChildrenSpecification	Expresses search criteria for querying the repository.

Return information

The following table identifies the information returned by the *getChildrenOptions* operation.

Table 32. Return Value	
Type	Description
ResourceSpecifier[]	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Java example

The following example retrieves an array of all children that have the label *TEST* from the folder *myFolder*.

```

IdentificationSpecifier folderId = getIdentificationSpecifier("/somefolder", HierarchyType.FOLDER);
ChildrenSpecification childSpec = new ChildrenSpecification();

Version version = new Version();
version.addLabel("TEST");
ConditionalTerm term = new ConditionalTerm();
term.setMetadata(version);
//only the EQUAL condition is supported
term.setCondition(QueryCondition.EQUAL);
childSpec.addTerm(term);

SelectedMetadata selectedMetadata = new SelectedMetadata();
childSpec.setSelectedMetadata(selectedMetadata);

ResourceSpecifier[] results = stub.getChildrenOptions(folderId, childSpec);

```

SOAP request example

Client invocation of the *getChildrenOptions* 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>
    <getChildrenOptions xmlns="http://xml.spss.com/repository/remote">
      <parentURI>spsscr://pubslinux:18000/myFolder</parentURI>
      <ChildrenSpecification xmlns="http://xml.spss.com/repository">
        <term condition="equal" xsi:type="ConditionalTerm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <metadata xsi:type="Version" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <label>TEST</label>
          </metadata>
        </term>
        <SelectedMetadata>
          <MetadataBase xsi:type="Version" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
        </SelectedMetadata>
      </ChildrenSpecification>
    </getChildrenOptions>
  </soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `getChildrenOptions` 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>
    <getChildrenOptionsResponse xmlns="http://xml.spss.com/repository/remote">
      <ResourceSpecifier xmlns="http://xml.spss.com/repository">
        <Resource isResourceLockingEnabled="true" canWrite="true" canDelete="true"
          canModifyPermissions="true" isOwner="true" canCreateNewVersion="true"
          isDeleted="false" xsi:type="File">
          <ResourceID value="0923d0b2b83629b3000001318aedb34c3d89"/>
          <Version marker="0:2011-08-29 15:37:27.867" latest="false">
            <label>TEST</label>
          </Version>
          <Title value="quarterly.rptdesign"/>
          <ResourcePath value="/myFolder/quarterly.rptdesign" hierarchyType="folder"/>
          <AccessControllist poe="CMOR">
            <AccessControlEntry Permission="READ">
              <Principal ID="//gNative/$$security/everyoneGroup"
                DisplayName="-- everyone --" Name="$$security/everyoneGroup"
                IsGroup="true"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="WRITE">
              <Principal ID="//gNative/$$security/everyoneGroup"
                DisplayName="-- everyone --" Name="$$security/everyoneGroup"
                IsGroup="true"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="READ">
              <Principal ID="//uNative/admin" DisplayName="admin" Name="admin"
                IsGroup="false"/>
            </AccessControlEntry>
            <owner ID="//uNative/admin" DisplayName="admin" Name="admin"
              IsGroup="false"/>
            </AccessControllist>
          </Resource>
        </ResourceSpecifier>
        <ResourceSpecifier xmlns="http://xml.spss.com/repository">
          <Resource isResourceLockingEnabled="true" canWrite="true" canDelete="true"
            canModifyPermissions="true" isOwner="true" canCreateNewVersion="true"
            isDeleted="false" xsi:type="File">
            <ResourceID value="0923d0b2b83629b3000001318aedb34c3d5a"/>
            <Version marker="1:2011-08-29 15:37:58.031" latest="true">
              <label>TEST</label>
            </Version>
            <Title value="monthly.rptdesign"/>
            <ResourcePath value="/myFolder/monthly.rptdesign" hierarchyType="folder"/>
            <AccessControllist poe="CMOR">
              <AccessControlEntry Permission="READ">
                <Principal ID="//gNative/$$security/everyoneGroup"
                  DisplayName="-- everyone --" Name="$$security/everyoneGroup"
                  IsGroup="true"/>
              </AccessControlEntry>
              <AccessControlEntry Permission="WRITE">
                <Principal ID="//gNative/$$security/everyoneGroup"
                  DisplayName="-- everyone --" Name="$$security/everyoneGroup"
                  IsGroup="true"/>
              </AccessControlEntry>
              <AccessControlEntry Permission="READ">
                <Principal ID="//uNative/admin" DisplayName="admin" Name="admin"
                  IsGroup="false"/>
              </AccessControlEntry>
              <owner ID="//uNative/admin" DisplayName="admin" Name="admin"
                IsGroup="false"/>
              </AccessControllist>
            </Resource>
          </ResourceSpecifier>
        </getChildrenOptionsResponse>
      </soapenv:Body>
    </soapenv:Envelope>
```

The getFault operation

For internal use only. Consumers of the web service should not use this operation.

Input fields

The following table lists the input fields for the `getFault` operation.

Table 33. Fields for getFault		
Field	Type/Valid Values	Description
test	int	

The getFile operation

Retrieving a resource can be achieved in several ways: `getResource`, `getFile`, and `getChildren`. In all cases but `getFile`, only the metadata of the resource is returned. The `getFile` operation returns the metadata as well as an `InputStream` (indirectly) to the content. The specific version of the file, both metadata and content, may be retrieved through the version information included in the URI. If the version information is absent, the most current version will be returned. The version can be specified by either marker or label. If the version does not exist, an exception is thrown.

The file content can be delivered directly (BASE64), or as an attachment (MIME or DIME). To optimize performance, only use BASE64 for small files.

The information returned by this operation corresponds to resource versions that are visible for the credentials used in the web service call. Visibility depends on both label security and expiration settings.

- Expired versions are visible only to the resource owner and administrators.
- If the credentials are associated with the *Show All Versions* action or correspond to the owner of the resource, all versions of the file are visible. However, all of the labels may not be visible. Users with these credentials can also see and use the *LATEST* label on the resource.
- If the credentials are not associated with the *Show All Versions* action or do not correspond to the owner of the resource, only labeled resource versions are visible. Users with these credentials can see the *LATEST* version only if they have *Show Latest* action.

Input fields

The following table lists the input fields for the `getFile` operation.

Table 34. Fields for getFile		
Field	Type/Valid Values	Description
uri	string	URI for a file resource.
DeliveryType	DeliveryType	Defines an enumeration that identifies accepted methods of delivery to retrieve binary content.
SelectedMetadata	SelectedMetadata	Provides a way to select specific metadata for retrieval or update
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

The following table identifies the information returned by the `getFile` operation.

Table 35. Return Value	
Type	Description
ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Java example

The following sample retrieves the file *P01_AgeCalculations.str* from the repository, writing it the folder *myOutput* in the file system.

```
String uri = "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str";
ResourceSpecifier rs = stub.getFile(uri, DeliveryType.MIME, null, null);
File f = (File)rs.getResource();
System.out.println("Title: " + f.getTitle().getValue());
System.out.println("ID: " + f.getResourceID().getValue());
System.out.println("Version marker: " + f.getVersion().getMarker());

Attachment at = f.getBinaryContent().getAttachment();
String href = at.getHref();
org.apache.axis.client.Stub st = (org.apache.axis.client.Stub)stub;
AttachmentPart[] attachments = (AttachmentPart[])st.getAttachments();
if (attachments.length != 0){
    AttachmentPart ap = null;
    boolean bAttFound = false;
    for (int i = 0; i < attachments.length; i++) {
        ap = attachments[i];
        if ( ap.getContentId().equals(href)) {
            bAttFound = true;
            break;
        }
    }
}
if (bAttFound) {
    java.io.File fo = new java.io.File("c:\\myOutput\\P01_AgeCalculations.str");
    FileOutputStream out = new FileOutputStream(fo);
    DataHandler dh = ap.getDataHandler();
    InputStream in = dh.getInputStream();
    byte[] buffer = new byte[256];
    int bytesRead = 0;
    while (true) {
        bytesRead = in.read(buffer);
        if (bytesRead == -1) {
            break;
        }
        out.write(buffer, 0, bytesRead);
    }
    out.close();
    in.close();
}
}
```

SOAP request example

Client invocation of the `getFile` 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>
    <getFile xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str</uri>
      <DeliveryType xmlns="http://xml.spss.com/repository">MIME</DeliveryType>
    </getFile>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getFile` 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>
    <getFileResponse xmlns="http://xml.spss.com/repository/remote">
      <ResourceSpecifier xmlns="http://xml.spss.com/repository">
        <Resource canWrite="false" canDelete="false" canModifyPermissions="false" xsi:type="File">
          <ResourceID value="0a0a4a35d98ee53f0000010ea9597eda80b2"/>
          <Version marker="0:2006-11-02 09:53:33.875"/>
          <CreationDate value="2006-11-02T09:53:33.877-06:00"/>
          <ModificationDate value="2006-11-02T09:53:34.047-06:00"/>
          <Title value="P01_AgeCalculations.str"/>
          <Description>

```

```

value="Modeler Stream Library: P1_AgeCalculations.str -&#x2013; Derive current age and age at joining"
language="en"/>
<ResourcePath value="/ModelerStreamLibrary/Data Preparation/P01_AgeCalculations.str"
  hierarchyType="folder"/>
<CreatedBy value="admin"/>
<ObjectCreationDate value="2006-11-02T09:53:33.780-06:00"/>
<ObjectLastModifiedBy value="admin"/>
<ObjectLastModifiedDate value="2006-11-02T09:53:34.030-06:00"/>
<AccessControlList poe="CMOR">
  <AccessControlEntry Permission="READ">
    <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
      Name="$$security/everyoneGroup" IsGroup="true"/>
  </AccessControlEntry>
  <AccessControlEntry Permission="READ">
    <Principal ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
  </AccessControlEntry>
  <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
</AccessControlList>
<Author value="admin"/>
<MimeType value="application/x-vnd.spss-clementine-stream"/>
<ContentSize value="4969"/>
<BinaryContent>
  <Attachment href="698F46E26D4CBE0DDA6E04CD7328C242"/>
</BinaryContent>
</Resource>
</ResourceSpecifier>
</getFileResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The getResource operation

Returns all metadata describing a resource. In contrast, the `getChildren` operation returns a selected subset of metadata. To obtain content for a resource, use `getFile`.

The metadata for a specific version of a resource may be specified by specifying a version in the URI. If the version information is absent, the most current version will be returned. The version can be specified by either marker or label. If the version does not exist, an exception will be thrown.

The information returned by this operation corresponds to resource versions that are visible for the credentials used in the web service call. Visibility depends on both label security and expiration settings.

- Expired versions are visible only to the resource owner and administrators.
- If the credentials are associated with the *Show All Versions* action or correspond to the owner of the resource, all versions of the file are visible. However, all of the labels may not be visible. Users with these credentials can also see and use the *LATEST* label on the resource.
- If the credentials are not associated with the *Show All Versions* action or do not correspond to the owner of the resource, only labeled resource versions are visible. Users with these credentials can see the *LATEST* version only if they have *Show Latest* action.

Input fields

The following table lists the input fields for the `getResource` operation.

Table 36. Fields for getResource		
Field	Type/Valid Values	Description
uri	string	URI for a resource.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

The following table identifies the information returned by the `getResource` operation.

Table 37. Return Value	
Type	Description
ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Java example

The following sample displays information about the resource at the supplied URL.

