IBM SPSS Collaboration and Deployment Services 5 Content Repository Service Developer's Guide

Note: Before using this information and the product it supports, read the general information under Notices on p. 182.

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

Preface

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

About IBM Business Analytics

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

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

Technical support

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

Contents

Appendices

Chapter

1

Introduction to web services

What are web services?

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

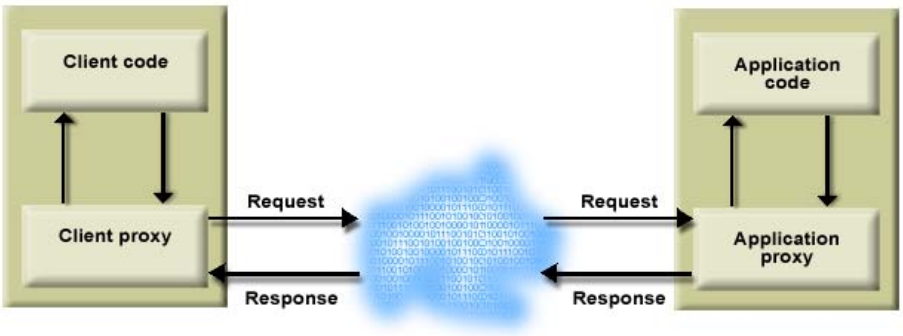
Advantages to implementing application functionality as web services include the following:

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

Web service system architecture

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

Figure 1-1
Web service architecture



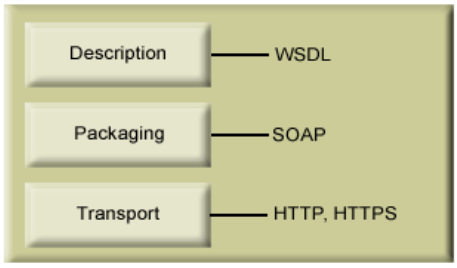
The client code supplies input to an operation offered by a proxy class. The proxy class generates a request containing a standardized representation of the input and sends it across the network to the application. A proxy class on the server receives the request and unmarshals the contents into objects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

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

Web service protocol stack

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

Figure 1-2
Web service protocol stack



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

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

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

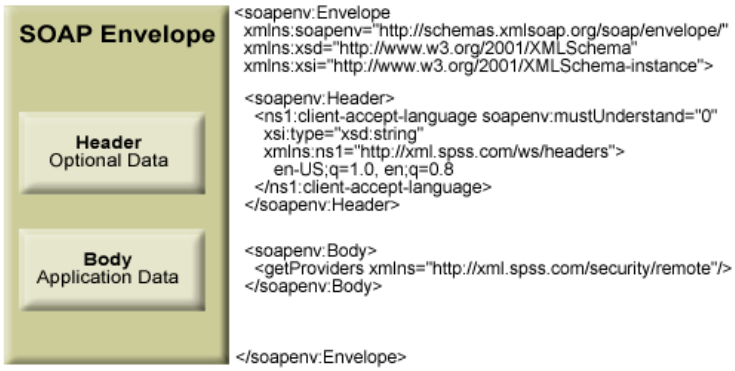
Simple Object Access Protocol

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

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

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

Figure 1-3
An example SOAP packet



Web Service Description Language

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

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

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

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

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

Types

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

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

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

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

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

Messages

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

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

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

Port types

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

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

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

Bindings

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

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

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

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

Services

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

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

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

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

Proxies

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

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

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

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

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

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

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

Chapter 2

Content Repository 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 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. To retrieve the object, a query can be defined to search for objects having the specified name and keywords. Objects returned by the query can be retrieved for subsequent use. Retrieval of the object also returns the meta-data associated with it.

Accessing the Content Repository Service

To access the functionality offered by the Content Repository 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>/cr-ws/services/ContentRepository
```

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

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

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

Calling Content Repository 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/ContentRepository";
URL url = new URL("http", "cads_server", 80, context);
ContentRepositoryService service = new ContentRepositoryServiceLocator();
stub = service.getContentRepository(url);
```

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

```
stub.findAllLabels();
```

Some operations may report errors for client applications using JAX-WS for proxy generation. These errors result from an incompatibility between the server implementation and the proxy code generated by the tool. For example, operations that accept an array of TargetIdentificationSpecifier objects as inputs, such as lockResources and updateResourcesSpecific, return errors due to the proxy code omitting the correct instance type of IdentificationSpecifier. Fortunately, there are often alternative operations offering similar functionality that can be used when such errors occur.

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 meta-data 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 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 Service involves use of the createResource operation to define folders. Create a child for a folder by specifying the desired 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 of the Content Repository Service to retrieve a file from the repository. If the file meta-data only is needed, use the getResource operation.

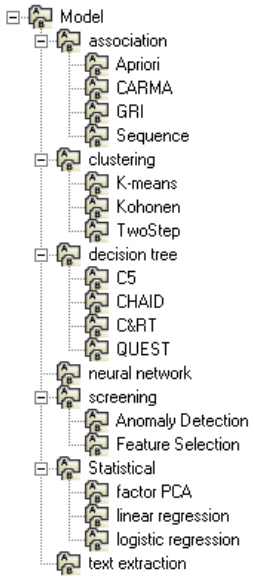
Topics

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 topic hierarchy shown in the “Example topic hierarchy” figure. This hierarchy uses model type as the basis for classifying resources.

Figure 3-1
Example topic hierarchy



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 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.

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 rs;
}
```

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;
}
```

Label security

Label security provides control over which principals can view or modify user-defined version labels in the system, allowing the specification of which users can move or delete a label. Although users may be able to view a version of a resource in the IBM® SPSS® Collaboration and Deployment Services Repository, only those users with permissions for the labels associated with the version can access the labels. Moreover, label security applies to any version of any resource using the label. For example, to control who can assign production versions of resources, an administrator may restrict access to the *Production* label to prevent other users from moving that label from one version to a newer version. The security defined for the *Production* label applies to every version of every resource using that label.

A label is secured by associating it with an access control list (ACL). The ACL defines the permission level for principals in the system. Principals fall into one of two categories:

- A **user** is an individual who needs access to the label.
- A **group** is a set of users who need access to the label.

Each principal in the system is characterized by the following attributes:

- **ID.** A unique identifier of the principal. The ID may be useful for debugging purposes, but should generally not be shown to users.
- **Type.** An indicator of whether the principal is a user or group.
- **Display name.** A name for the principal suitable for display by a client application. This name may include the provider name and domain for some system configurations.

Valid permissions for principals include:

- **Modify.** Principals with this permission can apply, move, and delete the label, as well as delete the version of a resource having the label. If a user does not have this permission for a label, that user cannot delete a resource or version of a resource to which the label is applied.
- **Read.** Any principal with this permission can view the version of a resource associated with the label by referencing the label.

The Content Repository Service includes operations for retrieving and updating the security for existing labels.

Custom properties

The standard meta-data 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 meta-data 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 Content Repository Service 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 by 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 Content Repository Service 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 are 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 Content Repository Service includes operations for initiating, monitoring, and canceling resource transfers.

Promotion

Promotion provides a policy-based approach to transferring individual objects between IBM® SPSS® Collaboration and Deployment Services Repository instances. A promotion request requires the specification of the following two items:

- an object to promote
- a promotion policy governing the promotion process

A promotion policy identifies the rules used for a particular promotion. The policy determines which related repository objects, if any, are promoted along with the specified object. For example, when promoting a job, you often want to include any files referenced by the job steps. In addition, you may to include execution server and credential definitions. The policy dictates how these items are handled.

Promotion is typically used in conjunction with label event notifications. For example, an analyst can set up the *Production* label to notify an administrator when the label is applied to a job so that the job can subsequently be promoted to production status.

To submit a promotion request, the role for the user must include the *Promote Objects* action. In addition, the creation of new items in the target server may require that the role associated with the target credentials include certain actions. For more information, see the topic Promotion considerations on p. 17.

Use the promoteResource operation to promote objects.

Promotion Policies

A promotion policy is a resource definition that specifies rules and properties to apply when promoting objects. Use of a promotion policy prevents having to redefine the set of rules each time an object is promoted. Instead, you apply the policy to the promotion request, and the policy rules are automatically enforced. A promotion policy is defined once, but can be applied to any number of promotion requests. Information specified in a promotion policy includes the following:

- promotion timing
- resource definition inclusion
- related resource exclusion

Promotion timing determines when the promoted object becomes available in a new IBM® SPSS® Collaboration and Deployment Services Repository. In immediate promotion, the promoted object is added to the new repository server automatically as part of the promotion request. In this case, the promotion policy must specify the target repository server as well as valid credentials for accessing that server to enable the source and target repository servers to communicate with each other. In contrast, delayed promotion saves the promoted object to a specified file on the file system for subsequent transfer at a later time. In this case, the promotion policy includes no information about the target repository server. The promotion is completed by manually importing the resulting file into the target server.

The resource definition portion of a promotion policy specifies how to handle the resource definitions referenced by a promoted object. For example, a job may include steps that refer to server and credential definitions. Some steps may rely on data source definitions or on a version of the Enterprise View. If these definitions do not exist in the target server, the promoted item will not function properly without manual intervention to redefine these properties. By including the necessary resource definitions in the promotion, the promoted item should function in the target server exactly as it does in the source server without any redefinition. However, if you are sure the necessary resource definitions exist in the target server, the definitions from the source server can be omitted from the promotion.

If a object being promoted uses other IBM SPSS Collaboration and Deployment Services Repository items, the promotion policy should include information about which of those items should be excluded from the promotion. For example, when promoting IBM® Analytical Decision Management objects, it may be necessary to include a version of an associated rule file for a IBM® SPSS® Modeler stream but exclude the Data Provider Definition version used. The promotion policy specifies the items to exclude by their MIME types. In the previous example, the policy would specify *application/x-vnd.spss-data-provider* as a MIME type to exclude from the promotion.

To create or modify a promotion policy, the role for the user must include the *Define Promotion Policies* action. In addition, the user must have write permissions to the *Promotion Policies* subfolder of the *Resource Definitions* folder.

Use the createResource operation to create promotion policies.

Promotion considerations

For an object, resource definitions, and referenced files to all be promoted successfully, the role for the target server credentials must include all relevant actions for the items included in the promotion request. The "Actions affecting promotion" table identifies the actions needed for a variety of items that may be included in the promotion set.

Table 3-1
Actions affecting promotion

Included in promotion set	Action needed for target credentials
Resource having notifications	Define and Manage Notifications
Credentials	Define Credentials
Resource having custom properties	Define Custom Properties
Data source	Define Datasources
Message domain	Define Message Domains
Server cluster definition	Define Server Clusters
Server definition	Define Servers
Resource having topics	Define Topics
Job	Job Edit
Enterprise View, Application View, or Data Provider Definition	Manage Enterprise View
Resource having subscriptions for a different principal	Create Subscriptions

Included in promotion set	Action needed for target credentials
Resource having subscriptions for the target principal	Manage Subscriptions
Job having a schedule	Schedules
Resource version that is not the latest version in the target server	Show All Versions
Latest resource version	Show LATEST

In addition, the target credentials must have the *Manage label* permission for any label associated with a promoted object version.

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 4-1
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 desired 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.

The cancelTransfer operation

Cancels a resource transfer.

Input fields

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

Table 4-2
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 identifer *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 4-3
Fields for cascadePermissions

Field	Type/Valid Values	Description
ParentIdentificationSpecifier	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.
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.

Java example

The following example cascades the change in permissions, beginning with the retrieved folder. The Options parameter in the operation allows the client to cascade only the delta as well as set the new permission in the access control list on the parent. These are the default settings, so the options are not explicitly changed. This method will set the new access control list for the folder and cascade only the permission added. In this case, the principal MJM gets WRITE permission.

```
ResourcePath rp = new ResourcePath();
rp.setValue("/" + getFolderName());
ResourceSpecifier rs = stub.getResource(getIDSpecifier(rp, null));
Folder f = (Folder)rs.getResource();
AccessControlList acl = f.getAccessControlList();
AccessControlEntry ace = new AccessControlEntry();
ace.setPrincipal("MJM");
ace.setPermission(com.spss.security.access.ObjectPermission.WRITE.getID());
acl.addAccessControlEntry(ace);
stub.cascadePermissions(
  getIDSpecifier(f.getResourceID(), null), new Options(), acl);
```

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

Input fields

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

Table 4-4
Fields for copyResource

Field	Type/Valid Values	Description
TargetParent	TargetParent	Specifies a parent target for a move or copy operation.
Source	Source	Specifies the source for a move or copy operation.

Return information

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

Table 4-5
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.

Java example

The following sample copies an array of resources to a designated folder. The setIdentifier method sets the identifier for the TargetParent object to the identifier for the desired target folder. The function then parses the resource array, creating a new Source object for each resource in the array.

```
public void copyResources(AdaptableResource[] resources, AdaptableFolder targetFolder)
  throws IOException, ServiceException {
  ContentRepository repository = getContentRepository();
  TargetParent tgtParent = new TargetParent();
  tgtParent.setIdentifier(targetFolder.getResourceID());
  for (int i = 0; i < resources.length; i++) {
    AdaptableResource resource = resources[i];
    Source source = new Source();
    source.setIdentifier(resource.getResourceID());
    IdentificationSpecifier objSpecification =
      repository.copyResource(tgtParent, source);
```

```
    targetFolder.setHasChildren(true);
  }
}
```

The setIdentifier method sets the Source identifier to the identifier for the resource being copied. The copyResource operation then copies the Source object to the desired target, returning the identifier for the copied object. The function sets the HasChildren property for the target folder to *true* to reflect the presence of the newly copied resources.

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">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>
  <copyResource xmlns="http://xml.spss.com/repository/remote">
   <TargetParent xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106561fccf88047" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </TargetParent>
   <Source xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106561fccf88177" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </Source>
  </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">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106a7c0f12d8116" xsi:type="ResourceID"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <Version marker="0:2005-09-30 14:50:59.086"/>
   </IdentificationSpecifier>
  </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 4-6
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 4-7
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(CustomPropertyValueType.BOOLEAN);
cnst.setFreeform(ff);
cp.setConstraint(cnst);

ContentRepository repository = getContentRepository();
String id = repository.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, respectively. The setFileApplicable method, on the other hand, 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 desired type for the property using setType. Use setFreeform to assign the type to the Constraint object.