```
String uri = "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str";
ResourceSpecifier resourceSpecifier = stub.getResource(uri, null);
Resource resource = resourceSpecifier.getResource();
System.out.println("Title: " + resource.getTitle().getValue());
System.out.println("ID: " + resource.getResourceID().getValue());
System.out.println("Version marker: " + resource.getVersion().getMarker());
```

SOAP request example

Client invocation of the `getResource` 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>
    <getResource xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str</uri>
    </getResource>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getResource` 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>
    <getResourceResponse xmlns="http://xml.spss.com/repository/remote">
      <ResourceSpecifier xmlns="http://xml.spss.com/repository">
        <Resource canWrite="false" canDelete="false" canModifyPermissions="false" xsi:type="File">
          <ResourceID value="0a0a4a35d98ee53f0000010ea9597eda80b2"/>
          <Version marker="0:2006-11-02 09:53:33.875"/>
          <CreationDate value="2006-11-02T09:53:33.877-06:00"/>
          <ModificationDate value="2006-11-02T09:53:34.047-06:00"/>
          <Title value="P01_AgeCalculations.str"/>
          <Description
            value="Modeler Stream Library: P1_AgeCalculations.str -&#x2013; Derive current age and age at joining"
            language="en"/>
          <ResourcePath value="/ModelerStreamLibrary/Data Preparation/P01_AgeCalculations.str"
            hierarchyType="folder"/>
          <CreatedBy value="admin"/>
          <ObjectCreationDate value="2006-11-02T09:53:33.780-06:00"/>
          <ObjectLastModifiedBy value="admin"/>
          <ObjectLastModifiedDate value="2006-11-02T09:53:34.030-06:00"/>
          <AccessControllist poe="CMOR">
            <AccessControlEntry Permission="READ">
              <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
                Name="$$security/everyoneGroup" IsGroup="true"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="READ">
              <Principal ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
            </AccessControlEntry>
            <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
          </AccessControllist>
          <Author value="admin"/>
          <MimeType value="application/x-vnd.spss-clementine-stream"/>
          <ContentSize value="4969"/>
        </Resource>
      </ResourceSpecifier>
    </getResourceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

```

    </Resource>
  </ResourceSpecifier>
</getResourceResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The getResourceSnapshot operation

Retrieves a snapshot of a resource generated as a result of a resource transfer request. Note that this is a blocking call and will hold on to the SOAP connection until the export file is created on the server.

Input fields

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

Table 38. Fields for getResourceSnapshot		
Field	Type/Valid Values	Description
TransferIdentifier	TransferIdentifier	Unique identifier of the resource transfer.
DeliveryType	DeliveryType	Defines an enumeration that identifies accepted methods of delivery to retrieve binary content.

Java example

The following sample retrieves the result of the transfer with identifier *transferId*.

```
stub.getResourceSnapshot(transferId, DeliveryType.MIME);
```

SOAP request example

Client invocation of the getResourceSnapshot 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>Native/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>
    <getResourceSnapshot xmlns="http://xml.spss.com/repository/remote">
      <ns2:TransferIdentifier engineVersion="1.0" xmlns:ns2="http://xml.spss.com/repository">
        >5fhcf631h6</ns2:TransferIdentifier>
      <ns3:DeliveryType xmlns:ns3="http://xml.spss.com/repository">MIME</ns3:DeliveryType>
    </getResourceSnapshot>
  </soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a getResourceSnapshot 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>
    <getResourceSnapshotResponse xmlns="http://xml.spss.com/repository/remote"/>

```

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

The getResourceWithLock operation

Checks if a resource is locked, and if not, creates a lock for the resource, returning the resource to the caller. The lock remains until a call is made to unlock it. An exception is thrown if the resource is already locked.

Input fields

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

Table 39. Fields for getResourceWithLock		
Field	Type/Valid Values	Description
uri	string	URI for a resource.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

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

Table 40. Return Value	
Type	Description
ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

SOAP request example

Client invocation of the getResourceWithLock 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>
    <getResourceWithLock xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr://pes_server:80/Statistics/tree_model.sps</uri>
    </getResourceWithLock>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getResourceWithLock` 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: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>
  </soapenv:Header>
  <soapenv:Body>
    <getResourceWithLockResponse xmlns="http://xml.spss.com/repository/remote">
      <ResourceSpecifier xmlns="http://xml.spss.com/repository">
        <Resource isLockOwner="true" canWrite="true" canDelete="true" canModifyPermissions="true"
          xsi:type="File">
          <ResourceID value="0a0a4a3535482fb800000116a585d52081a3"/>
          <Version marker="0:2007-12-04 15:20:58.669"/>
          <CreationDate value="2007-12-04T15:20:58.670-06:00"/>
          <ModificationDate value="2007-12-04T15:20:58.670-06:00"/>
          <Title value="tree_model.sps"/>
          <ResourcePath value="/Statistics/tree_model.sps" hierarchyType="folder"/>
          <CreatedBy value="admin"/>
          <ObjectCreationDate value="2007-12-04T15:20:58.590-06:00"/>
          <ObjectLastModifiedBy value="admin"/>
          <ObjectLastModifiedDate value="2007-12-04T15:20:59.203-06:00"/>
          <AccessControlList poe="CMOR">
            <AccessControlEntry Permission="READ">
              <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
                Name="$$security/everyoneGroup" IsGroup="true"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="WRITE">
              <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="-- everyone --"
                Name="$$security/everyoneGroup" IsGroup="true"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="READ">
              <Principal ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
            </AccessControlEntry>
            <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
          </AccessControlList>
          <Author value="admin"/>
          <MimeType value="application/x-vnd.spss-spss-syntax"/>
          <ContentSize value="682"/>
        </Resource>
      </ResourceSpecifier>
    </getResourceWithLockResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The `getTransferResults` operation

Retrieves the transfer conflict information for a resource transfer.

Input fields

The following table lists the input fields for the `getTransferResults` operation.

Table 41. Fields for <code>getTransferResults</code>		
Field	Type/Valid Values	Description
TransferIdentifier	TransferIdentifier	Unique identifier of the resource transfer.
TransferResultSelector	TransferResultSelector	Defines a selection criterion for the results generated during transferring activities.

Return information

The following table identifies the information returned by the `getTransferResults` operation.

Table 42. 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

To retrieve the conflict resolution table for a transfer:

1. Create a `TransferResultSelector` object.
2. Create a `TransferConflictCriterion` object.
3. Create a `Filter` object. Define the filter characteristics as needed. An empty filter yields the first page of results.
4. Assign the filter to the criterion using the `setFilter` method.
5. Define the page size, sort column, and sort order for the results using the `setPageSize`, `setSortColumn`, and `setSortOrder` methods for the criterion.
6. Assign the criterion to the selector using the `setPageSelector` method.
7. Provide the `getTransferResults` operation with the transfer identifier and the selector.

The following sample returns the first 10 conflicts for the transfer with the identifier of *transferId*.

```
TransferResultSelector selector = new TransferResultSelector();
TransferConflictCriterion criterion = new TransferConflictCriterion();
Filter filter = new Filter();
criterion.setFilter(filter);
criterion.setPageSize(10);
criterion.setSortColumn("$$repository/conflict_res_source_path");
criterion.setSortOrder(PageSelectorSortOrderType.DESENDING);
selector.setPageSelector(criterion);
PageResult pageResult = stub.getTransferResults(transferId, selector);
```

To return subsequent result pages:

1. Return the navigator for the page result using the `getNavigator` method.
2. Return a `NavigatorItem` object from the navigator using the `getNext` method.
3. Create a `TransferConflictCriterion` object.
4. Create a `PageRequest` object.
5. Define the client key, an internal identifier used to synchronize requests for specific pages, using the `setClientKey` method.
6. Specify the starting row for the results to return by supplying the `setStartingRow` method with a selector obtained from the navigator item.
7. Assign the page request to the criterion using the `setPageRequest` method.
8. Assign the criterion to the results selector using the `setPageSelector` method.
9. Provide the `getTransferResults` operation with the transfer identifier and the selector.

```
NavigatorItem next = pageResult.getNavigator().getNext();
TransferConflictCriterion criterion = new TransferConflictCriterion();
PageRequest pageRequest = new PageRequest();
pageRequest.setClientKey(clientKey);
pageRequest.setStartingRow(next.getSelector());
criterion.setPageRequest(pageRequest);
selector.setPageSelector(criterion);
pageResult = contentRepository.getTransferResults(transferId, selector);
```

To resolve conflicts, use the [“The applyTransferPolicy operation”](#) on [page 15](#) operation.

SOAP request example

Client invocation of the `getTransferResults` 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>
    <getTransferResults xmlns="http://xml.spss.com/repository/remote">
      <ns2:TransferIdentifier engineVersion="2.0" xmlns:ns2="http://xml.spss.com/repository"
        >60317814c6</ns2:TransferIdentifier>
      <ns3:TransferResultSelector xmlns:ns3="http://xml.spss.com/repository">
        <ns3:PageSelector sortColumn="$$repository/conflict_res_source_path" sortOrder="ascending"
          pageSize="9" xsi:type="TransferConflictCriterion">
          <ns3:Filter/>
        </ns3:PageSelector>
      </ns3:TransferResultSelector>
    </getTransferResults>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getTransferResults` 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>
    <getTransferResultsResponse xmlns="http://xml.spss.com/repository/remote">
      <pageResult hitCount="3" pageSize="9" pageNumber="1" clientKey="6031781dg1"
        sortColumn="$$repository/conflict_res_source_path" sortOrder="ascending"
        xmlns="http://www.spss.com/pes/pager">
        <column display="Source Path" fieldName="$$repository/conflict_res_source_path"
          colType="string"/>
        <column display="Source Hierarchy" fieldName="$$repository/conflict_res_source_hierarchy"
          colType="string"/>
        <childColumn display="Conflict" fieldName="$$repository/conflict_res_conflict"
          colType="string"/>
        <childColumn display="Destination Path" fieldName="$$repository/conflict_res_dest_path"
          colType="string"/>
        <childColumn display="Marker" fieldName="$$repository/conflict_res_marker" colType="string"/>
        <childColumn display="Additional Info" fieldName="$$repository/conflict_res_add_info"
          colType="string"/>
        <row rowNumber="1" uri="spsscr:///id=0a0a4a356c24e80b0000011585ab4a0b854e"
          objectID="0a0a4a356c24e80b0000011585ab4a0b854e">
          <cell>
            <value>
              <display xmlns="">Jobs</display>
              <rawString xmlns="">Jobs</rawString>
            </value>
          </cell>
          <cell>
            <value>
              <display xmlns="">Folder</display>
              <rawString xmlns="">folder</rawString>
            </value>
          </cell>
          <childRow rowNumber="1" uri="spsscr:///id=0a0a4a356c24e80b0000011585ab4a0b854e">
            <cell>
              <value>
                <display xmlns="">Duplicate ID</display>
                <rawString xmlns="">$$repository/conflict_duplicate_id</rawString>
              </value>
            </cell>
            <cell>
              <value>
                <display xmlns="">Jobs</display>
                <rawString xmlns="">Jobs</rawString>
              </value>
            </cell>
            <cell>
              <value>
                <display xmlns="">/>
                <rawString xmlns="">/>
              </value>
            </cell>
            <cell>
              <value>
                <display xmlns="">/>
                <rawString xmlns="">/>
              </value>
            </cell>
          </childRow>
        </row>
      </pageResult>
    </getTransferResultsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

```

</row>
<row rowNumber="2" uri="spsscr:///id=0a0a4a356c24e80b0000011585ab4a0b8562"
  objectID="0a0a4a356c24e80b0000011585ab4a0b8562">
  <cell>
    <value>
      <display xmlns="">/Jobs/January</display>
      <rawString xmlns="">/Jobs/January</rawString>
    </value>
  </cell>
  <cell>
    <value>
      <display xmlns="">Folder</display>
      <rawString xmlns="">folder</rawString>
    </value>
  </cell>
  <childRow rowNumber="1" uri="spsscr:///id=0a0a4a356c24e80b0000011585ab4a0b8562">
    <cell>
      <value>
        <display xmlns="">Duplicate ID</display>
        <rawString xmlns="">$$repository/conflict_duplicate_id</rawString>
      </value>
    </cell>
    <cell>
      <value>
        <display xmlns="">/Jobs/January</display>
        <rawString xmlns="">/Jobs/January</rawString>
      </value>
    </cell>
    <cell>
      <value>
        <display xmlns="">/</display>
        <rawString xmlns="">/</rawString>
      </value>
    </cell>
    <cell>
      <value>
        <display xmlns="">/</display>
        <rawString xmlns="">/</rawString>
      </value>
    </cell>
  </childRow>
</row>
<row rowNumber="3" uri="spsscr:///id=0a0a4a356c24e80b0000011585ab4a0b8551"
  objectID="0a0a4a356c24e80b0000011585ab4a0b8551">
  <cell>
    <value>
      <display xmlns="">/Jobs/Results</display>
      <rawString xmlns="">/Jobs/Results</rawString>
    </value>
  </cell>
  <cell>
    <value>
      <display xmlns="">Folder</display>
      <rawString xmlns="">folder</rawString>
    </value>
  </cell>
  <childRow rowNumber="1" uri="spsscr:///id=0a0a4a356c24e80b0000011585ab4a0b8551">
    <cell>
      <value>
        <display xmlns="">Duplicate ID</display>
        <rawString xmlns="">$$repository/conflict_duplicate_id</rawString>
      </value>
    </cell>
    <cell>
      <value>
        <display xmlns="">/Jobs/Results</display>
        <rawString xmlns="">/Jobs/Results</rawString>
      </value>
    </cell>
    <cell>
      <value>
        <display xmlns="">/</display>
        <rawString xmlns="">/</rawString>
      </value>
    </cell>
    <cell>
      <value>
        <display xmlns="">/</display>
        <rawString xmlns="">/</rawString>
      </value>
    </cell>
  </childRow>
</row>
<navigator>
  <page display="1" selector="1" current="true" xmlns="">
</navigator>
</pageResult>
</getTransferResultsResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The getTransferStatus operation

Retrieves status information for a resource transfer.

Input fields

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

Table 43. Fields for *getTransferStatus*

Field	Type/Valid Values	Description
TransferIdentifier	TransferIdentifier	Unique identifier of the resource transfer.

Return information

The following table identifies the information returned by the *getTransferStatus* operation.

Table 44. Return Value

Type	Description
TransferStatus	Provides status and progress information for the resource transfer.

Java example

The following sample uses a `try...catch` statement to report on transfer status. Within the `try` block, the *getTransferStatus* operation returns the current *TransferStatus* object for the transfer corresponding to *transferId*, the identifier returned by the *transferResource* operation. The *getTransferState* method returns the state of the transfer. If the transfer has not completed, `try` block reports the resource currently being transferred.

```
try {
    TransferStatus transferStatus = stub.getTransferStatus(transferId);
    TransferState transferState = transferStatus.getTransferState();
    while (!transferState.equals(TransferState.COMPLETED)) {
        Identifier identifier = transferStatus.getCurrentIdentifier();
        if (identifier != null) {
            System.out.println("Transferring " + identifier.getValue());
        } else {
            System.out.println("Preparing to transfer resource...");
        }
        Thread.sleep(5000);
        transferStatus = stub.getTransferStatus(transferId);
        transferState = transferStatus.getTransferState();
    }
} catch (RepositoryException repositoryException) {
    // handle transfer failure
}
```

SOAP request example

Client invocation of the *getTransferStatus* 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>
    <getTransferStatus xmlns="http://xml.spss.com/repository/remote">
      <ns2:TransferIdentifier engineVersion="2.0"
        xmlns:ns2="http://xml.spss.com/repository">603g8a7ah7</ns2:TransferIdentifier>
    </getTransferStatus>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getTransferStatus` 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>
    <getTransferStatusResponse xmlns="http://xml.spss.com/repository/remote">
      <TransferStatus transferState="running"
        currentProcess="$repository/transfer_export_xmi_process"
        xmlns="http://xml.spss.com/repository">
        <currentIdentifier value="/Jobs/January" hierarchyType="folder" xsi:type="ResourcePath"/>
      </TransferStatus>
    </getTransferStatusResponse>
  </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 45. Return Value	
Type	Description
string	The version 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:Body>
    <getVersion xmlns="http://xml.spss.com/repository/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/repository/remote">
      <version>4.20.000</version>
    </getVersionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The getVersionLabels operation

Returns a list of version labels for a specified resource.

Input fields

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

Table 46. Fields for getVersionLabels		
Field	Type/Valid Values	Description
uri	string	URI for a resource.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

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

Table 47. Return Value	
Type	Description
string[]	Version labels for the resource.

Java example

The following example returns an array of strings containing the labels for the file *P01_AgeCalculations.str*.

```
String url = "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str";
String[] result = stub.getVersionLabels(url, null);
System.out.println("Labels:");
for (int i = 0; i < result.length; i++) {
    System.out.println(result[i]);
}
```

SOAP request example

Client invocation of the getVersionLabels 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>
    <getVersionLabels xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str</uri>
    </getVersionLabels>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getVersionLabels` 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>
    <getVersionLabelsResponse xmlns="http://xml.spss.com/repository/remote">
      <labels>Test</labels>
      <labels>Production</labels>
    </getVersionLabelsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The lockResource operation

Locks a resource, preventing another user from making changes. The resource remains locked until a call is made to unlock the resource. An exception is thrown if an attempt is made to lock a resource that is already locked.

Input fields

The following table lists the input fields for the `lockResource` operation.

Table 48. Fields for lockResource		
Field	Type/Valid Values	Description
targetURI	string	URI for the resource to lock.

Java example

The following sample locks the job *Results* at the specified location.

```
String uri = "spsscr://pes_server:80/Jobs/Results";
stub.lockResource(uri);
```

SOAP request example

Client invocation of the `lockResource` 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>
    <lockResource xmlns="http://xml.spss.com/repository/remote">
      <targetURI>spsscr://localhost:8080/Jobs/Results</targetURI>
    </lockResource>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a lockResource 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>
    <lockResourceResponse xmlns="http://xml.spss.com/repository/remote"/>
  </soapenv:Body>
</soapenv:Envelope>
```

The lockResources operation

Locks multiple resources, preventing another user from making changes. A resource remains locked until a call is made to unlock the resource. An exception is thrown if an attempt is made to lock a resource that is already locked.

Input fields

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

Table 49. Fields for lockResources		
Field	Type/Valid Values	Description
targetURI	string[]	Array of URIs for the resource to lock.

SOAP request example

Client invocation of the lockResources 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>
    <lockResources xmlns="http://xml.spss.com/repository/remote">
      <targetURI>spsscr://localhost:8080/SPV/factor_c_09.spv</targetURI>
      <targetURI>spsscr://localhost:8080/SPV/multiresponse_bod_02.spv</targetURI>
      <targetURI>spsscr://localhost:8080/SPV/regression_c_07.spv</targetURI>
    </lockResources>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a lockResources 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: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>
  </soapenv:Header>
```

```
<soapenv:Body>
  <lockResourcesResponse xmlns="http://xml.spss.com/repository/remote"/>
</soapenv:Body>
</soapenv:Envelope>
```

The moveResource operation

Moves a resource to a specified parent. The resource version can be specified in the resource URI. If there is an attempt to move the resource to a topic or a file parent, an exception is thrown. The object being moved maintains the same object identifier, as well as all other metadata.

Input fields

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

Field	Type/Valid Values	Description
targetParentURI	string	URI for the destination of the move.
sourceURI	string	URI for the resource being moved.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Java example

The following example moves the file *P03_CombineFilters.str* to the folder *Streams*.

```
String tgtParent = "spsscr://pes_server:80/Streams";
String source = "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P03_CombineFilters.str";
stub.moveResource(tgtParent, source, null);
```

SOAP request example

Client invocation of the moveResource 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>
    <moveResource xmlns="http://xml.spss.com/repository/remote">
      <targetParentURI>spsscr://pes_server:80/Streams</targetParentURI>
      <sourceURI>
        spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P03_CombineFilters.str</sourceURI>
      </moveResource>
    </soapenv:Body>
  </soapenv:Envelope>
```

SOAP response example

The server responds to a `moveResource` 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>
    <moveResourceResponse xmlns="http://xml.spss.com/repository/remote"/>
  </soapenv:Body>
</soapenv:Envelope>
```

The query operation

This operation is for internal use only and has been deprecated. Consumers of the web service that need to search for resources should use the Search service instead of this operation.

The query operation provides a way to search for and retrieve resources from the repository. Much like `getChildren`, this operation allows the client to specify which metadata should be returned in the resources. Any metadata not requested that is not part of a collection will have a null value, so it should not attempt to be retrieved without checking for null or monitoring for a `NullPointerException`. The query specification includes the following information:

- **Scope.** The type of objects to be searched. Specify either resource, file, folder, or topic. The search can be restricted to particular file types by specifying a MIME type.
- **Terms.** An ordered list of conditions and predicates used to build the query. The server processes the terms in the order specified.
- **Selected metadata.** Metadata included in the resources that are returned as a result of running the query. To get all metadata, specify null.

By default, the query operation returns the latest version of a resource. To return another version, specify a version in a conditional term. Set either the marker or the label to the version that you want. Alternatively, query can also be used to return all versions of a resource. This can be accomplished by specifying a version in a conditional term and leaving both the marker and label null.

By specifying a version in the selected metadata, the caller will receive version information with each resource that is returned. By also specifying a specific marker in the conditional term, the caller will obtain an array of all labels in the version object in the returned resource. If the caller leaves the marker and label null in the conditional term, a resource for each version (each marker) will be returned. By examining each of these resources, the caller can derive a list of all labels for a given object (all versions of the object).

The information returned by this operation corresponds to resource versions that are visible for the credentials used in the web service call. Visibility depends on both label security and expiration settings.

- Expired versions are visible only to the resource owner and administrators.
- If the credentials are associated with the *Show All Versions* action or correspond to the owner of the resource, all versions of the file are visible. However, all of the labels may not be visible. Users with these credentials can also see and use the *LATEST* label on the resource.
- If the credentials are not associated with the *Show All Versions* action or do not correspond to the owner of the resource, only labeled resource versions are visible. Users with these credentials can see the *LATEST* version only if they have *Show Latest* action.

Input fields

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

Table 51. Fields for query		
Field	Type/Valid Values	Description
QuerySpecification	QuerySpecification	Expresses search criteria for querying the repository.