After assiging 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/>
     <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>0a0a4aac00072ffb00000106f3f7b05b348f</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 4-8
Fields for createOrAddResource

Field	Type/Valid Values	Description
ParentIdentificationSpecifier	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.

Field	Type/Valid Values	Description
ResourceSpecifier	ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.
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 4-9
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 createResource operation

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

Input fields

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

Table 4-10
Fields for createResource

Field	Type/Valid Values	Description
ParentIdentificationSpecifier	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.
ResourceSpecifier	ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Return information

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

Table 4-11
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.

Java example

Creating a resource typically involves the following steps:

1. Create a ResourcePath object and set its value to the repository path for the object 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 creates a folder in the repository.

```
ResourcePath rp = new ResourcePath();
rp.setValue("/");
Folder folder = new Folder();
Title title = new Title();
title.setValue(“myfolder”);
folder.setTitle(title);
stub.createResource(getIDSpecifier(rp, null), getResourceSpecifier( folder));
```

The following code creates a new topic in the repository. Topics are similar to folders, but should not be considered parents of a file. Topics from multiple hierarchies can be associated with a file.

```
ResourcePath rp = new ResourcePath();
rp.setValue("/");
Topic topic = new Topic();
Title title = new Title();
title.setValue(getTopicName());
topic.setTitle(title);
stub.createResource(getIDSpecifier(rp, null), getResourceSpecifier(topic));
```

The following code adds a file to the repository. The content of the file can be transmitted as part of the SOAP message or as an attachment. The former should only be used for small files. The example here uses an attachment.

```
ResourcePath rp = new ResourcePath();
rp.setValue("/" + getFolderName() + "/");
```

```
File f = new File();
Title t = new Title();
t.setValue("myNewFile.xls");
f.setTitle(t);
MimeType mt = new MimeType ();
mt.setValue("application/msexcel");
f.setMimeType(mt);
Description english = new Description();
english.setValue("This is my file");
f.addDescription(english);
Author a = new Author();
a.setValue("PES User");
f.setAuthor(a);
URL("file:///c:/myFile.xls");
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);
stub.createResource(getIDSpecifier(rp, null), getResourceSpecifier(f));
```

The following code adds a new version of a file that already exists in the repository. Notice that we use the getResource operation to retrieve only the metadata for the existing file, avoiding the overhead of sending the file content to the client.

```
ResourcePath rp = new ResourcePath();
rp.setValue("/" + getFolderName() + "/myNewFile.xls" );
File f = (File)stub.getResource(rp, null);
Version v = new Version();
v.addLabel("PRODUCTION");
v.addLabel("GOLDEN CODE");
v.setMarker("Version 2");
f.setVersion(v);
Keyword key = new Keyword();
key.setValue("April");
f.addKeyword(key);
URL url = new URL("file:///c:/newFile.xls");
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);
storedFileID = stub.createResource(getIDSpecifier(rp,null),getResourceSpecifier( f));
```

To create a promotion policy, perform the following steps:

1. Create an IdentificationSpecifier object for the *Promotion Policies* folder in the repository. Set the identifier value for this specifier object to the value for the policies folder.

```
IdentificationSpecifier parentIdSpec = new IdentificationSpecifier();
parentIdSpec.setIdentifier(new ResourceID());
parentIdSpec.getIdentifier().setValue("000000000000000000000000000000000e7");
```

2. Create a ResourceSpecifier object for the promotion policy.
3. Create a File object. Define title, author, and MIME type metadata values for the promotion policy. Be sure the MIME type is specified as *application/x-vnd.spss-promotion-policy*.
4. Assign the file object to the resource specifier using the setResource method.

```
ResourceSpecifier resSpec = new ResourceSpecifier();
File file = new File();
resSpec.setResource(file);
file.setTitle(new Title());
file.getTitle().setValue("My Promotion Policy");
file.setAuthor(new Author());
file.getAuthor().setValue("admin");
file.setMimeType(new MimeType());
file.getMimeType().setValue("application/x-vnd.spss-promotion-policy");
```

5. Create a PromotionSpecification object for the promotion policy specification.
6. Create two TransferSpecifier objects, one for the source being promoted and one for the target of the promotion.
7. For the target specifier, supply the setEndpoint method with the identifier for the server definition corresponding to the target repository server.
8. For the target specifier, supply the setCredentials method with the identifier for the credentials definition corresponding to valid credentials for connecting to the target repository server.
9. Create a ResourcePath object for the location within the target server in which to promote the item. Use the setValue method to specify the root folder within the target. Assign the path object to the target specifier by using the setResourcePath method.

```
final PromotionSpecification proSpecification = new PromotionSpecification();
TransferSpecifier sourceTransferSpec = new TransferSpecifier();
TransferSpecifier targetTransferSpec = new TransferSpecifier();
targetTransferSpec.setEndpoint("spsscr:///?id=09412e7d9120c69a0000012d62846bd3803b");
targetTransferSpec.setCredentials("spsscr:///?id=09412e7d9120c69a0000012d62846bd3804a");
ResourcePath targetResPath = new ResourcePath();
```

```
targetResPath.setValue("/");
targetTransferSpec.setResourcePath(targetResPath);
```

10. Define the rules governing the import of items for the target server. Use an AppendImportPolicy object for the rules applied for adding versions to existing target objects.
11. Supply the setInvalidVersions method with a Boolean indicating whether or not invalid versions from the source should be promoted. Supply the setHonorResourceLocks method with a Boolean indicating whether or not to honor the locks on any locked objects. Supply the setVersionLabelPolicy method with a value indicating whether labels from the source server or the target server should be used.
12. Assign the policy object to the target specifier by using the addTransferPolicy method.

```
AppendImportPolicy appendImportPolicy = new AppendImportPolicy();
appendImportPolicy.setInvalidVersions(true);
appendImportPolicy.setHonorResourceLocks(false);
appendImportPolicy.setVersionLabelPolicy(VersionLabelPolicy.SOURCE);
targetTransferSpec.addTransferPolicy(appendImportPolicy);
```

13. Define the rules governing the exclusion of resource definitions for the promotion. Create an ExcludeTransferPolicy object for the rules.
14. Create a ResourcePath object for each hierarchy to exclude. Supply the setHierarchyType method with a value indicating the hierarchy type. Use the addResourceIdentifier method to add the path object to the policy.
15. Assign the policy object to the target specifier by using the addTransferPolicy method.

```
ExcludeTransferPolicy excludeTransferPolicy = new ExcludeTransferPolicy();
ResourcePath excludedHierarchy = new ResourcePath();
excludedHierarchy.setHierarchyType(HierarchyType.TOPIC);
excludeTransferPolicy.addResourceIdentifier(excludedHierarchy);
excludedHierarchy = new ResourcePath();
excludedHierarchy.setHierarchyType(HierarchyType.CUSTOMPROPERTY);
excludeTransferPolicy.addResourceIdentifier(excludedHierarchy);
targetTransferSpec.addTransferPolicy(excludeTransferPolicy);
```

16. Define the rules governing the exclusion of items used by the promoted item. Create an ExcludeDependentTransferPolicy object for the rules.
17. Create an ExcludeTransferSpecification object to specify the excluded items.
18. Create a SelectedMetadata object to specify the metadata corresponding to the excluded items.
19. Create a MimeType object to specify the MIME type for the excluded type. Supply the setValue method with the MIME type. Use the addMetadataBase method to add the MIME type object to the metadata object.

20. Assign the metadata object to the transfer specification for the exclusion by using the setSelectedMetadata method.
21. Assign the transfer specification to the policy by using the setExcludeTransferSpecification method.
22. Assign the policy object to the target specifier by using the addTransferPolicy method.
23. Use the setTransferTarget and setTransferSourcemethod to assign the transfer specifications to the promotion specification.

```
ExcludeDependentTransferPolicy excludeDependentTransferPolicy = new ExcludeDependentTransferPolicy();
ExcludeTransferSpecification excludeTransferSpecification = new ExcludeTransferSpecification();
SelectedMetadata selectedMetadata = new SelectedMetadata();
MimeType mimeType = new MimeType();
mimeType.setValue("application/x-vnd.pasw-dms-rule");
selectedMetadata.addMetadataBase(mimeType);
excludeTransferSpecification.setSelectedMetadata(selectedMetadata);
excludeDependentTransferPolicy.setExcludeTransferSpecification(excludeTransferSpecification);
targetTransferSpec.addTransferPolicy(excludeDependentTransferPolicy);

proSpecification.setTransferTarget(targetTransferSpec);
proSpecification.setTransferSource(sourceTransferSpec);
```

24. Convert the promotion specification to a ByteArrayInputStream object and add the result to the web service call. The following code uses the DataHandler class to make the conversion.

```
DataHandler dataHandler = new DataHandler(new DataSource() {

    public String getContentType() {
      return "application/x-vnd.spss-promotion-policy";
    }

    public InputStream getInputStream() throws IOException {
    byte[] bytes = null;
      StringWriter stringWriter = new StringWriter();
      try {
        proSpecification.marshal(stringWriter);
        bytes = stringWriter.toString().getBytes("UTF8");
      } catch (Exception e) {
        e.printStackTrace();
      }
      return new ByteArrayInputStream(bytes);
    }

    public String getName() {
      return null;
    }

  public OutputStream getOutputStream() throws IOException {
```

```
            return null;
        }
});

AttachmentPart attachmentPart = new AttachmentPart();
attachmentPart.setDataHandler(dataHandler);
((Stub) contentRepository).addAttachment(attachmentPart);

Attachment attachment = new Attachment();
attachment.setHref(attachmentPart.getContentId());
file.setBinaryContent(new BinaryContent());
file.getBinaryContent().addAttachment(attachment);
```

25. Supply the createResource operation with the specifier for the *Promotion Policies* folder and the resource specifier for the promotion policy.

```
IdentificationSpecifier identificationSpecifier = contentRepository.createResource(parentIdSpec, resSpec);
```

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: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>
  <createResource xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb000001069387e017857c" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
   <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>
   </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">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106f3f7b05b3a03" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <Version marker="0:2005-11-16 16:54:38.844"/>
   </IdentificationSpecifier>
  </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 4-12
Fields for createResourcePropagate

Field	Type/Valid Values	Description
ParentIdentificationSpecifier	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.

Field	Type/Valid Values	Description
ResourceSpecifier	ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.
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 4-13
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.

Java example

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

1. Create an IdentificationSpecifier object and set its value to the identifier for the parent object 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 metadate 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, propagating the metadata from the *TEST* version.

```
IdentificationSpecifier parentID = new IdentificationSpecifier();
ResourceID id = new ResourceID();
id.setValue("0a0a4a351eb101db0000010f584899f284c4");
parentID.setIdentifier(id);
```

```
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("TEST");
IdentificationSpecifier idSpec = stub.createResourcePropagate(parentID, specRes, 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">
   <ParentIdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a351eb101db0000010f584899f284c4" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </ParentIdentificationSpecifier>
```

```
    <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>
    <PropagateVersionSpecifier label="TEST" 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">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106f3f7b05b3a03" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <Version marker="0:2005-11-16 16:54:38.844"/>
   </IdentificationSpecifier>
  </createResourcePropagateResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The createResourcePropagateSpecific operation

This operation is deprecated and will be removed in a future release. Consumers of the web service should use the `createResourcePropagate` operation.

Input fields

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

Table 4-14
Fields for createResourcePropagateSpecific

Field	Type/Valid Values	Description
ParentIdentificationSpecifier	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.
SpecificResourceSpecifier	SpecificResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.
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 createResourcePropagateSpecific operation.

Table 4-15
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 createResourceSpecific operation

This operation is deprecated and will be removed in a future release. Consumers of the web service needing to create resources should use the `createResource` operation.

Input fields

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

Table 4-16
Fields for createResourceSpecific

Field	Type/Valid Values	Description
ParentIdentificationSpecifier	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.
SpecificResourceSpecifier	SpecificResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Return information

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

Table 4-17
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 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 4-18
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 4-19
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 4-20
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 identification specifier. If there is not a version instance, the latest version is deleted.

Input fields

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

Table 4-21
Fields for deleteResource

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Java example

The following function uses getResourceID to access the identifier for a supplied resource. The setIdentifier method assigns this value to an IdentificationSpecifier. The deleteResource operation deletes the corresponding repository resource.

```
public void deleteResource(AdaptableResource resource)
  throws ResourceNotFoundException, RepositoryDatabaseException,
    ResourceAuthorizationException, RepositoryException,
    RemoteException, IOException, ServiceException {
  IdentificationSpecifier id = new IdentificationSpecifier();
  id.setIdentifier(resource.getResourceID());
  id.setVersion(null);
  getContentRepository().deleteResource(id);
}
```

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">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>
  <deleteResource xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb000001069387e01785c5"
     xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
  </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 4-22
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 identifer *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 4-23
Return Value

Type	Description
string[]	Labels in use in the repository.

Java example

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

```
public String[] findAllLabels()
  throws RepositoryDatabaseException, RepositoryException,
    RemoteException, IOException, ServiceException {
  return getContentRepository().findAllLabels();
}
```

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 4-24
Fields for getActions

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Return information

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

Table 4-25
Return Value

Type	Description
ActionList	List of action specifications.

Java example

The following function returns an array of action specifications for a resource.

```
public ActionSpecificationType[] getList(IdentificationSpecifier identificationSpecifier, String action)
  throws IOException, ServiceException {
  ContentRepository repository = getContentRepository();
  ActionList actionlist = repository.getActions(identificationSpecifier);
  ActionSpecificationType[] actionSpecificationType =
    actionlist.getActionSpecification();
  return actionSpecificationType;
}
```

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">
    <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
     <identifier value="0a0a4aac00072ffb000001094fa9f57e8028" xsi:type="ResourceID"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    </IdentificationSpecifier>
   </getActions>
  </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 4-26
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.

```
public CustomProperty[] getCustomProperties() throws IOException, ServiceException {
  ContentRepository repository = getContentRepository();
  CustomProperty[] customProperties = repository.getAllCustomProperties();
  return customProperties;
}
```

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 getAllLabelSecurity operation

Retrieves the security information for all labels in the system.

Return information

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

Table 4-27
Return Value

Type	Description
SecureLabel[]	Wrapper for a secure label, contains the label and it's Access Control List (ACL).

Java example

The following sample returns an array of SecureLabel objects corresponding to the labels in the system. Each label object contains an AccessControlList object defining which principals have which permissions.

```
SecureLabel[] labelSec = stub.getAllLabelSecurity();
for (int i = 0; i < labelSec.length; i++) {
  System.out.println("Label = " + labelSec[i].getLabel());
  System.out.println("Can read = " + labelSec[i].getCanRead());
  System.out.println("Can modify = " + labelSec[i].getCanModify());
  AccessControlList acl = labelSec[i].getAcl();
  AccessControlEntry[] ace = acl.getAccessControlEntry();
  for (int j = 0; j < ace.length; j++) {
    System.out.println("Permission = " + ace[j].getPermission().toString());
    System.out.println("Principal = " + ace[j].getPrincipal().getDisplayName());
  }
}
```

SOAP request example

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

SOAP response example

The server responds to a getAllLabelSecurity 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>
    <getAllLabelSecurityResponse xmlns="http://xml.spss.com/repository/remote">
      <SecureLabel label="Draft" canRead="true" canModify="true"
        xmlns="http://xml.spss.com/repository">
        <acl>
          <AccessControlEntry Permission="MODIFY">
            <Principal ID="//gNative//$$security/everyoneGroup"
              DisplayName="–everyone –" Name="$$security/everyoneGroup"
              IsGroup="true"/>
          </AccessControlEntry>
          <AccessControlEntry Permission="READ">
            <Principal ID="//gNative//$$security/everyoneGroup"
              DisplayName="–everyone –" Name="$$security/everyoneGroup"
              IsGroup="true"/>
          </AccessControlEntry>
          <owner ID="" Name=""/>
        </acl>
      </SecureLabel>
      <SecureLabel label="Production" canRead="true" canModify="true"
        xmlns="http://xml.spss.com/repository">
        <acl>
          <AccessControlEntry Permission="MODIFY">
```

```
            <Principal ID="//uAD/spss/kkruger" DisplayName="kkruger (spss)"
              Name="kkruger" IsGroup="false"/>
          </AccessControlEntry>
          <AccessControlEntry Permission="READ">
            <Principal ID="//uAD/spss/kkruger" DisplayName="kkruger (spss)"
              Name="kkruger" IsGroup="false"/>
          </AccessControlEntry>
          <owner ID="" Name=""/>
        </acl>
      </SecureLabel>
    </getAllLabelSecurityResponse>
  </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 (*http://java.sun.com/xml/ns/jdbc/webrowset.xsd*) schema.

Return information

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

Table 4-28
Return Value

Type	Description
RowSetContent	Allows the user to transport a row set either as an out-of-band attachement 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());
```

SOAP request example

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.

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

SOAP response example

The server responds to a getAllLocks 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>
  <getAllLocksResponse xmlns="http://xml.spss.com/repository/remote">
   <RowSetContent pageNumber="1" xmlns="http://xml.spss.com/repository">
    <content>&lt;?xml version=&quot;1.0&quot;?&gt; &lt;webRowSet
     xmlns=&quot;http://java.sun.com/xml/ns/jdbc&quot;
     xmlns:xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
     xsi:schemaLocation=&quot;http://java.sun.com/xml/ns/jdbc
     http://java.sun.com/xml/ns/jdbc/webrowset.xsd&quot;&gt; &lt;properties&gt;
     &lt;command&gt;&lt;null/&gt;&lt;/command&gt;
     &lt;concurrency&gt;1008&lt;/concurrency&gt;
     &lt;datasource&gt;&lt;null/&gt;&lt;/datasource&gt;
     &lt;escape-processing&gt;true&lt;/escape-processing&gt;
     &lt;fetch-direction&gt;1000&lt;/fetch-direction&gt;
     &lt;fetch-size&gt;0&lt;/fetch-size&gt;
     &lt;isolation-level&gt;2&lt;/isolation-level&gt;
     &lt;key-columns&gt; &lt;/key-columns&gt; &lt;map&gt;
     &lt;/map&gt; &lt;max-field-size&gt;0&lt;/max-field-size&gt;
     &lt;max-rows&gt;0&lt;/max-rows&gt;
     &lt;query-timeout&gt;0&lt;/query-timeout&gt;
     &lt;read-only&gt;true&lt;/read-only&gt;
     &lt;rowset-type&gt;ResultSet.TYPE_SCROLL_INSENSITIVE&lt;/rowset-type&gt;
     &lt;show-deleted&gt;false&lt;/show-deleted&gt;
     &lt;table-name&gt;&lt;null/&gt;&lt;/table-name&gt;
     &lt;url&gt;&lt;null/&gt;&lt;/url&gt; &lt;sync-provider&gt;
     &lt;sync-provider-name&gt;com.sun.rowset.providers.RIOptimisticProvider&lt;/sync-provider-name&gt;
     &lt;sync-provider-vendor&gt;Sun Microsystems
     Inc.&lt;/sync-provider-vendor&gt;
     &lt;sync-provider-version&gt;1.0&lt;/sync-provider-version&gt;
     &lt;sync-provider-grade&gt;2&lt;/sync-provider-grade&gt;
     &lt;data-source-lock&gt;1&lt;/data-source-lock&gt;
```

</sync-provider> </properties> <metadata>
<column-count>6</column-count>
<column-definition>
<column-index>1</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>36</column-display-size>
<column-label>objid</column-label>
<column-name>objid</column-name>
<schema-name></schema-name>
<column-precision>18</column-precision>
<column-scale>0</column-scale>
<table-name>SPSSCMOR_RELATIONSHIP</table-name>
<catalog-name></catalog-name>
<column-type>-2</column-type>
<column-type-name>binary</column-type-name>
</column-definition> <column-definition>
<column-index>2</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>path</column-label>
<column-name>path</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>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> </metadata> <data>
<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> <currentRow>
<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> <currentRow>
<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> </data> </webRowSet> </content>
</RowSetContent>
</getAllLocksResponse>
</soapenv:Body>
</soapenv:Envelope>

The getAllVersions operation

Retrieves metadata information for all versions of a specified resource. Any version information in the resource is ignored.

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 4-29
Fields for getAllVersions

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Return information

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

Table 4-30
Return Value

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

Java example

The following function returns an array of file resources corresponding to all versions of a specified file. The getResourceID method returns the identifier for the file, which is assigned to an IndentificationSpecifier object using setIdentifier. The getAllVersions operation returns an array of resource specifiers for the file versions.

```
public AdaptableFile[] getAllVersions(AdaptableFile file)
  throws IOException, ServiceException, CoreException {
  ContentRepository repository = getContentRepository();
  IdentificationSpecifier idSpec = new IdentificationSpecifier();
  idSpec.setIdentifier(file.getResourceID());
  ResourceSpecifier[] versions = repository.getAllVersions(idSpec);
  AdaptableFile[] fileVersions = new AdaptableFile[versions.length];
  Object parent = file.getParent();
  for (int i = 0; i < fileVersions.length; i++) {
    Resource resourceVersion = versions[i].getResource();
    fileVersions[i] = (AdaptableFile) getAdaptableResource(parent, resourceVersion);
  }
```

```
  return fileVersions;
}
```

To access the actual resources, use the getResource method for the resource specifiers.

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">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb000001094fa9f57ecd59" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
  </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 xmlns="http://xml.spss.com/repository">
    <Resource canWrite="true" canDelete="true" canModifyPermissions="true"
     xsi:type="File" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <ResourceID value="0a0a4aac00072ffb000001094fa9f57ecd59"/>
```

```
      <Version marker="0:2006-02-21 17:19:50.796"/>
      <CreationDate value="2006-02-21T17:19:50.793-06:00"/>
      <ModificationDate value="2006-02-21T17:19:50.793-06:00"/>
      <Title value="Job1"/>
      <ResourcePath value="/Job1" hierarchyType="folder"/>
      <CreatedBy value="kkreuter"/>
      <ObjectCreationDate value="2006-02-21T17:19:50.593-06:00"/>
      <ObjectLastModifiedBy value="kkreuter"/>
      <ObjectLastModifiedDate value="2006-02-21T17:19:51.000-06:00"/>
      <AccessControlList poe="CMOR">
       <AccessControlEntry Permission="READ">
        <Principal ID="//gNative//$$security/everyoneGroup"
         DisplayName="–everyone –" Name="$$security/everyoneGroup"
         IsGroup="true"/>
       </AccessControlEntry>
       <owner ID="//uNative//kkreuter" DisplayName="kkreuter" Name="kkreuter" IsGroup="false"/>
      </AccessControlList>
      <Author value="kkreuter"/>
      <MimeType value="application/x-vnd.spss-prms-job"/>
     </Resource>
    </ResourceSpecifier>
   </getAllVersionsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The getAllVersionsReturnSpecific operation

Retrieves metadata information for all versions of a specified resource. Any version information in the resource is ignored.

The difference between this operation and the getAllVersions operation is the structure of the wrapper for the returned resources. The getAllVersions operation returns an array of ResourceSpecifier objects detailing the resources. Different resource types are handled through the use of XMLSchema instance types. In contrast, the getAllVersionsReturnSpecific operation returns an array of SpecificResourceSpecifier objects. Different resource types are handled through the use of different objects for the types. For example, instead of returning a Resource object of the *File* type, getAllVersionsReturnSpecific returns a File object. Clients that cannot properly handle instance types should use getAllVersionsReturnSpecific instead of getAllVersions.

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 getAllVersionsReturnSpecific operation.

Table 4-31
Fields for getAllVersionsReturnSpecific

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Return information

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

Table 4-32
Return Value

Type	Description
SpecificResourceSpecifier[]	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Java example

The following function returns an array of specific resource specifiers corresponding to all versions of a specified file.

```
IdentificationSpecifier id = new IdentificationSpecifier();
Identifier id = new Identifier();
id.setValue("0a0a4a35c75d16910000010eec1efbfc8006");
is.setIdentifier(id);
SpecificResourceSpecifier[] rs = stub.getAllVersionsReturnSpecific(is);
for (int i = 0; i < rs.length; i++) {
  System.out.println(rs[i].getFile().getVersion().getMarker());
}
```

SOAP request example

Client invocation of the getAllVersionsReturnSpecific 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>
   <getAllVersionsReturnSpecific xmlns="http://xml.spss.com/repository/remote">
    <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
     <identifier value="0a0a4a35c75d16910000010eec1efbfc8006"/>
    </IdentificationSpecifier>
   </getAllVersionsReturnSpecific>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getAllVersionsReturnSpecific 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>
  <getAllVersionsReturnSpecificResponse xmlns="http://xml.spss.com/repository/remote">
   <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
    <File canWrite="true" canDelete="true" canModifyPermissions="true">
     <ResourceID value="0a0a4a35c75d16910000010eec1efbfc8006"/>
     <Version marker="0:2006-11-15 09:01:14.203"/>
     <CreationDate value="2006-11-15T09:01:14.203-06:00"/>
     <ModificationDate value="2006-11-15T09:01:14.203-06:00"/>
     <Title value="Tree"/>
     <ResourcePath value="/Jobs/Tree" hierarchyType="folder"/>
     <CreatedBy value="admin"/>
     <ObjectCreationDate value="2006-11-15T09:01:14.127-06:00"/>
     <ObjectLastModifiedBy value="admin"/>
     <ObjectLastModifiedDate value="2006-11-17T10:00:06.537-06:00"/>
     <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-prms-job"/>
```

```
  </File>
</SpecificResourceSpecifier>
<SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
  <File canWrite="true" canDelete="true" canModifyPermissions="true">
    <ResourceID value="0a0a4a35c75d16910000010eec1efbfc8006"/>
    <Version marker="1:2006-11-17 10:02:19.276"/>
    <CreationDate value="2006-11-17T10:02:19.277-06:00"/>
    <ModificationDate value="2006-11-17T10:02:20.527-06:00"/>
    <Title value="Tree"/>
    <ResourcePath value="/Jobs/Tree" hierarchyType="folder"/>
    <CreatedBy value="admin"/>
    <ObjectCreationDate value="2006-11-15T09:01:14.127-06:00"/>
    <ObjectLastModifiedBy value="admin"/>
    <ObjectLastModifiedDate value="2006-11-17T10:00:06.537-06:00"/>
    <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-prms-job"/>
    <ContentSize value="7272"/>
  </File>
</SpecificResourceSpecifier>
<SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
  <File canWrite="true" canDelete="true" canModifyPermissions="true">
    <ResourceID value="0a0a4a35c75d16910000010eec1efbfc8006"/>
    <Version marker="2:2006-11-17 10:04:03.406"/>
    <CreationDate value="2006-11-17T10:04:03.407-06:00"/>
    <ModificationDate value="2006-11-17T10:04:03.670-06:00"/>
    <Title value="Tree"/>
    <ResourcePath value="/Jobs/Tree" hierarchyType="folder"/>
    <CreatedBy value="admin"/>
    <ObjectCreationDate value="2006-11-15T09:01:14.127-06:00"/>
    <ObjectLastModifiedBy value="admin"/>
    <ObjectLastModifiedDate value="2006-11-17T10:00:06.537-06:00"/>
    <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-prms-job"/>
      <ContentSize value="7585"/>
     </File>
    </SpecificResourceSpecifier>
   </getAllVersionsReturnSpecificResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The getBulkResourceMetadata operation

Retrieves an array of resources for a given array of identification specifiers. 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 4-33
Fields for getBulkResourceMetadata

Field	Type/Valid Values	Description
IdentificationSpecifier	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.
SelectedMetadata	SelectedMetadata	Provides a way to select specific metadata for retrieval or update

Return information

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

Table 4-34
Return Value

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

Java example

To retrieve resource metadata in bulk:

1. Create an array of IdentificationSpecifier objects for the resources. Set the identification and version information for each resource.
2. Create a SelectedMetadata object to identify the metadata fields to be returned.
3. Supply the getBulkResourceMetadata operation with the identification and metadata objects.

   The following code retrieves the IDs, paths, and versions for two files.