Return information

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

Table 52. Return Value	
Type	Description
ResourceSpecifier[]	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

The removeLabel operation

Removes a designated label from a specified version of a resource. If the URI for the resource includes a timestamp marker, the label must correspond to that version of the resource. If the URI omits version information, the operation attempts to remove the label from the latest version of the resource.

Input fields

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

Table 53. Fields for removeLabel		
Field	Type/Valid Values	Description
targetURI	string	URI for the resource version.
label	string	The label to be removed.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

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

Table 54. Return Value	
Type	Description
string	URI for the resource.

Java example

The following sample removes the label *Production* for the latest version of the file *P01_AgeCalculations.str*.

```
String uri = "spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str";
```

```
String label = "Production";
String result = stub.removeLabel(uri, label, null);
```

SOAP request example

Client invocation of the `removeLabel` 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>
    <removeLabel xmlns="http://xml.spss.com/repository/remote">
      <targetURI>
        spsscr://pes_server:80/ModelerStreamLibrary/Data%20Preparation/P01_AgeCalculations.str</targetURI>
      <label>Production</label>
    </removeLabel>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `removeLabel` 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>
    <removeLabelResponse xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr:///id=0a0a4a359cb71b550000010e75728dd18211#m.1:2006-10-23%2012:34:39.804</uri>
    </removeLabelResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The runCqlQuery operation

This operation is for internal use only.

Input fields

The following table lists the input fields for the `runCqlQuery` operation.

Table 55. Fields for runCqlQuery		
Field	Type/Valid Values	Description
QueryCqlSpecification	QueryCqlSpecification	Expresses search criteria for querying the repository.

Return information

The following table identifies the information returned by the `runCqlQuery` operation.

Table 56. Return Value	
Type	Description
QueryCqlResult	Result of the CQL query.

The setBulkResourceMetadata operation

Assigns a set of metadata values to a list of specified resources. All specified files or file versions will end up with the same specified metadata.

Input fields

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

Table 57. Fields for setBulkResourceMetadata		
Field	Type/Valid Values	Description
uri	string[]	Array of resource URIs.
ResourceSpecifier	ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.
SelectedMetadata	SelectedMetadata	Provides a way to select specific metadata for retrieval or update
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

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

Table 58. Return Value	
Type	Description
string[]	Array of resource URIs.

Java example

The following sample updates the descriptions for three file resources.

```
String[] uri = {
    "spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M03b_ClusterProfilesTwoStep.str",
    "spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M03a_ClusterProfilesKohonen.str",
    "spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M08_PropensityClustering.str"
};
ResourceSpecifier rs = new ResourceSpecifier();
Resource resource = new Resource();
rs.setResource(resource);
File f = (File)rs.getResource();
Description desc = new Description();
desc.setValue("Clustering Model");
f.setDescription(desc);
SelectedMetadata sm = new SelectedMetadata();
sm.setMetadataBase(new Description());
String[] result = stub.setBulkResourceMetadataResponse(uri, rs, sm, null);
```

SOAP request example

Client invocation of the `setBulkResourceMetadata` 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>
    <setBulkResourceMetadata xmlns="http://xml.spss.com/repository/remote">
      <uri>
        spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M03b_ClusterProfilesTwoStep.str</uri>
      <uri>
        spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M03a_ClusterProfilesKohonen.str</uri>
      <uri>
        spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M08_PropensityClustering.str</uri>
      <ns3:ResourceSpecifier xmlns:ns3="http://xml.spss.com/repository">
        <ns3:Resource xsi:type="File">
          <ns3:Description value="Clustering Model"/>
        </ns3:Resource>
      </ns3:ResourceSpecifier>
      <SelectedMetadata xmlns="http://xml.spss.com/repository">
        <Description/>
      </SelectedMetadata>
    </setBulkResourceMetadata>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `setBulkResourceMetadata` 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>
    <setBulkResourceMetadataResponse xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr:///id=0a0a4a35d98ee53f0000010ea9597eda8403</uri>
      <uri>spsscr:///id=0a0a4a35d98ee53f0000010ea9597eda8392</uri>
      <uri>spsscr:///id=0a0a4a35d98ee53f0000010ea9597eda82e8</uri>
    </setBulkResourceMetadataResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The transferResource operation

Transfers resources between different content repositories.

Input fields

The following table lists the input fields for the `transferResource` operation.

Table 59. Fields for transferResource		
Field	Type/Valid Values	Description
TransferSpecification	TransferSpecification	Defines a specification for transferring resources between two different instances of the content repository

Return information

The following table identifies the information returned by the `transferResource` operation.

Table 60. Return Value	
Type	Description
TransferIdentifier	Unique identifier of the resource transfer.

Java example

The `transferResource` operation provides the ability to both export and import resources. The general process involves the following steps:

1. Initiate the transfer using the `transferResource` operation.
2. Monitor the status of the transfer using the [“The `getTransferStatus` operation” on page 56](#) operation.
3. When exporting, get the export results as an attachment using the [“The `getResourceSnapshot` operation” on page 51](#) operation.
4. Release system resources used for the transfer using the [“The `disposeTransfer` operation” on page 29](#) operation.

The service exports resources in the form of MIME or DIME attachments. To export resources from the repository to an export file:

1. Create a `TransferSpecification` object.
2. For handling of transfer conflicts, set the transfer engine version to `2.0` using the `setEngineVersion` method.
3. Create a `TransferSpecifier` object for the source.
4. Create a `ResourcePath` object. Specify the properties of the path for the resources to export using the `setValue` and `setHierarchyType` methods. Assign the resource path to the source specifier using the `setResourcePath` method.
5. Create an `ExportPolicy` object for the source. Use the `setExternalReferences` method to define whether or not any resources located outside of the selected hierarchy but referenced by the other exported resources should be included in the transfer. Assign the policy to the source specifier using the `addTransferPolicy` method.
6. Assign the source to the transfer specification using the `setTransferSource` method.
7. Create a `TransferSpecifier` object for the target and assign it to the transfer specification using the `setTransferTarget` method.
8. Provide the `transferResource` operation with the transfer specification. The returned `TransferIdentifier` value-object can be used to monitor the current status of the transfer asynchronously.

The following code sample exports the contents of the *Jobs* folder.

```
TransferSpecification transferSpec = new TransferSpecification();
transferSpec.setEngineVersion("2.0");

TransferSpecifier source = new TransferSpecifier();

ResourcePath sourceResourcePath = new ResourcePath();
sourceResourcePath.setValue("/Jobs");
sourceResourcePath.setHierarchyType(HierarchyType.FOLDER);
source.setResourcePath(sourceResourcePath);

ExportPolicy exportPolicy = new ExportPolicy();
exportPolicy.setExternalReferences(true);
source.addTransferPolicy(exportPolicy);

transferSpec.setTransferSource(source);

TransferSpecifier target = new TransferSpecifier();
transferSpec.setTransferTarget(target);

TransferIdentifier transferId = stub.transferResource(transferSpec);
```

Note that exporting large folders in enterprise environments can take hours or days and may require significant system resources.

To import resources from an export file into the repository:

1. Create a `TransferSpecification` object.
2. Create a `TransferSpecifier` object for the source. The resource snapshot of the content being imported is sent as a SOAP attachment so an empty source object specifier is sufficient. Assign it to the transfer specification using the `setTransferSource` method.
3. Create a `TransferSpecifier` object for the target.
4. Create a `ResourcePath` object. Specify an existing folder into which the source resources should be imported using the `setValue` method. Assign the resource path to the target specifier using the `setResourcePath` method.
5. Create an `ImportPolicy` object for the target. Possible policies include `AppendImportPolicy` (add new versions to the resource in the target), `NoChangeImportPolicy` (do not create any new versions if resource already exists in the target), `OverwriteImportPolicy` (delete existing versions in target and replace them with imported versions), and `ConflictResolutionImportPolicy` (generate custom conflict resolution tables for subsequent processing). Assign the policy to the target specifier using the `addTransferPolicy` method.
6. Assign the target to the transfer specification using the `setTransferTarget` method.
7. Add the resource snapshot as a SOAP attachment for the operation request.
8. Provide the `transferResource` operation with the transfer specification. The returned `TransferIdentifier` value-object can be used to monitor the current status of the transfer asynchronously.

The following code sample imports the contents of the *JobsExport.pes* file, generating a conflict resolution table to identify problems.

```
TransferSpecification transferSpec = new TransferSpecification();

TransferSpecifier source = new TransferSpecifier();
transferSpec.setTransferSource(source);

TransferSpecifier target = new TransferSpecifier();
ResourcePath targetResourcePath = new ResourcePath();
targetResourcePath.setValue("/");
target.setResourcePath(targetResourcePath);

ConflictResolutionImportPolicy conflictPolicy = new ConflictResolutionImportPolicy();
ResourcePath policyPath = new ResourcePath();
policyPath.setHierarchyType(HierarchyType.FOLDER);
conflictPolicy.setResourceIdentifier(policyPath);
target.addTransferPolicy(conflictPolicy);

transferSpec.setTransferTarget(target);

FileDataSource fileDataSource = new FileDataSource("c:/temp/JobsExport.pes");
DataHandler dataHandler = new DataHandler(fileDataSource);
AttachmentPart attachmentPart = new org.apache.axis.attachments.AttachmentPart();
attachmentPart.setDataHandler(dataHandler);
((Stub) contentRepository).addAttachment(attachmentPart);

TransferIdentifier transferIdentifier = stub.transferResource(transferSpec);
```

SOAP request example

Client invocation of the `transferResource` 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>
    <transferResource xmlns="http://xml.spss.com/repository/remote">
      <ns2:TransferSpecification engineVersion="2.0" xmlns:ns2="http://xml.spss.com/repository">
        <ns2:TransferSource>
          <ns2:ResourcePath value="/Jobs" hierarchyType="folder"/>
          <ns2:TransferPolicy externalReferences="true" xsi:type="ExportPolicy"/>
        </ns2:TransferSource>
        <ns2:TransferTarget/>
      </ns2:TransferSpecification>
    </transferResource>
  </soapenv:Body>
</soapenv:Envelope>
```

```

</transferResource>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `transferResource` 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>
    <transferResourceResponse xmlns="http://xml.spss.com/repository/remote">
      <TransferIdentifier engineVersion="2.0"
        xmlns="http://xml.spss.com/repository">603g8a7ah7</TransferIdentifier>
    </transferResourceResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

The unlockResource operation

Unlocks a resource. This operation will be ignored if the resource is not locked.

Input fields

The following table lists the input fields for the `unlockResource` operation.

Table 61. Fields for unlockResource		
Field	Type/Valid Values	Description
targetURI	string	URI for the resource to unlock.

Java example

The following sample unlocks the job *Results* at the specified location.

```

String uri = "spsscr://pes_server:80/Jobs/Results";
stub.unlockResource(uri);

```

SOAP request example

Client invocation of the `unlockResource` 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>
    <unlockResource xmlns="http://xml.spss.com/repository/remote">
      <targetURI>spsscr://localhost:8080/Jobs/Results</targetURI>
    </unlockResource>
  </soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `unlockResource` 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>
    <unlockResourceResponse xmlns="http://xml.spss.com/repository/remote"/>
  </soapenv:Body>
</soapenv:Envelope>
```

The unlockResources operation

Unlocks an array of resources. Resources that are not locked will be ignored.

Input fields

The following table lists the input fields for the `unlockResources` operation.

Table 62. Fields for unlockResources		
Field	Type/Valid Values	Description
targetURI	string[]	Array of URIs for the resource to unlock.

Java example

The following sample unlocks the jobs *January* and *February* at the specified locations.

```
String[] uri = {
    "spsscr://pes_server:80/Jobs/January",
    "spsscr://pes_server:80/Jobs/February"
};
stub.unlockResources(uri);
```

SOAP request example

Client invocation of the `unlockResources` 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>
    <unlockResources xmlns="http://xml.spss.com/repository/remote">
      <targetURI>spsscr://localhost:8080/Jobs/January</targetURI>
      <targetURI>spsscr://localhost:8080/Jobs/February</targetURI>
    </unlockResources>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `unlockResources` 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>
  <unlockResourcesResponse xmlns="http://xml.spss.com/repository/remote"/>
</soapenv:Body>
</soapenv:Envelope>

```

The updateCustomProperty operation

Updates the definition of a custom property.

Input fields

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

Table 63. Fields for updateCustomProperty		
Field	Type/Valid Values	Description
CustomProperty	CustomProperty	Metadata describing a custom property defined by a customer to apply to resources.

Return information

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

Table 64. Return Value	
Type	Description
string	The identifier of the updated custom property.

Java example

The following sample uses the getAllCustomProperties operation to retrieve all custom properties in the system. The setLabel method changes the label for the first custom property to Review Complete and the updateCustomProperty operation applies this change to the custom property in the system.

```

CustomProperty[] customProperties = stub.getAllCustomProperties();
customProperties[0].setLabel("Review Complete");
String id = stub.updateCustomProperty(customProperties[0]);

```

SOAP request example

Client invocation of the updateCustomProperty 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">Native/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>
    <updateCustomProperty xmlns="http://xml.spss.com/repository/remote">
      <CustomProperty label="Review Complete" identifier="0a0a4aac00072ffb00000106f3f7b05b348f"
        xmlns="http://xml.spss.com/repository">
        <appliesTo>
          <fileApplicable/>
          <jobApplicable>false</jobApplicable>
          <folderApplicable>false</folderApplicable>
        </appliesTo>
        <constraint>
          <freeform type="boolean"/>
        </constraint>
      </CustomProperty>
    </updateCustomProperty>
  </soapenv:Body>
</soapenv:Envelope>

```

```

        </constraint>
      </CustomProperty>
    </updateCustomProperty>
  </soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `updateCustomProperty` 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>
    <updateCustomPropertyResponse xmlns="http://xml.spss.com/repository/remote">
      <identifier>0a0a4aac00072ffb00000106f3f7b05b348f</identifier>
    </updateCustomPropertyResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

The `updateCustomPropertyValuesInBulk` operation

Updates a list of files or file versions to have the same custom property values.

Input fields

The following table lists the input fields for the `updateCustomPropertyValuesInBulk` operation.

Table 65. Fields for <code>updateCustomPropertyValuesInBulk</code>		
Field	Type/Valid Values	Description
<code>QualifiedCustomPropertyValuesURISpecifier</code>	<code>QualifiedCustomPropertyValuesURISpecifier[]</code>	Wrapper for the resource URI, resource specifier and its custom properties

Return information

The following table identifies the information returned by the `updateCustomPropertyValuesInBulk` operation.

Table 66. Return Value	
Type	Description
<code>string[]</code>	Array of resource URIs.

Java example

The following sample updates the custom property *Approved By* for three files to have the value Langdon Algar.

```

QualifiedCustomPropertyValuesURISpecifier[] qus = new QualifiedCustomPropertyValuesURISpecifier[3];

String uri =
  "spsscr:///id=0a0a4a35d98ee53f0000010ea9597eda8403#m.0:2006-11-02%2009:54:27.282";
ResourceSpecifier rs = new ResourceSpecifier();
File f = new File();
ResourceID rsid = new ResourceID();
rsid.setValue("0a0a4a35d98ee53f0000010ea9597eda8403");
f.setResourceID(rsid);
ResourcePath rp = new ResourcePath();
rp.setValue("/ModelerStreamLibrary/Modeling/M03b_ClusterProfilesTwoStep.str");
rp.setHierarchyType(HierarchyType.folder);
f.setResourcePath(rp);
rs.setResource(f);
CustomPropertyValue cpValue = new CustomPropertyValue();
cpValue.setLabel("Approved By");
cpValue.setIdentifier("0a0a4a3526737cd10000010ec7edde56814d");
SelectionValue[] sv = new SelectionValue[3];
sv[0].setValue("Langdon Algar");
sv[0].setIsSelected(true);
sv[1].setValue("Steve Bennett");
sv[1].setIsSelected(false);

```

```

sv[2].setValue("Lois Sanborn");
sv[2].setIsSelected(false);
cpValue.setSelect(sv);

qus[0].setResourceURI(uri);
qus[0].setResourceSpecifier(rs);
qus[0].setCustomPropertyValue(cpValue);

uri = "spsscr:///?id=0a0a4a35d98ee53f0000010ea9597eda81f6#m.0:2006-11-02%2009:54:16.935";
rsid.setValue("0a0a4a35d98ee53f0000010ea9597eda81f6");
f.setResourceID(rsid);
rp.setValue("/ModelerStreamLibrary/Modeling/M05_ScoringModels.str");
rp.setHierarchyType(HierarchyType.folder);
f.setResourcePath(rp);
rs.setResource(f);

qus[1].setResourceURI(uri);
qus[1].setResourceSpecifier(rs);
qus[1].setCustomPropertyValue(cpValue);

uri = "spsscr:///?id=0a0a4a35d98ee53f0000010ea9597eda8392#m.0:2006-11-02%2009:54:25.594";
rsid.setValue("0a0a4a35d98ee53f0000010ea9597eda8392");
f.setResourceID(rsid);
rp.setValue("/ModelerStreamLibrary/Modeling/M03a_ClusterProfilesKohonen.str");
rp.setHierarchyType(HierarchyType.folder);
f.setResourcePath(rp);
rs.setResource(f);

qus[2].setResourceURI(uri);
qus[2].setResourceSpecifier(rs);
qus[2].setCustomPropertyValue(cpValue);

String[] result = stub.updateCustomPropertyValuesInBulk(qus);

```

SOAP request example

Client invocation of the `updateCustomPropertyValuesInBulk` 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>
    <updateCustomPropertyValuesInBulk xmlns="http://xml.spss.com/repository/remote">
      <QualifiedCustomPropertyValuesURISpecifier xmlns="http://xml.spss.com/repository">
        <ResourceURI>
          spsscr://pes_server:80/?id=0a0a4a35d98ee53f0000010ea9597eda8403#m.0:2006-11-02%2009:54:27.282
        </ResourceURI>
        <ns4:ResourceSpecifier xmlns:ns4="http://xml.spss.com/repository">
          <ns4:Resource xsi:type="File">
            <ns4:ResourceID value="0a0a4a35d98ee53f0000010ea9597eda8403"/>
            <ns4:ResourcePath value="/ModelerStreamLibrary/Modeling/M03b_ClusterProfilesTwoStep.str"
              hierarchyType="folder"/>
            <ns4:associatedTopicList/>
          </ns4:Resource>
        </ns4:ResourceSpecifier>
        <ns3:CustomPropertyValue label="Approved By" identifier="0a0a4a3526737cd10000010ec7edde56814d"
          xmlns:ns3="http://xml.spss.com/repository">
          <ns3:select multipleSelect="false">
            <ns3:selectionValue isSelected="true" value="Langdon Algar"/>
            <ns3:selectionValue isSelected="false" value="Steve Bennett"/>
            <ns3:selectionValue isSelected="false" value="Lois Sanborn"/>
          </ns3:select>
        </ns3:CustomPropertyValue>
      </QualifiedCustomPropertyValuesURISpecifier>
      <QualifiedCustomPropertyValuesURISpecifier xmlns="http://xml.spss.com/repository">
        <ResourceURI>
          spsscr://pes_server:80/?id=0a0a4a35d98ee53f0000010ea9597eda81f6#m.0:2006-11-02%2009:54:16.935
        </ResourceURI>
        <ns4:ResourceSpecifier xmlns:ns4="http://xml.spss.com/repository">
          <ns4:Resource xsi:type="File">
            <ns4:ResourceID value="0a0a4a35d98ee53f0000010ea9597eda81f6"/>
            <ns4:ResourcePath value="/ModelerStreamLibrary/Modeling/M05_ScoringModels.str"
              hierarchyType="folder"/>
            <ns4:associatedTopicList/>
          </ns4:Resource>
        </ns4:ResourceSpecifier>
        <ns3:CustomPropertyValue label="Approved By" identifier="0a0a4a3526737cd10000010ec7edde56814d"
          xmlns:ns3="http://xml.spss.com/repository">
          <ns3:select multipleSelect="false">
            <ns3:selectionValue isSelected="true" value="Langdon Algar"/>
            <ns3:selectionValue isSelected="false" value="Steve Bennett"/>
            <ns3:selectionValue isSelected="false" value="Lois Sanborn"/>
          </ns3:select>
        </ns3:CustomPropertyValue>
      </QualifiedCustomPropertyValuesURISpecifier>
      <QualifiedCustomPropertyValuesURISpecifier xmlns="http://xml.spss.com/repository">
        <ResourceURI>
          spsscr://pes_server:80/?id=0a0a4a35d98ee53f0000010ea9597eda8392#m.0:2006-11-02%2009:54:25.594
        </ResourceURI>

```

```

<ns4:ResourceSpecifier xmlns:ns4="http://xml.spss.com/repository">
  <ns4:Resource xsi:type="File">
    <ns4:ResourceID value="0a0a4a35d98ee53f0000010ea9597eda8392"/>
    <ns4:ResourcePath value="/ModelerStreamLibrary/Modeling/M03a_ClusterProfilesKohonen.str"
      hierarchyType="folder"/>
    <ns4:associatedTopicList/>
  </ns4:Resource>
</ns4:ResourceSpecifier>
<ns3:CustomPropertyValue label="Approved By" identifier="0a0a4a3526737cd10000010ec7edde56814d"
  xmlns:ns3="http://xml.spss.com/repository">
  <ns3:select multipleSelect="false">
    <ns3:selectionValue isSelected="true" value="Langdon Algar"/>
    <ns3:selectionValue isSelected="false" value="Steve Bennett"/>
    <ns3:selectionValue isSelected="false" value="Lois Sanborn"/>
  </ns3:select>
</ns3:CustomPropertyValue>
</QualifiedCustomPropertyValuesURISpecifier>
</updateCustomPropertyValuesInBulk>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `updateCustomPropertyValuesInBulk` 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>
    <updateCustomPropertyValuesInBulkResponse xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr:///id=0a0a4a35d98ee53f0000010ea9597eda8403</uri>
      <uri>spsscr:///id=0a0a4a35d98ee53f0000010ea9597eda81f6</uri>
      <uri>spsscr:///id=0a0a4a35d98ee53f0000010ea9597eda8392</uri>
    </updateCustomPropertyValuesInBulkResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

The updateResource operation

Updates the metadata for an existing resource version. The main difference between `updateResource` and `createResource` is that `update` cannot be used to create an initial or subsequent version of a resource.