```
IdentificationSpecifier[] is = new IdentificationSpecifier[2];
ResourceID id = new ResourceID();
id.setValue("0a0a4a35c72b39630000010f588e773d8046");
is[0].setIdentifier(id);
Version vsion = new Version();
vsion.setMarker("0:2006-12-06 10:30:13.338");
is[0].setVersion(vsion);
id.setValue("0a0a4a351eb101db0000010f584899f284c7");
is[1].setIdentifier(id);
vsion.setMarker("0:2006-12-06 09:39:29.992");
is[1].setVersion(vsion);

SelectedMetadata sm = new SelectedMetadata();
sm.setMetadataBase(new ResourceID());
ResourcePath rp = new ResourcePath();
rp.setHierarchyType(HierarchyType.FOLDER);
sm.setMetadataBase(rp);
sm.setMetadataBase(new Version());

ResourceSpecifier[] resSpec = stub.getBulkResourceMetadata(is, sm);
for (int i = 0; i < resSpec.length; i++) {
  System.out.println("ID = " + resSpec[i].getResource().getResourceID().getValue());
  System.out.println("ID = " + resSpec[i].getResource().getResourcePath().getValue());
  System.out.println("ID = " + resSpec[i].getResource().getVersion().getMarker());
}
```

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: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>
  <getBulkResourceMetadata xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a35c72b39630000010f588e773d8046" xsi:type="ResourceID"/>
    <Version marker="0:2006-12-06 10:30:13.338"/>
   </IdentificationSpecifier>
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a351eb101db0000010f584899f284c7" xsi:type="ResourceID"/>
    <Version marker="0:2006-12-06 09:39:29.992"/>
   </IdentificationSpecifier>
   <SelectedMetadata xmlns="http://xml.spss.com/repository">
    <MetadataBase xsi:type="ResourceID" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase hierarchyType="folder" xsi:type="ResourcePath"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase xsi:type="Version" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </SelectedMetadata>
  </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="true" canModifyPermissions="true" xsi:type="File">
     <ResourceID value="0a0a4a35c72b39630000010f588e773d8046"/>
     <Version marker="0:2006-12-06 10:30:13.338"/>
     <Title value="tree_model.sps"/>
     <ResourcePath value="/StatisticsLibrary/tree_model.sps" 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="true" canDelete="true" canModifyPermissions="true" xsi:type="File">
      <ResourceID value="0a0a4a351eb101db0000010f584899f284c7"/>
      <Version marker="0:2006-12-06 09:39:29.992"/>
      <Title value="pes.rptdesign"/>
      <ResourcePath value="/Reports/pes.rptdesign" 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>
   </getBulkResourceMetadataResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The getBulkResourceMetadataReturnSpecific operation

Retrieves an array of resources for a given array of identification specifiers. The information returned may be limited to a select set of metadata.

The difference between this operation and the getBulkResourceMetadata operation is the structure of the wrapper for the returned resources. The getBulkResourceMetadata operation returns an array of ResourceSpecifier objects detailing the resources. Different resource types are handled through the use of XMLSchema instance types. In contrast, the getBulkResourceMetadataReturnSpecific operation returns an array of SpecificResourceSpecifier objects. Different resource types are handled through the use of different objects for the types. For example, instead of returning a Resource object of the *File* type, getBulkResourceMetadataReturnSpecific returns a File object. Clients that cannot properly handle instance types should use getBulkResourceMetadataReturnSpecific instead of getBulkResourceMetadata.

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 getBulkResourceMetadataReturnSpecific operation.

Table 4-35
Fields for getBulkResourceMetadataReturnSpecific

Field	Type/Valid Values	Description
IdentificationSpecifier	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.
SelectedMetadata	SelectedMetadata	Provides a way to select specific metadata for retrieval or update

Return information

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

Table 4-36
Return Value

Type	Description
SpecificResourceSpecifier[]	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Java example

To retrieve resource metadata in bulk:

1. Create an array of IdentificationSpecifier objects for the resources. Set the identification and version information for each resource.
2. Create a SelectedMetadata object to identify the metadata fields to be returned.
3. Supply the getBulkResourceMetadataReturnSpecific operation with the identification and metadata objects.

   The following code retrieves the IDs, paths, and versions for two files.

```
IdentificationSpecifier[] is = new IdentificationSpecifier[2];
ResourceID id = new ResourceID();
id.setValue("0a0a4a35c72b39630000010f588e773d8046");
is[0].setIdentifier(id);
Version vsion = new Version();
vsion.setMarker("0:2006-12-06 10:30:13.338");
is[0].setVersion(vsion);
id.setValue("0a0a4a351eb101db0000010f584899f284c7");
is[1].setIdentifier(id);
vsion.setMarker("0:2006-12-06 09:39:29.992");
is[1].setVersion(vsion);

SelectedMetadata sm = new SelectedMetadata();
sm.setMetadataBase(new ResourceID());
ResourcePath rp = new ResourcePath();
rp.setHierarchyType(HierarchyType.FOLDER);
sm.setMetadataBase(rp);
sm.setMetadataBase(new Version());

SpecificResourceSpecifier[] resSpec = stub.getBulkResourceMetadataReturnSpecific(is, sm);
for (int i = 0; i < resSpec.length; i++) {
  System.out.println("ID = " + resSpec[i].getFile().getResourceID().getValue());
  System.out.println("ID = " + resSpec[i].getFile().getResourcePath().getValue());
  System.out.println("ID = " + resSpec[i].getFile().getVersion().getMarker());
}
```

SOAP request example

Client invocation of the getBulkResourceMetadataReturnSpecific 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>
  <getBulkResourceMetadataReturnSpecific xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a35c72b39630000010f588e773d8046" xsi:type="ResourceID"/>
    <Version marker="0:2006-12-06 10:30:13.338"/>
   </IdentificationSpecifier>
```

```
    <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
     <identifier value="0a0a4a351eb101db0000010f584899f284c7" xsi:type="ResourceID"/>
     <Version marker="0:2006-12-06 09:39:29.992"/>
    </IdentificationSpecifier>
    <SelectedMetadata xmlns="http://xml.spss.com/repository">
     <MetadataBase xsi:type="ResourceID" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
     <MetadataBase hierarchyType="folder" xsi:type="ResourcePath"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
     <MetadataBase xsi:type="Version" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    </SelectedMetadata>
   </getBulkResourceMetadataReturnSpecific>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getBulkResourceMetadataReturnSpecific 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>
  <getBulkResourceMetadataReturnSpecificResponse xmlns="http://xml.spss.com/repository/remote">
   <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
    <File canWrite="true" canDelete="true" canModifyPermissions="true">
     <ResourceID value="0a0a4a35c72b39630000010f588e773d8046"/>
     <Version marker="0:2006-12-06 10:30:13.338"/>
     <Title value="tree_model.sps"/>
     <ResourcePath value="/StatisticsLibrary/tree_model.sps" 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>
    </File>
   </SpecificResourceSpecifier>
   <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
    <File canWrite="true" canDelete="true" canModifyPermissions="true">
     <ResourceID value="0a0a4a351eb101db0000010f584899f284c7"/>
     <Version marker="0:2006-12-06 09:39:29.992"/>
     <Title value="pes.rptdesign"/>
     <ResourcePath value="/Reports/pes.rptdesign" 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>
     </File>
    </SpecificResourceSpecifier>
   </getBulkResourceMetadataReturnSpecificResponse>
  </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.

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 getChildren operation.

Table 4-37
Fields for getChildren

Field	Type/Valid Values	Description
ParentIdentificationSpecifier	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.
SelectedMetadata	SelectedMetadata	Provides a way to select specific metadata for retrieval or update

Return information

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

Table 4-38
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, which would consist of both subfolders and files, of a parent folder. The only metadata returned would be content size, creation date, modification date, ACL, and title. It is not necessary to set values for the desired metadata, only to instantiate the objects.

```
ResourcePath rp = new ResourcePath();
rp.setValue("/myFolder");
SelectedMetadata sm = new SelectedMetadata();
sm.addMetadataBase(new Title());
sm.addMetadataBase(new CreationDate());
sm.addMetadataBase(new AccessControlList());
sm.addMetadataBase(new ModificationDate());
sm.addMetadataBase(new ContentSize());
ResourceSpecifier[] resources = stub.getChildren(getIDSpecifier(rp), sm);
```

Note that for the case of a folder child, which does not have a *content size* metadata instance, that particular request will be ignored. Only the selected metadata which is valid for the resource can be returned.

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">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>
  <getChildren xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106747cdec691f5" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
   <SelectedMetadata xmlns="http://xml.spss.com/repository">
    <MetadataBase xsi:type="Author"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase language="EN" xsi:type="Description"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase xsi:type="ObjectLastModifiedBy"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase xsi:type="ObjectLastModifiedDate"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase xsi:type="Title"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase language="EN" xsi:type="Description"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase xsi:type="MimeType"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase hierarchyType="folder" xsi:type="ResourcePath"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <MetadataBase xsi:type="Version"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </SelectedMetadata>
  </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="true" canDelete="true" canModifyPermissions="true"
     xsi:type="File" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <ResourceID value="0a0a4aac00072ffb00000106747cdec691f9"/>
     <Version marker="0:2005-09-23 15:16:59.842"/>
     <Title value="test.txt"/>
     <ResourcePath value="/TextFiles/test.txt" hierarchyType="folder"/>
```

```
      <ObjectLastModifiedBy value="admin"/>
      <ObjectLastModifiedDate value="2005-09-23T15:17:00.217-05:00"/>
      <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="text/plain"/>
     </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 4-39
Fields for getChildrenOptions

Field	Type/Valid Values	Description
ParentIdentificationSpecifier	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.
ChildrenSpecification	ChildrenSpecification	Expresses search criteria for querying the repsoitory.

Return information

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

Table 4-40
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">
   <ParentIdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="/myFolder" xsi:type="ResourcePath"/>
   </ParentIdentificationSpecifier>
   <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 getChildrenReturnSpecific 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.

The difference between this operation and the getChildren operation is the structure of the wrapper for the returned resources. The getChildren operation returns an array of ResourceSpecifier objects detailing the resources. Different resource types are handled through the use of XMLSchema instance types. In contrast, the getChildrenReturnSpecific operation returns an array of SpecificResourceSpecifier objects. Different resource types are handled through the use of different objects for the types. For example, instead of returning a Resource object of the *File* type, getChildrenReturnSpecific returns a File object. Clients that cannot properly handle instance types should use getChildrenReturnSpecific instead of getChildren.

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 getChildrenReturnSpecific operation.

Table 4-41
Fields for getChildrenReturnSpecific

Field	Type/Valid Values	Description
ParentIdentificationSpecifier	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.
SelectedMetadata	SelectedMetadata	Provides a way to select specific metadata for retrieval or update

Return information

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

Table 4-42
Return Value

Type	Description
SpecificResourceSpecifier[]	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, which would consist of both subfolders and files, of a parent folder. The only metadata returned would be the author, who last modified the resource, and the date of the modification. It is not necessary to set values for the desired metadata, only to instantiate the objects.

```
IdentificationSpecifier id = new IdentificationSpecifier();
ResourceID id = new ResourceID();
id.setValue("0a0a4a35c75d16910000010eec1efbfc802f");
is.setIdentifier(id);

SelectedMetadata sm = new SelectedMetadata();
sm.addMetadataBase(new Author());
sm.addMetadataBase(new ObjectLastModifiedBy());
sm.addMetadataBase(new ObjectLastModifiedDate());
SpecificResourceSpecifier[] resources =
  stub.getChildrenReturnSpecific(is, sm);
```

SOAP request example

Client invocation of the getChildrenReturnSpecific 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>
  <getChildrenReturnSpecific xmlns="http://xml.spss.com/repository/remote">
   <ParentIdentificationSpecifier xmlns="http://xml.spss.com/repository">
```

```
      <identifier value="0a0a4a35c75d16910000010eec1efbfc802f"/>
    </ParentIdentificationSpecifier>
    <SelectedMetadata xmlns="http://xml.spss.com/repository">
     <Author/>
     <ObjectLastModifiedBy/>
     <ObjectLastModifiedDate/>
    </SelectedMetadata>
   </getChildrenReturnSpecific>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getChildrenReturnSpecific 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>
  <getChildrenReturnSpecificResponse xmlns="http://xml.spss.com/repository/remote">
   <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
    <File canWrite="true" canDelete="true" canModifyPermissions="true">
     <ResourceID value="0a0a4a35bad1b0750000010f3a50fe4980e6"/>
     <Title value="drugplot.str"/>
     <ResourcePath value="/ModelerStreamLibrary/drugplot.str" hierarchyType="folder"/>
     <ObjectLastModifiedBy value="admin"/>
     <ObjectLastModifiedDate value="2006-11-30T14:19:21.253-06:00"/>
     <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"/>
    </File>
   </SpecificResourceSpecifier>
   <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
    <Folder canWrite="true" canDelete="true" canModifyPermissions="true" hasChildren="true"
     hasFolder="false">
     <ResourceID value="0a0a4a35c75d16910000010eec1efbfc8032"/>
     <Title value="Data Preparation"/>
     <ResourcePath value="/ModelerStreamLibrary/Data Preparation" hierarchyType="folder"/>
     <ObjectLastModifiedBy value="admin"/>
     <ObjectLastModifiedDate value="2006-11-15T09:07:15.060-06:00"/>
     <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>
    </Folder>
   </SpecificResourceSpecifier>
   <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
    <Folder canWrite="true" canDelete="true" canModifyPermissions="true" hasChildren="true"
     hasFolder="false">
     <ResourceID value="0a0a4a35c75d16910000010eec1efbfc8171"/>
     <Title value="Modeling"/>
     <ResourcePath value="/ModelerStreamLibrary/Modeling" hierarchyType="folder"/>
     <ObjectLastModifiedBy value="admin"/>
     <ObjectLastModifiedDate value="2006-11-15T09:08:18.740-06:00"/>
     <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>
    </Folder>
   </SpecificResourceSpecifier>
   <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
    <Folder canWrite="true" canDelete="true" canModifyPermissions="true" hasChildren="true"
     hasFolder="false">
     <ResourceID value="0a0a4a35c75d16910000010eec1efbfc8132"/>
     <Title value="Data Understanding"/>
     <ResourcePath value="/ModelerStreamLibrary/Data Understanding" hierarchyType="folder"/>
     <ObjectLastModifiedBy value="admin"/>
     <ObjectLastModifiedDate value="2006-11-15T09:07:36.020-06:00"/>
     <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>
    </Folder>
   </SpecificResourceSpecifier>
  </getChildrenReturnSpecificResponse>
```

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

The getCustomPropertyValues operation

Returns the values for custom properties that have been defined for a resource.