Updating a resource is similar to creating a resource. In order to establish some limited concurrency control, when a resource is updated, the metadata for that resource must have been explicitly obtained through the `getResource` or the `getChildren` operations. The modification date of the resource must be explicitly passed as part of the resource when an update request occurs. If the modification date/time of the request does not match the current modification date of the resource in the repository, the update will not be allowed and an exception will be thrown.

It is not necessary that all the metadata for a resource be obtained before the request. For each piece of metadata, there is an instance in the resource. This instance is null unless explicitly set by the client or the server implementation. There is an important distinction between an instance of metadata being null and the value within the instance being null. If the server implementation handling the update comes across a metadata instance that is null, it will ignore it because it has not been set by either the client or a previous server request. This insures that on an update request no metadata that previously had a valid value will be nulled accidentally.

For example, suppose that on a previous request a client had obtained a resource with just the modification date, author, and resource ID explicitly specified. All the other metadata instances in the request will be null. If there was not an author associated with the resource, the instance would not be null but the value of the instance (the value inside the `Author` object) would be null. The client could set a valid author and issue the update request. The only fields that would be modified would be those previously requested.

As another example, suppose `getResource` had been previously issued to obtain the metadata for the resource. A `getResource` call always returns all the metadata associated with the resource, with the exception of binary content, which is not considered metadata. The user changes the value for author and then issues the update request. All of the metadata for the resource would be updated but, with the exception of author, it would just be updated to its existing value.

Input fields

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

Table 67. Fields for updateResource		
Field	Type/Valid Values	Description
targetURI	string	URI for the resource to update.
ResourceSpecifier	ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

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

Table 68. Return Value	
Type	Description
string	URI for the updated resource.

Java example

The following sample updates a file resource by adding a keyword.

```
String url="spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M03b_ClusterProfilesTwoStep.str";
ResourceSpecifier rs = stub.getResource(url);
File f = (File)rs.getResource();
Keyword keyword = new Keyword();
keyword.setValue("clustering");
f.getAssociatedKeywordList().setKeyword(keyword);
String result = stub.updateResource(url, rs, null);
```

SOAP request example

Client invocation of the updateResource 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>
    <updateResource xmlns="http://xml.spss.com/repository/remote">
      <targetURI>
        spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M03b_ClusterProfilesTwoStep.str</targetURI>
      <ns3:ResourceSpecifier xmlns:ns3="http://xml.spss.com/repository">
        <ns3:Resource xsi:type="File">
          <ns3:associatedKeywordList>
            <ns3:keyword value="clustering"/>
          </ns3:associatedKeywordList>
        </ns3:Resource>
      </ns3:ResourceSpecifier>
    </updateResource>
  </soapenv:Body>
</soapenv:Envelope>
```

```

        </ns3:associatedKeywordList>
      </ns3:Resource>
    </ns3:ResourceSpecifier>
  </updateResource>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `updateResource` 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>
    <updateResourceResponse xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr:///id=0a0a4a35d98ee53f0000010ea9597eda8403</uri>
    </updateResourceResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

The updateResources operation

Updates a list of files or file versions with differing metadata sets. Each file or version is updated individually. The results of this operation are identical to calling the `updateResource` operation for each resource to be updated. However, using `updateResources`, only one web service call is needed.

The URI and `ResourceSpecifier` arrays must be of the same size, with the location in the array identifying which update applies to which URI. For example, the first URI corresponds to the first `ResourceSpecifier`.

Input fields

The following table lists the input fields for the `updateResources` operation.

Table 69. Fields for updateResources		
Field	Type/Valid Values	Description
targetURI	string[]	Array of resource URIs.
ResourceSpecifier	ResourceSpecifier[]	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.
ResourceRetrievalOptions	ResourceRetrievalOptions	For use with the Content Repository URI web service. Options on this object can be set to determine how the Resource is retrieved. Usage of this object is optional when using the Content Repository URI service.

Return information

The following table identifies the information returned by the `updateResources` operation.

Table 70. Return Value	
Type	Description
string[]	Array of URIs for the updated resources.

Java example

The following sample updates two file resources by adding a different keyword to each.

```
String[] uri = {
    "spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M03b_ClusterProfilesTwoStep.str",
    "spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M05_ScoringModels.str"
};
ResourceSpecifier[] rs = new ResourceSpecifier[2];
rs[0] = stub.getResource(uri[0]);
File f0 = (File)rs[0].getResource();
Keyword keyword = new Keyword();
keyword.setValue("clustering");
f0.getAssociatedKeywordList().setKeyword(keyword);
rs[1] = stub.getResource(uri[1]);
File f1 = (File)rs[1].getResource();
keyword.setValue("scoring");
f1.getAssociatedKeywordList().setKeyword(keyword);
String[] result = stub.updateResources(uri, rs, null);
```

SOAP request example

Client invocation of the updateResources 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>
    <updateResources xmlns="http://xml.spss.com/repository/remote">
      <targetURI>
        spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M03b_ClusterProfilesTwoStep.str</targetURI>
      <targetURI>
        spsscr://pes_server:80/ModelerStreamLibrary/Modeling/M05_ScoringModels.str</targetURI>
      <ns3:ResourceSpecifier xmlns:ns3="http://xml.spss.com/repository">
        <ns3:Resource xsi:type="File">
          <ns3:associatedKeywordList>
            <ns3:keyword value="clustering"/>
          </ns3:associatedKeywordList>
        </ns3:Resource>
      </ns3:ResourceSpecifier>
      <ns3:ResourceSpecifier xmlns:ns3="http://xml.spss.com/repository">
        <ns3:Resource xsi:type="File">
          <ns3:associatedKeywordList>
            <ns3:keyword value="scoring"/>
          </ns3:associatedKeywordList>
        </ns3:Resource>
      </ns3:ResourceSpecifier>
    </updateResources>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a updateResources 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>
    <updateResourcesResponse xmlns="http://xml.spss.com/repository/remote">
      <uri>spsscr:///id=0a0a4a35d98ee53f0000010ea9597eda8403</uri>
      <uri>spsscr:///id=0a0a4a35d98ee53f0000010ea9597eda81f6</uri>
    </updateResourcesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Chapter 5. JAX-WS clients

Java developers can create custom web service clients by using JAX-WS.

The discussion here assumes the use of Java 6. In general, the process for accessing IBM SPSS Collaboration and Deployment Services web services involves the following steps:

1. Generate a web service client using `wsimport`
2. Package the client
3. Programmatically configure the client
4. Exercise the web service

Generating a JAX-WS client

To generate a JAX-WS client, open a command prompt and execute the Java 6 `wsimport` command.

The `wsimport` command creates JAX-WS service classes and JAXB classes that represent the WSDL schema. For example, the following command executes `wsimport` for the `Scoring.HttpV2` service, storing the output in the current directory:

```
"c:\Program Files\IBM\Java60\bin\wsimport.exe" http://localhost:7001/scoring/services/Scoring.HttpV2?wsdl
```

In this example, the command obtained the WSDL from the server by using the endpoint name followed by `?wsdl`. The `wsimport` command requires access to the WSDL in order to generate the files. JAX-WS also requires access to the WSDL file during runtime, so this example hard codes the value provided to `wsimport` in the Java code. The generated client fetches the WSDL from that same location unless otherwise specified. An alternative is to store the WSDL locally and refer to the local copy rather than downloading the WSDL from the server.

Packaging a JAX-WS client

A JAX-WS client must be packaged as a jar file.

The following example command creates a jar file named `scoring.jar`:

```
"c:\Program Files\IBM\Java60\bin\jar.exe" -cvf scoring.jar *
```

This command assumes the command prompt is in the same location in which the client was generated.

If you store the WSDL locally, include the WSDL and XSD files in the jar file. Place the files in the `\META-INF\wsdl` directory within the file. Refer to that directory programmatically when configuring the client.

Configuring a JAX-WS client

JAX-WS clients can obtain the WSDL file remotely from the server or locally from within the jar file.

The following example demonstrates obtaining the WSDL from the server:

```
com.spss.scoring.ws.jaxws.ScoringServices service =  
    new com.spss.scoring.ws.jaxws.ScoringServices(  
        new URL("http://localhost:7001/scoring/services/Scoring.HttpV2?wsdl"),  
        new QName("http://xml.spss.com/scoring/wsdl", "ScoringServices"));
```

The URL includes the host and port for your server.

The following example demonstrates obtaining the WSDL from the within the jar file:

```
com.spss.scoring.ws.jaxws.ScoringServices service =  
    new com.spss.scoring.ws.jaxws.ScoringServices(  
        new URL("http://localhost:7001/scoring/services/Scoring.HttpV2?wsdl"),  
        new QName("http://xml.spss.com/scoring/wsdl", "ScoringServices"));
```

```
DemoClass.class.getResource("/META-INF/wsdl/scoring.wsdl"),
new QName("http://xml.spss.com/scoring/wsdl", "ScoringServices"));
```

In order to include the required SOAP security headers, create an object that implements `SOAPHandler<SOAPMessageContext>`. See [“SOAPHandler example” on page 80](#) for an example handler object. The following example shows how this object is used:

```
service.setHandlerResolver(new HandlerResolver()
{
    @Override
    public List<Handler> getHandlerChain(PortInfo portInfo)
    {
        List<Handler> handlerChain = new ArrayList<Handler>();
        handlerChain.add(new SecurityHandler("user", "password", "en-US;q=1.0, en;q=0.8"));
        return handlerChain;
    }
});
```

Next, access the service endpoint:

```
ScoringV2 serviceEndpoint = service.getHttpV2();
```

After obtaining the service endpoint, set the JAX-WS standard endpoint address property, which specifies the URL at which to access the endpoint.

```
Map<String, Object> requestContext = ((BindingProvider)serviceEndpoint).getRequestContext();
requestContext.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
"http://localhost:7001/scoring/services/Scoring.HttpV2");
```

SOAPHandler example

JAX-WS clients must include an object that implements `SOAPHandler<SOAPMessageContext>`.

The following code provides an example of this object.

```

/*****
** Licensed Materials - Property of IBM
** IBM SPSS Products: Collaboration and Deployment Services
** © Copyright IBM Corp. 2000, 2013
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
*****/

import java.util.Collections;
import java.util.Set;

import javax.xml.namespace.QName;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPFactory;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.ws.handler.MessageContext;
import javax.xml.ws.handler.soap.SOAPHandler;
import javax.xml.ws.handler.soap.SOAPMessageContext;

/**
 * This is a SOAP handler that applies a security header and a language header to a SOAP message.
 */
public class SecurityHandler implements SOAPHandler<SOAPMessageContext>
{
    // WS-Security header values
    public static final String SECURITY = "Security";
    public static final String USERNAME_TOKEN = "UsernameToken";
    public static final String USERNAME = "Username";
    public static final String PASSWORD = "Password";
    public static final String WS_SECURITY_NAMESPACE =
        "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd";

    // prefixes
    public static final String WSSE_PREFIX = "wsse"; // ws service security
    public static final String SPSS_PREFIX = "spss"; // spss prefix

    // SPSS custom language header values
    public static final String SPSS_HEADER_NAMESPACE = "http://xml.spss.com/ws/headers";
    public static final String CLIENT_ACCEPT_LANGUAGE_HEADER = "client-accept-language";

    private String i_username;
    private String i_password;
    private String i_acceptLanguage;

    /**
     * Creates a security and language handler
     * @param username A user name to access the web service. Cannot be null.
     * @param password A password to access the web service. Cannot be null.
     * @param acceptLanguage The language that should be used by the web service.
     */
}

```

```

    * This value should be formatted according to the HTTP specification regarding
    * the Accept-Language HTTP header (e.g. en-US;q=1.0, en;q=0.8)
    * If the value is null, the language header will not be added.
    */
    public SecurityHandler(String username, String password, String acceptLanguage)
    {
        i_username = username;
        i_password = password;
        i_acceptLanguage = acceptLanguage;
    }

    @Override
    public boolean handleMessage(SOAPMessageContext context)
    {
        // Apply this handler to only outbound traffic
        if((Boolean)context.get(SOAPMessageContext.MESSAGE_OUTBOUND_PROPERTY))
        {
            // get the message
            SOAPMessage message = context.getMessage();
            try
            {
                // get the message header
                SOAPEnvelope envelope = message.getSOAPPart().getEnvelope();
                SOAPHeader header = envelope.getHeader();
                if (header == null)
                {
                    header = envelope.addHeader();
                }

                // add the UsernameToken header
                header.addChildElement(createUsernameTokenSecurityHeader());
                // assuming the language was provided, apply the custom language header
                if(i_acceptLanguage != null)
                {
                    header.addChildElement(createLanguageHeader());
                }
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
        }
        // allow any other handler to execute
        return true;
    }

    /**
    * This method creates a custom language header, which allows the scoring service
    * to use the given language if possible.
    * @return A custom language header
    * @throws Exception
    */
    private SOAPElement createLanguageHeader() throws Exception
    {
        SOAPFactory factory = SOAPFactory.newInstance();

        // create a custom language header
        SOAPElement languageHeader =
            factory.createElement(CLIENT_ACCEPT_LANGUAGE_HEADER, SPSS_PREFIX, SPSS_HEADER_NAMESPACE);

        // include the language text
        languageHeader.addTextNode(i_acceptLanguage);

        return languageHeader;
    }

    /**
    * Creates the WS-Security SOAP header for UsernameToken as SOAPElement.
    *
    * @return the WS-Security SOAP header for UsernameToken
    * @throws Exception as appropriate
    */
    private SOAPElement createUsernameTokenSecurityHeader() throws Exception
    {
        SOAPFactory factory = SOAPFactory.newInstance();

        // create a UsernameToken element
        SOAPElement usernameToken =
            factory.createElement(USERNAME_TOKEN, WSSE_PREFIX, WS_SECURITY_NAMESPACE);

        // add the username element
        SOAPElement usernameElement =
            factory.createElement(USERNAME, WSSE_PREFIX, WS_SECURITY_NAMESPACE);
        usernameElement.addTextNode(i_username);
        usernameToken.addChildElement(usernameElement);

        // add the password element
        SOAPElement passwordElement =
            factory.createElement(PASSWORD, WSSE_PREFIX, WS_SECURITY_NAMESPACE);
        passwordElement.addTextNode(i_password);
        usernameToken.addChildElement(passwordElement);

        // create the Security Header
        SOAPElement securityHeader =
            factory.createElement(SEcurity, WSSE_PREFIX, WS_SECURITY_NAMESPACE);
        securityHeader.addChildElement(usernameToken);

        return securityHeader;
    }

    @Override
    public boolean handleFault(SOAPMessageContext context)
    {
        // allow any other handler to execute
        return true;
    }

    @Override
    public void close(MessageContext context)

```

```
{
    // do nothing
}

@Override
public Set<QName> getHeaders()
{
    return Collections.emptySet();
}
}
```

Exercising web services from JAX-WS clients

Once properly configured, a JAX-WS client can make calls to IBM SPSS Collaboration and Deployment Services web services.

For example, the following code calls the `getConfigurations` operation of the Scoring Service:

```
serviceEndpoint.getConfigurations();
```

Chapter 6. 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. See the topic [“Adding a service reference”](#) on page 83 for more information.
2. Configure the web service endpoint. See the topic [“Configuring the web service endpoint”](#) on page 84 for more information.
3. Programmatically configure the necessary endpoint behaviors. See the topic [“Configuring endpoint behaviors”](#) on page 85 for more information.
4. Exercise the web service. See the topic [“Exercising the service”](#) on page 85 for more information.

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 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.

Important: If you have a .NET client created by using a version of IBM SPSS Collaboration and Deployment Services before 6.0, you must regenerate your service references from the current WSDL files to allow successful communication between your application and the current server. If you do not regenerate your service references, you may experience a variety of errors that may include incorrect namespace assignments, `NullPointerExceptions` in the web services being invoked, and data type assignment errors.

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](#) and [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>
```

```
</client>  
</system.serviceModel>
```

Configuring endpoint behaviors

The following two issues complicate the use of IBM SPSS Collaboration and Deployment Services web 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 referenced DLL 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;
```

Single sign-on authentication

You can use single sign-on authentication for web service calls by obtaining a service ticket that you include in your SOAP requests.

The general process of using single sign-on authentication for WCF clients includes the following steps:

1. Obtain a ticket-grating ticket (TGT) using .NET or WCF code.
2. Send the TGT to the IBM SPSS Collaboration and Deployment Services Repository server using the SSO Authentication Service `getToken` operation to obtain a service ticket. This ensures that single sign-on authentication occurs on the repository server.
3. Send the service ticket in the SOAP header for all subsequent web services calls from your client application.

Chapter 7. Message header reference

The headers for the transport and packaging layers contain vital information for processing a web service call.

For IBM SPSS Collaboration and Deployment Services, the SOAP headers contain the security information under which the web service call is processed. In addition, the HTTP headers contain information about the client that initiated the web service request.

Security headers

Most IBM SPSS Collaboration and Deployment Services web service calls require security information in the request message.

In general, the structure of this content follows the WS-Security extension to the SOAP 1.1 standard. This documentation provides details on the XML elements and attributes that are recognized by IBM SPSS Collaboration and Deployment Services. Some of the elements and attributes are required, some are optional, and some are ignored. Refer to the following official specifications for details, but IBM SPSS Collaboration and Deployment Services requires some special values not referenced in the official specifications.

- <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
- <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-UsernameTokenProfile.pdf>
- <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-KerberosTokenProfile.pdf>

The following table defines the values of namespaces that are used for the SOAP header elements.

Table 71. SOAP header namespaces	
Namespace prefix	Namespace value
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secect-1.0.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
soapenv	http://schemas.xmlsoap.org/soap/envelope/
spsssec	http://xml.spss.com/security

Security element

The `wsse:Security` element is the main security header element included in a `soapenv:Header` element.

Table 72. Attributes of <code>wsse:Security</code>		
Attribute	Description	Example
<code>soapenv:actor</code>	Targets a given endpoint along the message path. This value is ignored.	http://schemas.xmlsoap.org/soap/actor/next
<code>soapenv:mustUnderstand</code>	Clients can specify if the server must process this element. This value is ignored.	0

UsernameToken element

Use the `wsse:UsernameToken` element when a traditional user and password combination is required.

Table 73. Attributes of `wsse:UsernameToken`

Attribute	Description
<code>wsu:Id</code>	An optional string label for the security token. This value is ignored.

Table 74. Child elements of `wsse:UsernameToken`

Attribute	Description	Example
<code>wsse:Username</code>	The xml value represents the identity of the user.	<code>a_user</code>
<code>wsse:Password</code>	The attribute <code>Type</code> specifies the type of password. <code>PasswordText</code> is currently the only supported type. The xml value can handle plain text passwords and encrypted data.	<code>myPassword</code> <code>[{AES}KrY+KLIOYo4O6545tgGsYQ==]</code>
<code>wsse:Nonce</code>	The xml value represents a cryptographically random nonce encoded as base64 data. This is currently ignored.	<code>RUx1ugQo0o3g0Xyl+sUEsA==</code>
<code>wsu:Created</code>	The xml value represents the creation time as a timestamp conforming to <code>wsu:Timestamp</code> . This is currently ignored.	<code>2013-10-08T02:09:20Z</code>

BinarySecurityToken and BinarySecuritySSOToken elements

Binary security tokens may be used when IBM SPSS Collaboration and Deployment Services communicates with itself or when single sign-on (SSO) is used. Customer usage of these token types is limited to SSO.

The `wsse:BinarySecurityToken` and `wsse:BinarySecuritySSOToken` elements have the same format, but only `wsse:BinarySecurityToken` is recognized in the official WS-Security standard. The element `wsse:BinarySecuritySSOToken` was added as a nonstandard element when used in SSO.

Of these two elements, you should use `wsse:BinarySecurityToken` and you must supply the correct attributes for proper handling. The most critical attribute is the `wsu:Id` value which is used during web service request processing to handle the security token correctly.

Table 75. Attributes of `wsse:BinarySecurityToken`

Attribute	Description	Example
<code>ValueType</code>	Indicates the type of the security token. IBM SPSS Collaboration and Deployment Services always writes these values when creating its own XML, but this value is currently ignored during processing. You should use <code>spssec:BinarySecuritySSOToken</code> .	<code>spssec:BinarySecurityToken</code> <code>spssec:BinarySecuritySSOToken</code>

Table 75. Attributes of <code>wsse:BinarySecurityToken</code> (continued)		
Attribute	Description	Example
EncodingType	Indicates the encoding type for the token. The only currently supported type is base64, so this value should always be <code>wsse:Base64Binary</code> . IBM SPSS Collaboration and Deployment Services always writes these values when creating its own XML, but this value is currently ignored during processing.	<code>wsse:Base64Binary</code>
<code>wsu:Id</code>	An identifier for the token. This value must be correctly provided. You should always provide <code>spssSSOToken</code> . The only valid case for using <code>spssToken</code> is for internal web service calls, which use an internal token format.	<code>spssToken</code> <code>spssSSOToken</code>
anyAttribute	An extension mechanism to allow any arbitrary attribute in other namespaces. These extensions are ignored.	

The XML value for `wsse:BinarySecurityToken` and `wsse:BinarySecuritySSOToken` is string data in base64 format.

The client-accept-language element

This element restricts the set of natural languages that are preferred as a response to the request.

This element is inserted into a `soapenv:Header` element and is not related to WS-Security in any way. This is the same value found in the HTTP header named `Accept-Language` as defined in RFC2068. The xml value for this element might look like the following:

```
en-US;q=1.0, en;q=0.8
```

The namespace for this element could be any allowed value, such as `ns1`, which has an associated value of `http://xml.spss.com/ws/headers`.

HTTP headers

In addition to SOAP headers, it is possible to apply HTTP headers as well. None of the HTTP headers is required.

Table 76. HTTP headers	
HTTP header	Description
Accept-Language	The accept language header value, as defined in RFC2068 (e.g. <code>en-US;q=1.0, en;q=0.8</code>). If not supplied the server language setting is used as a default.
CLIENT_ADDR	The client IP address that ultimately initiated the request.
CLIENT_HOSTNAME	The client host name that ultimately initiated the request.

Table 76. HTTP headers (continued)

HTTP header	Description
X-FORWARDED-FOR	The client IP address that ultimately initiated the request. This is standard for determining the originating IP address.