Input fields

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

Table 4-43
Fields for getCustomPropertyValues

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Return information

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

Table 4-44
Return Value

Type	Description
CustomPropertyValue[]	The value assigned by a user to an object for a particular custom property.

Java example

The following function retunrs an array of CustomPropertyValue objects for a specified resource. The getResourceID method returns the identifier for the resource, which is assigned to an IndentificationSpecifier object using setIdentifier. The getCustomPropertyValues operation returns the array of values for this specifier.

```
public CustomPropertyValue[] getCustomPropertyValues(AdaptableResource resource)
  throws IOException, ServiceException {
  IdentificationSpecifier idSpecifier = new IdentificationSpecifier();
  idSpecifier.setIdentifier(resource.getResourceID());
  ContentRepository repository = getContentRepository();
  CustomPropertyValue[] customPropertyValues = repository.getCustomPropertyValues(idSpecifier);
  return customPropertyValues;
}
```

SOAP request example

Client invocation of the getCustomPropertyValues 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>
  <getCustomPropertyValues xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106561fccf88148" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
  </getCustomPropertyValues>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getCustomPropertyValues 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>
  <getCustomPropertyValuesResponse xmlns="http://xml.spss.com/repository/remote">
   <CustomPropertyValue label="Reviewed by" identifier="0a0a4aac00072ffb00000106f3f7b05b3856"
    xmlns="http://xml.spss.com/repository">
    <select multipleSelect="false">
     <selectionValue isSelected="true" value="Richard"/>
     <selectionValue value="Patrick"/>
     <selectionValue value="John"/>
    </select>
   </CustomPropertyValue>
   <CustomPropertyValue label="Reviewed" identifier="0a0a4aac00072ffb00000106561fccf88967"
    xmlns="http://xml.spss.com/repository">
```

```
      <freeform type="boolean"/>
    </CustomPropertyValue>
   </getCustomPropertyValuesResponse>
  </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 4-45
Fields for getFault

Field	Type/Valid Values	Description
test	int	

The getFile operation

Retrieving a resource can be achieved in several ways: getResource, getFile, getChildren, and query. In all cases but getFile, only the metadata of the resource is returned. The getFile operation, on the other hand, 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 identification specifier. If the version information is null, 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 4-46
Fields for getFile

Field	Type/Valid Values	Description
IdentificationSpecifier	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.
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

Return information

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

Table 4-47
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 the content for a file resource, writing the content to a file on the local file system.

Notice that the transport type is specified as MIME, which is the default. Use MIME for a java client and DIME for a .NET client. Both MIME and DIME are attachments. If the content is small, it can be transported inside the SOAP envelope using BASE64.

```
ResourcePath rp = new ResourcePath();
rp.setValue("/myfolderName/myNewFile.xls");
Version v = new Version();
v.addLabel("PRODUCTION");
ResourceSpecifier rs=stub.getFile(getResourceSpecifier(rp,v), DeliveryType.MIME);
File f = (File)rs.getResource();
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\\retrieve.xls");
      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">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>
  <getFile xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106a7c0f12d8116" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <Version marker="0:2005-09-30 14:50:59.086"/>
   </IdentificationSpecifier>
   <acceptDelivery>MIME</acceptDelivery>
   <ns2:SelectedMetadata xsi:nil="true" xmlns:ns2="http://xml.spss.com/repository"/>
```

```
  </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="true" canDelete="true" canModifyPermissions="true"
     xsi:type="File" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <ResourceID value="0a0a4aac00072ffb00000106561fccf88177"/>
     <Version marker="0:2005-09-14 14:49:23.234">
      <label>Production</label>
     </Version>
     <CreationDate value="2005-09-14T14:49:23.233-05:00"/>
     <ModificationDate value="2005-09-14T14:49:24.170-05:00"/>
     <Title value="M01_TrainTestPartition.str"/>
     <Description value="Modeler Stream Library:
      M01_TrainTestPartition.str.&#xa;" language="en"/>
     <ResourcePath value="/ModelerStreamLibrary/Modeling/M01_TrainTestPartition.str"
      hierarchyType="folder"/>
     <CreatedBy value="admin"/>
     <ObjectCreationDate value="2005-09-14T14:49:22.967-05:00"/>
     <ObjectLastModifiedBy value="admin"/>
     <ObjectLastModifiedDate value="2005-09-14T14:49:24.203-05:00"/>
     <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="Bart Simpson"/>
     <MimeType value="application/x-vnd.spss-clementine-stream"/>
     <ContentSize value="9727"/>
     <BinaryContent>
      <Attachment href="3AAFA98704B70A827D56DD9057B3DE5D"/>
```

```
      </BinaryContent>
      <associatedTopicList>
       <topicIdentifier value="0a0a4aac00072ffb00000106561fccf88043"
        xsi:type="ResourceID"/>
       <topicIdentifier value="0a0a4aac00072ffb00000106561fccf88041"
        xsi:type="ResourceID"/>
      </associatedTopicList>
     </Resource>
    </ResourceSpecifier>
   </getFileResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The getFileReturnSpecific operation

Retrieving a resource can be achieved in several ways: getResource, getFile, getChildren, and query. In all cases but getFile, only the metadata of the resource is returned. The getFile operation, on the other hand, 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 identification specifier. If the version information is null, 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 difference between this operation and the getFile operation is the structure of the wrapper for the returned resources. The getFile operation returns a ResourceSpecifier object detailing the resource. Different resource types are handled through the use of XMLSchema instance types. In contrast, the getFileReturnSpecific operation returns a SpecificResourceSpecifier object. Different resource types are handled through the use of different objects for the types. For example, instead of returning a Resource object of the *File* type, getFileReturnSpecific returns a File object. Clients that cannot properly handle instance types should use getFileReturnSpecific instead of getFile.

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 getFileReturnSpecific operation.

Table 4-48
Fields for getFileReturnSpecific

Field	Type/Valid Values	Description
IdentificationSpecifier	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.
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

Return information

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

Table 4-49
Return Value

Type	Description
SpecificResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Java example

The following example retrieves the content for a file resource, writing the content to a file on the local file system.

```
IdentificationSpecifier id = new IdentificationSpecifier();
ResourceID id = new ResourceID();
id.setValue("0a0a4a353da969fa0000010ef0fcc01f8649");
is.setIdentifier(id);

SpecificResourceSpecifier rs = stub.getFileReturnSpecific(is, DeliveryType.MIME);
File f = rs.getFile();
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\\retrieve.xls");
    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 getFileReturnSpecific 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>
  <getFileReturnSpecific xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a353da969fa0000010ef0fcc01f8649"/>
   </IdentificationSpecifier>
   <acceptDelivery>MIME</acceptDelivery>
   </getFileReturnSpecific>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getFileReturnSpecific 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>
  <getFileReturnSpecificResponse xmlns="http://xml.spss.com/repository/remote">
   <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
    <File canWrite="true" canDelete="true" canModifyPermissions="true">
     <ResourceID value="0a0a4a353da969fa0000010ef0fcc01f8649"/>
     <Version marker="0:2006-11-17 09:59:42.666"/>
     <CreationDate value="2006-11-17T09:59:42.667-06:00"/>
     <ModificationDate value="2006-11-17T09:59:42.933-06:00"/>
     <Title value="tree_model.sps"/>
     <ResourcePath value="/StatisticsLibrary/tree_model.sps" hierarchyType="folder"/>
     <CreatedBy value="admin"/>
     <ObjectCreationDate value="2006-11-17T09:59:42.573-06:00"/>
     <ObjectLastModifiedBy value="admin"/>
     <ObjectLastModifiedDate value="2006-11-17T09:59:42.933-06:00"/>
     <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-spss-syntax"/>
     <ContentSize value="682"/>
     <BinaryContent>
      <Attachment href="B736AFD5CFD3736D19233CD1C0BB2BA3"/>
     </BinaryContent>
    </File>
   </SpecificResourceSpecifier>
  </getFileReturnSpecificResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The getIdentificationSpecifier operation

Retrieves an identification specifier for a specified resource based on the internal implementation identifier.

Input fields

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

Table 4-50
Fields for getIdentificationSpecifier

Field	Type/Valid Values	Description
implementationID	string	Implementation ID for the resource.

Return information

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

Table 4-51
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.

Java example

The following example retrieves the identification specifier for the resource corresponding to the supplied implementation ID.

```
String implId = "0a0a4aac00072ffb000001094fa9f57eca1c";
IdentificationSpecifier is = stub.getIdentificationSpecifier(implId);
System.out.println("Implementation ID = " + implID);
System.out.println("Identification Specifier = " +
  is.getIdentifier().getValue());
System.out.println("Version = " + is.getVersion().getMarker());
```

SOAP request example

Client invocation of the getIdentificationSpecifier 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>
   <getIdentificationSpecifier xmlns="http://xml.spss.com/repository/remote">
    <implementationID>0a0a4aac00072ffb000001094fa9f57eca1c</implementationID>
   </getIdentificationSpecifier>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getIdentificationSpecifier 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>
  <getIdentificationSpecifierResponse xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb000001094fa9f57ec98e" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <Version marker="3:2006-02-21 09:32:05.218"/>
   </IdentificationSpecifier>
  </getIdentificationSpecifierResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The getLabelSecurity operation

Retrieves the security information for a specified label in the system.

Input fields

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

Table 4-52
Fields for getLabelSecurity

Field	Type/Valid Values	Description
label	string	The label for which security information will be retrieved.

Return information

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

Table 4-53
Return Value

Type	Description
SecureLabel	Wrapper for a secure label, contains the label and it's Access Control List (ACL).

Java example

To access the security definition for a single label, supply the getLabelSecurity operation with a string corresponding to the label. The operation returns a SecureLabel object containing an AccessControlList object defining which principals have which permissions for the label.

```
String myLabel = "Production";
SecureLabel labelSec = stub.getLabelSecurity(myLabel);

System.out.println("Label = " + labelSec.getLabel());
System.out.println("Can read = " + labelSec.getCanRead());
System.out.println("Can modify = " + labelSec.getCanModify());
AccessControlList acl = labelSec.getAcl();
AccessControlEntry[] ace = acl.getAccessControlEntry();
for (int j = 0; j < ace.length; j++) {
  System.out.println("Permission = " + ace[j].getPermission().toString());
  System.out.println("Principal = " + ace[j].getPrincipal().getDisplayName());
}
```

SOAP request example

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

SOAP response example

The server responds to a getLabelSecurity 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>
    <getLabelSecurityResponse xmlns="http://xml.spss.com/repository/remote">
      <SecureLabel label="Production" canRead="true" canModify="true"
        xmlns="http://xml.spss.com/repository">
        <acl>
          <AccessControlEntry Permission="READ">
            <Principal ID="//uAD/spss/aessa" DisplayName="aessa (spss)"
              Name="aessa" IsGroup="false"/>
          </AccessControlEntry>
          <AccessControlEntry Permission="MODIFY">
            <Principal ID="//uAD/spss/kkruger" DisplayName="kkruger (spss)"
              Name="kkruger" IsGroup="false"/>
          </AccessControlEntry>
          <AccessControlEntry Permission="READ">
            <Principal ID="//uAD/spss/kkruger" DisplayName="kkruger (spss)"
              Name="kkruger" IsGroup="false"/>
          </AccessControlEntry>
          <owner ID="" Name=""/>
        </acl>
      </SecureLabel>
    </getLabelSecurityResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The getResource operation

Returns all metadata describing a resource. In contrast, operations such as getChildren or query return 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 identification specifier. If the version information is null, 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 4-54
Fields for getResource

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Return information

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

Table 4-55
Return Value

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

Java example

The following function accepts an identifier and a Version object identifying a specific resource to be retrieved. The setIdentifier and setVersion methods assign this information to an IdentificationSpecifier object. The getResource operation returns a resource specifier for this object.

```
private Resource getResourceFromServer(Identifier identifier, Version version)
  throws ResourceNotFoundException, RepositoryDatabaseException,
    ResourceAuthorizationException, RepositoryException,
    RemoteException, IOException, ServiceException {
  IdentificationSpecifier identification = new IdentificationSpecifier();
  identification.setIdentifier(identifier);
  identification.setVersion(version);
  ResourceSpecifier resourceSpecifier =
    getContentRepository().getResource(identification);
  return resourceSpecifier.getResource();
}
```

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">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>
  <getResource xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="/" hierarchyType="folder" xsi:type="ResourcePath"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
  </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="true" canDelete="true" canModifyPermissions="true"
     hasChildren="true" xsi:type="Folder" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <ResourceID value="00000000000000000000000000000000cc"/>
     <Title value="/"/>
     <ResourcePath value="/" hierarchyType="server"/>
     <ObjectCreationDate value="2005-06-13T16:04:33.000-05:00"/>
     <ObjectLastModifiedDate value="2005-09-14T14:30:05.597-05:00"/>
     <AccessControlList poe="CMOR">
      <AccessControlEntry Permission="READ">
```

```
        <Principal ID="//gNative//$$security/everyoneGroup"
          DisplayName="–everyone –" Name="$$security/everyoneGroup" IsGroup="true"/>
       </AccessControlEntry>
      </AccessControlList>
     </Resource>
    </ResourceSpecifier>
   </getResourceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The getResourceImplementationId operation

Returns the internal identifier for a repository resource. For example, the identifier for an EventCluster can be used when managing jobs from a client.

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 getResourceImplementationId operation.