The CLIENT_ADDR, CLIENT_HOSTNAME, and X-FORWARDED-FOR values are useful when a client application makes a call through an HTTP proxy, load balancer, or when IBM SPSS Collaboration and Deployment Services components make internal calls. The CLIENT_ADDR and CLIENT_HOSTNAME entries are specific HTTP headers that can be set by IBM SPSS Collaboration and Deployment Services itself. The X-FORWARDED-FOR header is a standard that some load balancers understand. These headers are used to make a best-effort attempt in determining the originating client for a given call, allowing information to be used for auditing purposes. The headers may not work as intended, but IBM SPSS Collaboration and Deployment Services will fall back to reasonable defaults in those situations.

Appendix A. Deprecated features

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

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

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

Table 77. Features deprecated in previous versions	
Deprecation	Recommended migration action
Security Provider: Active Directory with local override, which supports extended groups and allowed users	Use the standard Active Directory security provider with any necessary groups added
IBM SPSS Collaboration and Deployment Services Enterprise View	Use the Analytic Data View feature
IBM SPSS Collaboration and Deployment Services Enterprise View Driver	Use the Analytic Data View feature
Scenario files	Scenario files (.scn) are no longer supported. Enterprise View source nodes cannot be modified in Deployment Manager. Old scenario files can be modified in IBM SPSS Modeler client and resaved as stream files. Also, scoring configurations that used a scenario file must be deleted and recreated based on a stream file.
Web Install for IBM SPSS Deployment Manager	Use the standalone installer
BIRT Report Designer for IBM SPSS	None
BIRT Report Designer for IBM SPSS viewer	None
IBM SPSS Collaboration and Deployment Services Portlet	Use the IBM SPSS Collaboration and Deployment Services Deployment Portal directly, or use the web services APIs
IBM SPSS Collaboration and Deployment Services Web Part	Use the IBM SPSS Collaboration and Deployment Services Deployment Portal directly, or use the web services APIs
Scoring Service V1 API	Scoring Service V2 API
Scheduling Server Service	None
Reporting Service	None
Authentication Service login operation	Authentication Service doLogin operation
Search Service search operation	Search Service search2.5 operation
SPSS AXIS/Castor web services client jar	Use the tools provided with the Java Runtime Environment, Integrated Development Environment, or Eclipse Web Tools Platform (WTP)

<i>Table 77. Features deprecated in previous versions (continued)</i>	
Deprecation	Recommended migration action
clemrt1_setLogFile() API function	None

Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Privacy policy considerations

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

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

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

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

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be

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

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

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

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

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

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

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

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

Glossary

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

The following cross-references are used in this glossary:

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

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

A

access control list (ACL)

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

ACL

See [access control list](#).

action

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

Active Directory (AD)

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

AD

See [Active Directory](#).

allowed user

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

API

See [application programming interface](#).

appender

A component that receives logging requests from a logger and writes log statements to a specified file or console. See also [logger](#).

application programming interface (API)

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

B

batch file

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

binary large object (BLOB)

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

BLOB

See [binary large object](#).

break group

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

burst report

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

C

cascading permission

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

character large object (CLOB)

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

CLOB

See [character large object](#).

common warehouse metamodel (CWM)

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

content repository

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

context data

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

credential

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

CWM

See [common warehouse metamodel](#).

D

data warehouse

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

distinguished name (DN)

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

DN

See [distinguished name](#).

Document Object Model (DOM)

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

document type definition (DTD)

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

DOM

See [Document Object Model](#).

dormant schedule

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

DTD

See [document type definition](#).

E

EAR

See [enterprise archive](#).

enterprise archive (EAR)

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

execution server

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

export

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

extended group

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

Extensible Markup Language (XML)

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

Extensible Stylesheet Language (XSL)

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

F

field content assist

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

G

general job step

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

I

import

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

iterative consumer reporting job step

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

iterative producer reporting job step

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

J

JAAS

See [Java Authentication and Authorization Service](#).

JAR

See [Java archive](#).

Java archive (JAR)

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

Java Authentication and Authorization Service (JAAS)

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

Java Generic Security Services (JGSS)

A specification that provides Java programs access to the services that include the signing and sealing of messages and a generic authentication mechanism.

Java Naming and Directory Interface (JNDI)

An extension to the Java platform that provides a standard interface for heterogeneous naming and directory services.

JGSS

See [Java Generic Security Services](#).

JNDI

See [Java Naming and Directory Interface](#).

job

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

job step

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

K

KDC

See [key distribution center](#).

Kerberos

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

key distribution center (KDC)

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

keystore

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

L

LDAP

See [Lightweight Directory Access Protocol](#).

Lightweight Directory Access Protocol (LDAP)

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

lock

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

logger

A component that prepares log statements to be written to console or log file. See also [appender](#).

M

message-based schedule

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

metamodel

A model that defines the language for expressing a model.

meta-object

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

meta-object facility (MOF)

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

MIME

See [Multipurpose Internet Mail Extensions](#).

MOF

See [meta-object facility](#).

Multipurpose Internet Mail Extensions (MIME)

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

N

notification

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

O

Object Management Group (OMG)

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

ODS

See [Output Delivery System](#).

OMG

See [Object Management Group](#).

Output Delivery System (ODS)

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

P

package

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

principal

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

R

remote process server

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

repository content adapter

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

repository database

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

resource

A content repository object.

resource definition

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

role

A set of permissions or access rights. See also [action](#).

S

SAX

See [Simple API for XML](#).

schedule

A content repository object that triggers job execution.

scoring configuration

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

security provider

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

service provider interface (SPI)

An API that supports replaceable components and can be implemented or extended by a third party.

SGML

See [Standard Generalized Markup Language](#).

shell script

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

Simple API for XML (SAX)

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

single sign-on (SSO)

An authentication process in which a user can access more than one system or application by entering a single user ID and password.

SOAP

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

SPI

See [service provider interface](#).

SSO

See [single sign-on](#).

Standard Generalized Markup Language (SGML)

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

stop word

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

subscription

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

T

TGT

See [ticket-granting ticket](#).

ticket-granting ticket (TGT)

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

time-based schedule

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

U

Universally Unique Identifier (UUID)

The 128-bit numeric identifier that is used to ensure that two components do not have the same identifier.

UUID

See [Universally Unique Identifier](#).

V

Velocity

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

W

W3C

See [World Wide Web Consortium](#).

WAR

See [web archive](#).

web archive (WAR)

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

Web Services Description Language (WSDL)

An XML-based specification for describing networked services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

World Wide Web Consortium (W3C)

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

WSDL

See [Web Services Description Language](#).

X

XMI

See [XML Metadata Interchange](#).

XML

See [Extensible Markup Language](#).

XML Metadata Interchange (XMI)

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

XSL

See [Extensible Stylesheet Language](#).

Index

Special Characters

.NET framework [83](#)
.NET proxies [5](#)

A

actions [31](#)
app.config files
 WCF clients [84](#)
applyTransferPolicy operation [15](#)

B

BinarySecuritySSOToken element
 in SOAP headers [88](#)
BinarySecurityToken element
 in SOAP headers [88](#)
bindings
 in WSDL files [4](#)
body elements
 in SOAP messages [2](#)

C

cancelTransfer operation [16](#)
cascadePermissions operation [17](#)
children
 in topic hierarchies [10](#)
client-accept-language element
 in SOAP headers [89](#)
Content Repository service
 WCF clients [83](#)
Content Repository URI service
 stubs [7](#)
 WCF clients [83](#)
copyResource operation [19](#)
createCustomProperty operation [20](#)
Created element
 in SOAP headers [88](#)
createOrAddResource operation [22](#)
createResource operation [9](#), [10](#), [23](#)
createResourcePropagate operation [25](#)
createUniqueSubmittedFolder operation [27](#)
custom properties
 assigning values to [12](#)
 defining [12](#)
 updating values [73](#)

D

deleteCustomProperty operation [27](#)
deleteResource operation [28](#)
disposeTransfer operation [29](#)

F

findAllLabels operation [30](#)

G

getActions operation [31](#)
getAllCustomProperties operation [34](#)
getAllLocks operation [35](#)
getAllVersions operation [38](#)
getBinaryContent operation [40](#)
getBulkResourceMetadata operation [40](#)
getChildren operation [42](#)
getChildrenOptions operation [44](#)
getFault operation [46](#)
getFile operation [9](#), [47](#)
getResource operation [9](#), [49](#)
getResourceSnapshot operation [51](#)
getResourceWithLock operation [52](#)
getTransferResults operation [53](#)
getTransferStatus operation [56](#)
getVersion operation [58](#)
getVersionLabels operation [59](#)
glossary [97](#)

H

header elements
 in SOAP messages [2](#), [87](#)
 SOAP security elements [87](#)
Holder classes
 in JAX-WS [5](#)
HTTP [2](#)
HTTP headers
 for SOAP messages [89](#)
HTTPS [2](#)

I

identification specifiers [11](#)

J

Java clients [79](#), [80](#), [82](#)
Java proxies [5](#)
JAX-WS [5](#), [79](#), [80](#), [82](#)

L

labels [59](#), [64](#)
List collections
 in JAX-WS [5](#)
lockResource operation [60](#)
lockResources operation [61](#)

M

MessageBodyMemberAttribute
for WCF clients [85](#)
messages
in WSDL files [4](#)
moveResource operation [62](#)

N

namespaces
for SOAP security elements [87](#)
Nonce element
in SOAP headers [88](#)

P

Password element
in SOAP headers [88](#)
permissions [17](#)
PevServices service
WCF clients [83](#)
port types
in WSDL files [4](#)
Process Management service
WCF clients [83](#)
protocols
in web services [2](#)
proxies
.NET [5](#)
Java [5](#)

Q

query operation [63](#)

R

removeLabel operation [64](#)
resource specifiers [11](#)
resources
actions [31](#)
children [42](#)
copying [19](#)
creating [23](#), [25](#)
deleting [28](#)
identifiers [40](#)
labels [59](#), [64](#)
moving [62](#)
retrieving [47](#), [49](#)
topics [10](#)
updating [75](#), [77](#)
versions [38](#)
runCqlQuery operation [65](#)

S

Scoring service
WCF clients [83](#)
Security element
in SOAP headers [87](#)
services
in WSDL files [5](#)

setBulkResourceMetadata operation [66](#)
single sign-on
for WCF clients [86](#)
WCF clients [83](#)
SOAP [2](#)
SOAPHandler [80](#)
specifiers [11](#)
SSO, *See* single sign-on
stubs
Content Repository URI service [7](#)

T

transferResource operation [67](#)
types
in WSDL files [3](#)

U

unlockResource operation [70](#)
unlockResources operation [71](#)
updateCustomProperty operation [72](#)
updateCustomPropertyValuesInBulk operation [73](#)
updateResource operation [10](#), [75](#)
updateResources operation [77](#)
URIs [11](#)
Username element
in SOAP headers [88](#)
UsernameToken element
in SOAP headers [88](#)

V

versions [38](#)
Visual Studio [83](#)

W

WCF clients
endpoint behaviors [85](#)
endpoint configuration [84](#)
limitations [83](#)
service reference [83](#)
single sign-on [83](#)
web services
introduction to web services [1](#)
protocol stack [2](#)
system architecture [1](#)
what are web services? [1](#)
web.config files
WCF clients [84](#)
Windows Communication Foundation [83](#)
WSDL files
bindings [4](#)
messages [4](#)
port types [4](#)
services [5](#)
types [3](#)
wsdl.exe [5](#)
wsdl2java [5](#)
wsimport [5](#), [79](#)



XmlElementAttribute
for WCF clients [85](#)