Table 4-56
Fields for getResourceImplementationId

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Return information

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

Table 4-57
Return Value

Type	Description
ResourceImplementationId	Exposes the underlying identifier for the CMOR object representing a particular version of a particular resource

Java example

The following example retrieves the implementation ID for the resource corresponding to the supplied identification specifier.

```
IdentificationSpecifier is = new IdentificationSpecifier();
ResourceID id = new ResourceID();
id.setValue("0a0a4aac00072ffb000001094fa9f57ea237");
is.setIdentifier(id);
Version vsion = new Version();
vsion.setMarker("2:2006-02-14 18:01:34.257");
is.setVersion(vsion);
ResourceImplementationId implId = stub.getResourceImplementationId(is);
System.out.println("Identification Specifier = " +
  implId.getResourceId().getIdentifier().getValue());
System.out.println("Version = " + implId.getResourceId().getVersion().getMarker());
System.out.println("Implementation ID = " + implID.getId().getValue());
```

SOAP request example

Client invocation of the getResourceImplementationId 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>
  <getResourceImplementationId xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb000001094fa9f57ea237" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <Version marker="2:2006-02-14 18:01:34.257"/>
   </IdentificationSpecifier>
```

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

SOAP response example

The server responds to a getResourceImplementationId 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>
  <getResourceImplementationIdResponse xmlns="http://xml.spss.com/repository/remote">
   <ResourceImplementationId xmlns="http://xml.spss.com/repository">
    <id value="0a0a4aac00072ffb000001094fa9f57ea2d9"/>
    <version marker="0:2006-02-14 18:01:34.32"/>
    <resourceId>
     <identifier value="0a0a4aac00072ffb000001094fa9f57ea237" xsi:type="ResourceID"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
     <Version marker="2:2006-02-14 18:01:34.257"/>
    </resourceId>
   </ResourceImplementationId>
  </getResourceImplementationIdResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The getResourceImplementationIds operation

Returns a set of internal identifiers for repository resources.

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 getResourceImplementationIds operation.

Table 4-58
Fields for getResourceImplementationIds

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Return information

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

Table 4-59
Return Value

Type	Description
ResourceImplementationId[]	Exposes the underlying identifier for the CMOR object representing a particular version of a particular resource

Java example

The following example retrieves the implementation IDs for the resources corresponding to the supplied identification specifiers.

```
IdentificationSpecifier[] is = new IdentificationSpecifier[2];
ResourceID id = new ResourceID();
id.setValue("0a0a4aac00072ffb000001094fa9f57e8028");
is[0].setIdentifier(id);
id.setValue("0a0a4aac00072ffb000001094fa9f57e8035");
is[1].setIdentifier(id);
ResourceImplementationId[] implId = stub.getResourceImplementationIds(is);
for (int i = 0; i < implId.length; i++) {
  System.out.println("Identification Specifier = " +
    implId[i].getResourceId().getIdentifier().getValue());
  System.out.println("Implementation ID = " + implID[i].getId().getValue());
}
```

SOAP request example

Client invocation of the getResourceImplementationIds 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>
   <getResourceImplementationIds xmlns="http://xml.spss.com/repository/remote">
    <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
     <identifier value="0a0a4aac00072ffb000001094fa9f57e8028" xsi:type="ResourceID"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    </IdentificationSpecifier>
    <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
     <identifier value="0a0a4aac00072ffb000001094fa9f57e8035" xsi:type="ResourceID"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    </IdentificationSpecifier>
   </getResourceImplementationIds>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getResourceImplementationIds 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>
  <getResourceImplementationIdsResponse xmlns="http://xml.spss.com/repository/remote">
   <ResourceImplementationId xmlns="http://xml.spss.com/repository">
    <id value="0a0a4aac00072ffb000001094fa9f57e8031"/>
    <version marker="0:2006-02-09 10:47:56.211">
     <label>Production</label>
    </version>
    <resourceId>
     <identifier value="0a0a4aac00072ffb000001094fa9f57e8028" xsi:type="ResourceID"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    </resourceId>
   </ResourceImplementationId>
   <ResourceImplementationId xmlns="http://xml.spss.com/repository">
    <id value="0a0a4aac00072ffb000001094fa9f57e803d"/>
    <version marker="0:2006-02-09 10:47:58.336">
     <label>Production</label>
    </version>
    <resourceId>
     <identifier value="0a0a4aac00072ffb000001094fa9f57e8035" xsi:type="ResourceID"
```

```
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
      </resourceId>
    </ResourceImplementationId>
  </getResourceImplementationIdsResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The getResourceMaintenanceProviders operation

This operation is for internal use only.

Return information

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

Table 4-60
Return Value

Type	Description
ResourceMaintenanceProviderSpecifier[]	Wrapper for a resource maintenance provider. Used to define service interface parameter to preserve polymorphism.

The getResourceReturnSpecific operation

Returns all metadata describing a resource. In contrast, operations such as getChildren or query return 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 identification specifier. If the version information is null, 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 difference between this operation and the getResource operation is the structure of the wrapper for the returned resource. The getResource operation returns a ResourceSpecifier object detailing the resource. Different resource types are handled through the use of XMLSchema instance types. In contrast, the getResourceReturnSpecific operation returns a SpecificResourceSpecifier object. Different resource types are handled through the use of different objects for the types. For example, instead of returning a Resource object of the *File* type, getResourceReturnSpecific returns a File object. Clients that cannot properly handle instance types should use getResourceReturnSpecific instead of getResource.

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 getResourceReturnSpecific operation.

Table 4-61
Fields for getResourceReturnSpecific

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Return information

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

Table 4-62
Return Value

Type	Description
SpecificResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Java example

The following example returns the metadata for a file resource.

```
IdentificationSpecifier id = new IdentificationSpecifier();
ResourceID id = new ResourceID();
id.setValue("0a0a4a35514a6d600000010f3466362980bf");
is.setIdentifier(id);

SpecificResourceSpecifier rs = stub.getResourceReturnSpecific(is);
File f = rs.getFile();
```

SOAP request example

Client invocation of the getResourceReturnSpecific 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>
  <getResourceReturnSpecific xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a35514a6d600000010f3466362980bf"/>
   </IdentificationSpecifier>
  </getResourceReturnSpecific>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a getResourceReturnSpecific 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>
  <getResourceReturnSpecificResponse xmlns="http://xml.spss.com/repository/remote">
   <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
    <File canWrite="true" canDelete="true" canModifyPermissions="true">
     <ResourceID value="0a0a4a35514a6d600000010f3466362980bf"/>
     <Version marker="0:2006-11-29 11:54:51.039"/>
     <CreationDate value="2006-11-29T11:54:51.040-06:00"/>
     <ModificationDate value="2006-11-29T11:54:51.477-06:00"/>
     <Title value="sa"/>
     <ResourcePath value="/sa" hierarchyType="credential"/>
     <CreatedBy value="admin"/>
     <ObjectCreationDate value="2006-11-29T11:54:51.040-06:00"/>
     <ObjectLastModifiedBy value="admin"/>
     <ObjectLastModifiedDate value="2006-11-29T11:54:51.477-06:00"/>
     <AccessControlList poe="CMOR">
      <AccessControlEntry Permission="READ">
       <Principal ID="//gNative//$$security/everyoneGroup" DisplayName="–everyone –"
        Name="$$security/everyoneGroup" IsGroup="true"/>
      </AccessControlEntry>
      <owner ID="//uNative//admin" DisplayName="admin" Name="admin" IsGroup="false"/>
```

```
      </AccessControlList>
      <Author value="admin"/>
      <MimeType value="application/x-vnd.spss-credentials"/>
      <ContentSize value="0"/>
      <ContentLanguage value="Unknown"/>
     </File>
    </SpecificResourceSpecifier>
   </getResourceReturnSpecificResponse>
  </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 4-63
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 identifer *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: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>
   <getResourceSnapshot xmlns="http://xml.spss.com/repository/remote">
     <TransferIdentifier xmlns="http://xml.spss.com/repository">5d947f5e7f</TransferIdentifier>
     <ns2:DeliveryType xmlns:ns2="http://xml.spss.com/repository">MIME</ns2: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 4-64
Fields for getResourceWithLock

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Return information

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

Table 4-65
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">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>
  <getResourceWithLock xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a3548f7ccd200000115fc78fd788062" xsi:type="ResourceID"/>
   </IdentificationSpecifier>
  </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>Native/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="false" canModifyPermissions="false"
      xsi:type="File">
      <ResourceID value="0a0a4a3548f7ccd200000115fc78fd788062"/>
      <Version marker="0:2007-11-01 14:50:51.237"/>
      <CreationDate value="2007-11-01T14:50:51.237-05:00"/>
      <ModificationDate value="2007-11-01T14:50:51.237-05:00"/>
      <Title value="factor_c_09.spv"/>
      <ResourcePath value="/SPV/factor_c_09.spv" hierarchyType="folder"/>
      <CreatedBy value="admin"/>
      <ObjectCreationDate value="2007-11-01T14:50:51.160-05:00"/>
      <ObjectLastModifiedBy value="admin"/>
      <ObjectLastModifiedDate value="2007-11-01T14:50:51.657-05:00"/>
      <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-viewer"/>
      <ContentSize value="260444"/>
     </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 4-66
Fields for getTransferResults

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 4-67
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 cireterion 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.DESCENDING);
```

```
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 applyTransferPolicy operation.

The getTransferStatus operation

Retrieves status information for a resource transfer.

Input fields

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

Table 4-68
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 4-69
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">
   <TransferIdentifier xmlns="http://xml.spss.com/repository">5d947f5e7f</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" xmlns="http://xml.spss.com/repository">
    <currentIdentifier value="/pes_server" hierarchyType="server" xsi:type="ResourcePath"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </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 4-70
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 4-71
Fields for getVersionLabels

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Return information

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

Table 4-72
Return Value

Type	Description
string[]	Version labels for the resource.

Java example

The following function accepts an identifer for a resource. The getVersionLabels operation returns an array of strings containing the labels for this resource.

```
public String[] getVersionLabels(IdentificationSpecifier identificationSpecifier)
  throws IOException, ServiceException {
  ContentRepository repository = getContentRepository();
  String[] result = repository.getVersionLabels(identificationSpecifier);
  return result;
}
```

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">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb000001094fa9f57ec98e" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
  </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"/>
 </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 4-73
Fields for lockResource

Field	Type/Valid Values	Description
TargetIdentificationSpecifier	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.

Java example

The general steps involved in locking a resource include:

1. Create an IdentificationSpecifier object.
2. Create an Identifier object corresponding to the type of information used to identify the object. For example, when using a respository ID, create a ResourceID object. When using the path to the resource in the repository, create a ResourcePath object.
3. Define the identifer information using the setValue method.
4. Assign the identifier to the specifier using the setIdentifier method.

5. Create a Version object.
6. Specify the version information using the setLabel and setMarker methods as needed.
7. Assign the version information to the specifier using the setVersion method.
8. Supply the lockResource operation with the specifier object.

The following sample locks the *Production* version of a specified resource.

```
IdentificationSpecifier identification = new IdentificationSpecifier();

ResourceID resID = new ResourceID("0a0a4a352def327800000115cdabfc0a805a");
identification.setIdentifier(resID);

Version version = new Version();
version.setLabel("Production");
identification.setVersion(version);

stub.lockResource(identification);
```

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">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>
  <lockResource xmlns="http://xml.spss.com/repository/remote">
   <TargetIdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a352def327800000115cdabfc0a805a" xsi:type="ResourceID"/>
    <Version>
     <label>Production</label>
    </Version>
   </TargetIdentificationSpecifier>
  </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 4-74
Fields for lockResources

Field	Type/Valid Values	Description
TargetIdentificationSpecifier	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 moveResource operation

Moves a resource to a specified parent. The identifier for the parent folder must be specified. The resource version can be specified in the resource object. 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.

Table 4-75
Fields for moveResource

Field	Type/Valid Values	Description
TargetParent	TargetParent	Specifies a parent target for a move or copy operation.
Source	Source	Specifies the source for a move or copy operation.

Java example

The following function accepts an array of resources to be moved and a target folder for the move. The getResourceID method returns the identifier for the target folder, which is assigned to a TargetParent object using setIdentifier. The for loop iterates through the resource array, creating a Source object for each resource. The moveResource operation moves the resource corresponding to this object to the parent defined by the TargetParent object.

```
public void moveResources(AdaptableResource[] resources, AdaptableFolder targetFolder)
  throws IOException, ServiceException {
  ContentRepository repository = getContentRepository();
  TargetParent tgtParent = new TargetParent();
  tgtParent.setIdentifier(targetFolder.getResourceID());
  for (int i = 0; i < resources.length; i++) {
    AdaptableResource resource = resources[i];
    Source source = new Source();
    source.setIdentifier(resource.getResourceID());
    repository.moveResource(tgtParent, source);
    targetFolder.setHasChildren(true);
  }
}
```

Use setHasChildren to set the indicator for the presence of children for the target folder to *true* after moving the resources to the folder.

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">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>
  <moveResource xmlns="http://xml.spss.com/repository/remote">
   <TargetParent xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106561fccf88047" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </TargetParent>
   <Source xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106a7a10b13803c" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </Source>
  </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 promoteResource operation

Transfers resources between two different IBM® SPSS® Collaboration and Deployment Services Repository instances in accordance with a set of predefined rules. For more information, see the topic Promotion in Chapter 3 on p. 16.

Supply this operation with a specifier object that identifies the following:

- the resource to promote
- the promotion policy

The promotion policy defines the items included with the specified resource in the promotion. Create a promotion policy by using the createResource operation.For more information, see the topic Promotion Policies in Chapter 3 on p. 16.

Input fields

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

Table 4-76
Fields for promoteResource

Field	Type/Valid Values	Description
PromotionSpecifier	PromotionSpecifier	

Return information

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

Table 4-77
Return Value

Type	Description
TransferIdentifier	Unique identifier of the resource transfer.

Java example

The general process for promoting resources involves the following steps:

1. Initiate the transfer using the promoteResource operation.
2. Monitor the status of the transfer using the getTransferStatus operation.
3. If you are promoting the source version to a file for subsequent transfer at a later time, get the promotion results as an attachment using the getResourceSnapshot operation.
4. Release system resources used for the promotion using the disposeTransfer operation.

To promote a version:

1. Create a PromotionSpecifier object to define the promotion policy and the resources being promoted.
2. Create an Identifier object of the ResourceURI type to identify the promotion policy to use. Supply the setValue method of the promotion policy identifier object with the URI for the promotion policy.
3. Use the setPromotionPolicyIdentifier method to assign the policy identifier to the specifier object.
4. Create a ResourceURI object for the source being promoted. Supply the setValue method of the URI object with the URI for the source.
5. Create an IdentifierList object. Supply the addIdentifier method of the list object with the URI object for the source.
6. Use the setIdentifierList method to assign the source list to the specifier object.
7. Provide the promoteResource operation with the promotion specifier. The returned TransferIdentifier value-object can be used to monitor the current status of the transfer asynchronously.

The following code sample promotes the latest version of a file.

```
PromotionSpecifier promotionSpecifier = new PromotionSpecifier();
Identifier promotionPolicyIdentifier = new ResourceURI();
promotionPolicyIdentifier.setValue("spsscr:///?id=0923cb338743d301000001319601c2498574");
promotionSpecifier.setPromotionPolicyIdentifier(promotionPolicyIdentifier);

ResourceURI resourceURI = new ResourceURI();
resourceURI.setValue("spsscr:///?id=0923cb33df9e441400000130520bd6a5b952#I.LATEST");

IdentifierList identifierList = new IdentifierList();
identifierList.addIdentifier(resourceURI);
promotionSpecifier.setIdentifierList(identifierList);

TransferIdentifier transferIdentifier = stub.promoteResource(promotionSpecifier);
```

The following sample code monitors the promotion status, releasing system resources when the process completes.

```
TransferStatus transferStatus = null;
TransferState transferState = null;
try {
   transferStatus = stub.getTransferStatus(transferIdentifier);
   transferState = transferStatus.getTransferState();
   while (!isTransferEnded(transferState)) {
      Identifier identifier = transferStatus.getCurrentIdentifier();
      if (identifier != null) {
         System.out.println("" + transferStatus.getCurrentProcess() +
    " " + identifier.getValue());
      } else {
         System.out.println("Preparing to promote resource...");
      }
      Thread.sleep(1000);
      transferStatus = stub.getTransferStatus(transferIdentifier);
      transferState = transferStatus.getTransferState();
   }
} catch (RepositoryException repositoryException) {
  // handle transfer failure
}

if (!transferState.equals(TransferState.COMPLETED)) {
   throw new IllegalStateException("Something went terribly wrong...");
}

stub.disposeTransfer(transferIdentifier);
```

The isTransferEnded function used to monitor the process simply returns the state of the transfer.

```
private static boolean isTransferEnded(TransferState inTransferState) {
  return inTransferState.equals(TransferState.CANCELED) ||
   inTransferState.equals(TransferState.COMPLETED) ||
   inTransferState.equals(TransferState.FAILED);
}
```

SOAP request example

Client invocation of the promoteResource 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>
  <promoteResource xmlns="http://xml.spss.com/repository/remote">
   <PromotionSpecifier xmlns="http://xml.spss.com/repository">
    <promotionPolicyIdentifier value="spsscr:///?id=0923cb338743d3010000013196 01c2498574" xsi:type="ResourceURI"/>
    <IdentifierList>
     <identifier value="spsscr:///?id=0923cb33df9e441400000130520bd6a5b952#l.LATEST" xsi:type="ResourceURI"/>
    </IdentifierList>
   </PromotionSpecifier>
  </promoteResource>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a promoteResource 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>
  <promoteResourceResponse xmlns="http://xml.spss.com/repository/remote">
   <TransferIdentifier engineVersion="2.0" serviceEndpoint="http://cdsserver:8080/cr-ws/services/ContentRepository"
    xmlns="http://xml.spss.com/repository">0923cb338743d3010000013196 01c2498583
   </TransferIdentifier>
  </promoteResourceResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The query operation

This operation is for internal use only and has been deprecated. Consumers of the web service wishing 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 desired version. 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 4-78
Fields for query

Field	Type/Valid Values	Description
QuerySpecification	QuerySpecification	Expresses search criteria for querying the repsoitory.

Return information

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

Table 4-79
Return Value

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

Java example

The following example queries the repository for file resources in the */Models/Trees* that are authored by Steve or Lance. The resources returned by the query contain only the resource path information.

```
ResourcePath treemodelsPath = new ResourcePath();
treemodelsPath.setValue("/Models/Trees");
ConditionalTerm pathIsTreeModel = new ConditionalTerm();
pathIsTreeModel.setMetadata(treemodelsPath);
pathIsTreeModel.setCondition(QueryCondition.EQUAL);
Author steve = new Author();
steve.setValue("Steve");
ConditionalTerm authorIsSteve = new ConditionalTerm();
authorIsSteve.setMetadata(steve);
authorIsSteve.setCondition(QueryCondition.EQUAL);
Author lance = new Author();
lance.setValue("Lance");
ConditionalTerm authorIsLance = new ConditionalTerm();
authorIsLance.setMetadata(lance);
authorIsLance.setCondition(QueryCondition.EQUAL);
NestedTerm authorIsLanceOrSteve = new NestedTerm();
authorIsLanceOrSteve.addTerm(authorIsSteve);
authorIsLanceOrSteve.addTerm(new OrPredicateTerm());
authorIsLanceOrSteve.addTerm(authorIsLance);
QuerySpecification q = new QuerySpecification();
q.setScope("file");
ResourcePath p = new ResourcePath();
SelectedMetadata selected = new SelectedMetadata();
selected.addMetadataBase(p);
q.setSelectedMetadata(selected);
q.addTerm(pathIsTreeModel);
q.addTerm(new AndPredicateTerm());
q.addTerm(authorIsLanceOrSteve);
Resource[] resources = c_repository.query(q);
```

The queryReturnSpecific operation

The queryReturnSpecific operation provides a way to search for and retrieve resources from the repository. This operation is for internal use only and has been deprecated. Consumers of the web service wishing to search for resources should use the Search service instead of this operation.

The difference between this operation and the query operation is the structure of the wrapper for the returned resources. The query operation returns an array of ResourceSpecifier objects detailing the resources. Different resource types are handled through the use of XMLSchema instance types. In contrast, the queryReturnSpecific operation returns an array of SpecificResourceSpecifier objects. Different resource types are handled through the use of different objects for the types. For example, instead of returning a Resource object of the *File* type, queryReturnSpecific returns a File object. Clients that cannot properly handle instance types should use queryReturnSpecific instead of query.

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 queryReturnSpecific operation.

Table 4-80
Fields for queryReturnSpecific

Field	Type/Valid Values	Description
QuerySpecification	QuerySpecification	Expresses search criteria for querying the repsoitory.

Return information

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

Table 4-81
Return Value

Type	Description
SpecificResourceSpecifier[]	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.

Input fields

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

Table 4-82
Fields for removeLabel

Field	Type/Valid Values	Description
TargetIdentificationSpecifier	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.
label	string	The label to be removed.

Return information

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

Table 4-83
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.

Java example

The following function accepts a file resource and an array of labels to be removed from the resource. The getResourceID method returns the identifier for the file, which is assigned to an IndentificationSpecifier object using setIdentifier. Similarly, the getVersion method returns a Version object denoting the file version. The setVersion method uses this information to define the version for the IdentificationSpecifier object. Each label is removed by looping over the label array, using removeLabel for each entry in the array.

```
public void removeVersionLabel(AdaptableFile fileVersion, String[] labels)
  throws ResourceNotFoundException, RepositoryException,
    RemoteException, IOException, ServiceException {
  IdentificationSpecifier specifier = new IdentificationSpecifier();
  specifier.setIdentifier(fileVersion.getResourceID());
  Version version = fileVersion.getResource().getVersion();
  specifier.setVersion(version);
  LinkedHashSet set = new LinkedHashSet(Arrays.asList(version.getLabel()));
  for (int i=0; i < labels.length; ++i) {
    getContentRepository().removeLabel(specifier, labels[i]);
    set.remove(labels[i]);
  }
  String[] newLabels = (String[]) set.toArray(new String[set.size()]);
  fileVersion.getVersion().setLabel(newLabels);
```

```
  fireUpdateEvent(fileVersion);
}
```

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">
   <TargetIdentificationSpecifier  xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb000001094fa9f57ecd59" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <Version marker="0:2006-02-21 17:19:50.796"/>
   </TargetIdentificationSpecifier>
   <label>Test</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">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb000001094fa9f57ecd59" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <Version marker="0:2006-02-21 17:19:50.796"/>
```

```
    </IdentificationSpecifier>
   </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 4-84
Fields for runCqlQuery

Field	Type/Valid Values	Description
QueryCqlSpecification	QueryCqlSpecification	Expresses search criteria for querying the repsoitory.

Return information

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

Table 4-85
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 4-86
Fields for setBulkResourceMetadata

Field	Type/Valid Values	Description
IdentificationSpecifier	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.

Field	Type/Valid Values	Description
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

Return information

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

Table 4-87
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.

Java example

To define resource metadata in bulk:

1. Create an array of IdentificationSpecifier objects for the resources. Set the identification and version information for each resource.
2. Create a ResourceSpecifier object to contain the metadata values for the resources and set the values.
3. Create a SelectedMetadata object to identify the metadata field corresponding to the values defined for the resources.
4. Supply the setBulkResourceMetadata operation with the identification, resource, and metadata objects.

   The following code sets the expiration date for two files.

```
IdentificationSpecifier[] is = new IdentificationSpecifier[2];
ResourceID id = new ResourceID();
id.setValue("0a0a4a35bad1b0750000010f3a50fe4980e6");
is[0].setIdentifier(id);
Version vsion = new Version();
vsion.setMarker("0:2006-11-30 14:19:20.769");
is[0].setVersion(vsion);
id.setValue("0a0a4a353da969fa0000010ef0fcc01f8649");
is[1].setIdentifier(id);
vsion.setMarker("0:2006-11-17 09:59:42.666");
is[1].setVersion(vsion);

ResourceSpecifier rs = new ResourceSpecifier;
File file = new File();
```

```
ExpirationDate exp = new ExpirationDate();
Calendar date = new Calendar();
date.set(2007, 01, 31, 0, 0, 0);
exp.setValue(date);
file.setExpirationDate(exp);
rs.setResource(file);

SelectedMetadata sm = new SelectedMetadata();
sm.addMetadataBase(new ExpirationDate());

IdentificationSpecifier[] idSpec = stub.setBulkResourceMetadata(is, rs, sm);
```

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: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>
  <setBulkResourceMetadata xmlns="http://xml.spss.com/repository/remote">
   <ns2:IdentificationSpecifier xmlns:ns2="http://xml.spss.com/repository">
    <ns2:identifier value="0a0a4a35bad1b0750000010f3a50fe4980e6" xsi:type="ResourceID"/>
    <ns2:Version marker="0:2006-11-30 14:19:20.769"/>
   </ns2:IdentificationSpecifier>
   <ns3:IdentificationSpecifier xmlns:ns3="http://xml.spss.com/repository">
    <ns3:identifier value="0a0a4a353da969fa0000010ef0fcc01f8649" xsi:type="ResourceID"/>
    <ns3:Version marker="0:2006-11-17 09:59:42.666"> </ns3:Version>
   </ns3:IdentificationSpecifier>
   <ns4:ResourceSpecifier xmlns:ns4="http://xml.spss.com/repository">
    <ns4:Resource xsi:type="File">
     <ns4:ExpirationDate value="2007-01-31T00:00:00.000-06:00"/>
    </ns4:Resource>
   </ns4:ResourceSpecifier>
   <SelectedMetadata xmlns="http://xml.spss.com/repository">
    <ExpirationDate/>
   </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">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a35bad1b0750000010f3a50fe4980e6" xsi:type="ResourceID"/>
   </IdentificationSpecifier>
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a353da969fa0000010ef0fcc01f8649" xsi:type="ResourceID"/>
   </IdentificationSpecifier>
  </setBulkResourceMetadataResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The setBulkResourceMetadataSpecific 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.

The difference between this operation and the setBulkResourceMetadata operation is the structure of the wrapper for the resource input. The setBulkResourceMetadata operation accepts an array of ResourceSpecifier objects defining the resources. Different resource types are handled through the use of XMLSchema instance types. In contrast, the setBulkResourceMetadataSpecific operation accepts an array of SpecificResourceSpecifier objects. Different resource types are handled through the use of different objects for the types. For example, instead of handling a Resource object of the *File* type, setBulkResourceMetadataSpecific processes a File object. Clients that cannot properly handle instance types should use setBulkResourceMetadataSpecific instead of setBulkResourceMetadata.

Input fields

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

Table 4-88
Fields for setBulkResourceMetadataSpecific

Field	Type/Valid Values	Description
TargetIdentificationSpecifier	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.
SpecificResourceSpecifier	SpecificResourceSpecifier	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

Return information

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

Table 4-89
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.

Java example

To define resource metadata in bulk:

1. Create an array of IdentificationSpecifier objects for the resources. Set the identification and version information for each resource.
2. Create a SpecificResourceSpecifier object to contain the metadata values for the resources and set the values.
3. Create a SelectedMetadata object to identify the metadata field corresponding to the values defined for the resources.
4. Supply the setBulkResourceMetadataSpecific operation with the identification, resource, and metadata objects.

   The following code sets the expiration date for two files.

```
IdentificationSpecifier[] is = new IdentificationSpecifier[2];
ResourceID id = new ResourceID();
id.setValue("0a0a4a35bad1b0750000010f3a50fe4980e6");
is[0].setIdentifier(id);
Version vsion = new Version();
```

```
vsion.setMarker("0:2006-11-30 14:19:20.769");
is[0].setVersion(vsion);
id.setValue("0a0a4a353da969fa0000010ef0fcc01f8649");
is[1].setIdentifier(id);
vsion.setMarker("0:2006-11-17 09:59:42.666");
is[1].setVersion(vsion);

SpecificResourceSpecifier rs = new SpecificResourceSpecifier;
File file = new File();
ExpirationDate exp = new ExpirationDate();
Calendar date = new Calendar();
date.set(2007, 01, 31, 0, 0, 0);
exp.setValue(date);
file.setExpirationDate(exp);
rs.setFile(file);

SelectedMetadata sm = new SelectedMetadata();
sm.addMetadataBase(new ExpirationDate());

IdentificationSpecifier[] idSpec = stub.setBulkResourceMetadataSpecific(is, rs, sm);
```

SOAP request example

Client invocation of the setBulkResourceMetadataSpecific 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>
  <setBulkResourceMetadataSpecific xmlns="http://xml.spss.com/repository/remote">
   <ns2:IdentificationSpecifier xmlns:ns2="http://xml.spss.com/repository">
    <ns2:identifier value="0a0a4a35bad1b0750000010f3a50fe4980e6" xsi:type="ResourceID"/>
    <ns2:Version marker="0:2006-11-30 14:19:20.769"/>
   </ns2:IdentificationSpecifier>
   <ns3:IdentificationSpecifier xmlns:ns3="http://xml.spss.com/repository">
    <ns3:identifier value="0a0a4a353da969fa0000010ef0fcc01f8649" xsi:type="ResourceID"/>
    <ns3:Version marker="0:2006-11-17 09:59:42.666"> </ns3:Version>
   </ns3:IdentificationSpecifier>
   <ns4:SpecificResourceSpecifier xmlns:ns4="http://xml.spss.com/repository">
```

```
      <ns4:File>
        <ns4:ExpirationDate value="2007-01-31T00:00:00.000-06:00"/>
      </ns4:File>
    </ns4:SpecificResourceSpecifier>
    <SelectedMetadata xmlns="http://xml.spss.com/repository">
      <ExpirationDate/>
    </SelectedMetadata>
  </setBulkResourceMetadataSpecific>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a setBulkResourceMetadataSpecific 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>
  <setBulkResourceMetadataSpecificResponse xmlns="http://xml.spss.com/repository/remote">
    <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
      <identifier value="0a0a4a35bad1b0750000010f3a50fe4980e6" xsi:type="ResourceID"/>
    </IdentificationSpecifier>
    <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
      <identifier value="0a0a4a353da969fa0000010ef0fcc01f8649" xsi:type="ResourceID"/>
    </IdentificationSpecifier>
  </setBulkResourceMetadataSpecificResponse>
 </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 4-90
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 4-91
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 getTransferStatus operation.
3. When exporting, get the export results as an attachment using the getResourceSnapshot operation.
4. Release system resources used for the transfer using the disposeTransfer 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 code sample below 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 code sample below 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">
   <TransferSpecification xmlns="http://xml.spss.com/repository">
```

```
      <TransferSource>
       <ResourcePath value="/ModelerStreamLibrary/Data Understanding" hierarchyType="folder"/>
      </TransferSource>
      <TransferTarget/>
     </TransferSpecification>
    </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 xmlns="http://xml.spss.com/repository">5d947f5e7f</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 4-92
Fields for unlockResource

Field	Type/Valid Values	Description
TargetIdentificationSpecifier	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.

Java example

The general steps involved in unlocking a resource include:

1. Create an IdentificationSpecifier object.

2. Create an Identifier object corresponding to the type of information used to identify the object. For example, when using a respository ID, create a ResourceID object. When using the path to the resource in the repository, create a ResourcePath object.
3. Define the identifer information using the setValue method.
4. Assign the identifier to the specifier using the setIdentifier method.
5. Create a Version object.
6. Specify the version information using the setLabel and setMarker methods as needed.
7. Assign the version information to the specifier using the setVersion method.
8. Supply the unlockResource operation with the specifier object.

   The following sample unlocks the *Production* version of a specified resource.

```
IdentificationSpecifier identification = new IdentificationSpecifier();

ResourceID resID = new ResourceID("0a0a4a352def327800000115cdabfc0a805a");
identification.setIdentifier(resID);

Version version = new Version();
version.setLabel("Production");
identification.setVersion(version);

stub.unlockResource(identification);
```

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">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>
  <unlockResource xmlns="http://xml.spss.com/repository/remote">
   <TargetIdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a352def327800000115cdabfc0a805a" xsi:type="ResourceID"/>
```

```
      <Version>
       <label>Production</label>
      </Version>
     </TargetIdentificationSpecifier>
   </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 4-93
Fields for unlockResources

Field	Type/Valid Values	Description
TargetIdentificationSpecifier	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.

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">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>
 <unlockResources xmlns="http://xml.spss.com/repository/remote">
  <TargetIdentificationSpecifier xmlns="http://xml.spss.com/repository">
   <identifier value="0a0a4a352def327800000115cdabfc0a805a" xsi:type="ResourceID"/>
   <Version>
    <label>Production</label>
   </Version>
  </TargetIdentificationSpecifier>
  <TargetIdentificationSpecifier xmlns="http://xml.spss.com/repository">
   <identifier value="0a0a4a352def327800000115cdabfc0a805a" xsi:type="ResourceID"/>
   <Version>
    <label>Production</label>
   </Version>
  </TargetIdentificationSpecifier>
 </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 4-94
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 4-95
Return Value

Type	Description
string	The identifier of the updated custom property.

Java example

The following function accepts a CustomProperty object containing an updated property definition. The updateCustomProperty operation uses this information to update the corresponding property in the repository.

```
public String udpateCustomProperty(CustomProperty property)
  throws IOException, ServiceException,RepositoryException {
  ContentRepository repository = getContentRepository();
  String id = repository.updateCustomProperty(property);
  return id;
}
```

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="Reviewed by" 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>
```

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 updateCustomPropertyValues operation

Updates the values for custom properties defined for a resource.

Input fields

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

Table 4-96
Fields for updateCustomPropertyValues

Field	Type/Valid Values	Description
IdentificationSpecifierIn	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.
CustomPropertyValue	CustomPropertyValue[]	The value assigned by a user to an object for a particular custom property.
ResourceSpecifier	ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Return information

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

Table 4-97
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.

Java example

To update the value for a custom property:

1. Create an IdentificationSpecifier object for the resource to be updated. Set the identification and version information for the resource.
2. Retrieve the existing property values for the resource using the getCustomPropertyValues operation.
3. Update the values as needed.
4. Create a ResourceSpecifier object for the resource being updated and specify the properties for the resource.
5. Supply the updateCustomPropertyValues operation with the identification specifier, the updated custom property values, and the resource specifier.

   The following code updates the values for a single-select custom property for a file.

```
IdentificationSpecifier is = new IdentificationSpecifier;
Identifier id = new Identifier();
id.setValue("0a0a4aac00072ffb00000106561fccf88148");
```

```
is.setIdentifier(id);

CustomPropertyValue[] val = stub.getCustomPropertyValues(id);
SelectionValue[] selVal = new SelectionValue[3];
selVal[0].setValue("Richard");
selVal[0].setIsSelected(false);
selVal[1].setValue("Patrick");
selVal[1].setIsSelected(true);
selVal[2].setValue("John");
selVal[2].setIsSelected(false);
val[0].setSelect(selVal);

ResourceSpecifier rs = new ResourceSpecifier;
File file = new File();
id.setValue("0a0a4aac00072ffb00000106561fccf88148");
file.setResourceID(id);
ResourcePath rp = new ResourcePath();
rp.setValue("/ModelerStreamLibrary/Data Understanding/E02_ExploreDataQuality.str");
rp.setHierarchyType(HierarchyType.FOLDER);
file.setResourcePath(rp);
rs.setResource(file);

IdentificationSpecifier id = stub.updateCustomPropertyValues(id, val, rs);
```

SOAP request example

Client invocation of the updateCustomPropertyValues 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>
  <updateCustomPropertyValues xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106561fccf88148" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
   <CustomPropertyValue label="Reviewed by" identifier="0a0a4aac00072ffb00000106f3f7b05b3856"
    xmlns="http://xml.spss.com/repository">
```

```
      <select multipleSelect="false">
       <selectionValue isSelected="false" value="Richard"/>
       <selectionValue isSelected="true" value="Patrick"/>
       <selectionValue isSelected="false" value="John"/>
      </select>
     </CustomPropertyValue>
     <ResourceSpecifier xmlns="http://xml.spss.com/repository">
      <Resource xsi:type="File" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
       <ResourceID value="0a0a4aac00072ffb00000106561fccf88148"/>
       <ResourcePath value="/ModelerStreamLibrary/Data Understanding/E02_ExploreDataQuality.str"
        hierarchyType="folder"/>
      </Resource>
     </ResourceSpecifier>
    </updateCustomPropertyValues>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a updateCustomPropertyValues 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>
  <updateCustomPropertyValuesResponse xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106561fccf88148" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
  </updateCustomPropertyValuesResponse>
 </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 4-98
Fields for updateCustomPropertyValuesInBulk

Field	Type/Valid Values	Description
QualifiedCustomPropertyValuesSpecifier[]	QualifiedCustomPropertyValuesSpecifier	Wrapper for an indentification specifier, resource specifier and its custom properties

Return information

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

Table 4-99
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.

Java example

To update custom properties for multiple resources:

1. Create an array of IdentificationSpecifier objects for the resources to be updated. Set the identification and version information for each resource.
2. Create an array of ResourceSpecifier objects to contain the updated information for the resources. Set the properties for each resource.
3. Retrieve the existing custom properties for the resources using the getCustomPropertyValues operation.
4. Update the custom property values as desired.
5. Create an array of QualifiedCustomPropertyValuesSpecifier objects for the custom properties to be updated. Set the identification, resource, and new custom property for each item in the array.
6. Supply the updateCustomPropertyValuesInBulk operation with the QualifiedCustomPropertyValuesSpecifier array.

   The following code updates a boolean property for two files.

```
IdentificationSpecifier[] is = new IdentificationSpecifier[2];
ResourceID id = new ResourceID();
id.setValue("0a0a4a35c72b39630000010f588e773d8046");
is[0].setIdentifier(id);
Version vsion = new Version();
vsion.setMarker("0:2006-12-06 10:30:13.338");
is[0].setVersion(vsion);

ResourceSpecifier[] rs = new ResourceSpecifier[2];
File file = new File();
id.setValue("0a0a4a35c72b39630000010f588e773d8046");
file.setResourceID(id);
ResourcePath rp = new ResourcePath();
rp.setValue("/StatisticsLibrary/tree_model.sps");
rp.setHierarchyType(HierarchyType.FOLDER);
file.setResourcePath(rp);
rs[0].setResource(file);
```

```
id.setValue("0a0a4a35c72b39630000010f588e773d8108");
is[1].setIdentifier(id);
vsion.setMarker("0:2006-12-06 11:10:17.643");
is[1].setVersion(vsion);

file.setResourceID(id);
rp.setValue("/SASSyntaxLibrary/frequencies.sas");
rp.setHierarchyType(HierarchyType.FOLDER);
file.setResourcePath(rp);
rs[1].setResource(file);

CustomPropertyValue[] cpv = new CustomPropertyValue[2];
FreeformValue ffv = new FreeformValue();
ffv.setBooleanValue(true);
for (int i = 0; i < cpv.length; i++)
  cpv[i] = stub.getCustomPropertyValues(is[i]);
  cpv[i].setFreeform(ffv);
}

QualifiedCustomPropertyValuesSpecifier[] cpvSpec =
  new QualifiedCustomPropertyValuesSpecifier[2];
for (int i = 0; i < cpvSpec.length; i++)
  cpvSpec[i].setIdentificationSpecifier(is[i]);
  cpvSpec[i].setResourceSpecifier(rs[i]);
  cpvSpec[i].setCustomPropertyValue(cpv[i]);
}

IdentificationSpecifier[] idSpec = stub.updateCustomPropertyValuesInBulk(cpvSpec);
```

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">
```

```
    <ns2:QualifiedCustomPropertyValuesSpecifier xmlns:ns2="http://xml.spss.com/repository">
     <ns2:IdentificationSpecifier>
      <ns2:identifier value="0a0a4a35c72b39630000010f588e773d8046" xsi:type="ResourceID"/>
      <ns2:Version marker="0:2006-12-06 10:30:13.338"/>
     </ns2:IdentificationSpecifier>
     <ns2:ResourceSpecifier>
      <ns2:Resource xsi:type="File">
       <ns2:ResourceID value="0a0a4a35c72b39630000010f588e773d8046"/>
       <ns2:ResourcePath value="/StatisticsLibrary/tree_model.sps" hierarchyType="folder"/>
       <ns2:associatedTopicList/>
      </ns2:Resource>
     </ns2:ResourceSpecifier>
     <ns2:CustomPropertyValue label="Reviewed" identifier="0a0a4a3567b4a2cc0000010fed740baa800b">
      <ns2:freeform type="boolean">
       <ns2:booleanValue>true</ns2:booleanValue>
      </ns2:freeform>
     </ns2:CustomPropertyValue>
    </ns2:QualifiedCustomPropertyValuesSpecifier>
    <ns3:QualifiedCustomPropertyValuesSpecifier xmlns:ns3="http://xml.spss.com/repository">
     <ns3:IdentificationSpecifier>
      <ns3:identifier value="0a0a4a35c72b39630000010f588e773d8108" xsi:type="ResourceID"/>
      <ns3:Version marker="0:2006-12-06 11:10:17.643"/>
     </ns3:IdentificationSpecifier>
     <ns3:ResourceSpecifier>
      <ns3:Resource xsi:type="File">
       <ns3:ResourceID value="0a0a4a35c72b39630000010f588e773d8108"/>
       <ns3:ResourcePath value="/SASSyntaxLibrary/frequencies.sas" hierarchyType="folder"/>
       <ns3:associatedTopicList/>
      </ns3:Resource>
     </ns3:ResourceSpecifier>
     <ns3:CustomPropertyValue label="Reviewed" identifier="0a0a4a3567b4a2cc0000010fed740baa800b">
      <ns3:freeform type="boolean">
       <ns3:booleanValue>true</ns3:booleanValue>
      </ns3:freeform>
     </ns3:CustomPropertyValue>
    </ns3:QualifiedCustomPropertyValuesSpecifier>
   </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">
```

```
    <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
      <identifier value="0a0a4a35c72b39630000010f588e773d8046" xsi:type="ResourceID"/>
    </IdentificationSpecifier>
    <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
      <identifier value="0a0a4a35c72b39630000010f588e773d8108" xsi:type="ResourceID"/>
    </IdentificationSpecifier>
  </updateCustomPropertyValuesInBulkResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The updateCustomPropertyValuesInBulkSpecific operation

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

The difference between this operation and the updateCustomPropertyValuesInBulk operation is the structure of the wrapper for the resource input. The updateCustomPropertyValuesInBulk operation accepts an array of QualifiedCustomPropertyValuesSpecifier objects defining the resources. Different resource types are handled through the use of XMLSchema instance types. In contrast, the updateCustomPropertyValuesInBulkSpecific operation accepts an array of QualifiedSpecificCustomPropertyValuesSpecifier objects defining the resources. Different resource types are handled through the use of different objects for the types. For example, instead of handling a Resource object of the *File* type, updateCustomPropertyValuesInBulkSpecific processes a File object. Clients that cannot properly handle instance types should use updateCustomPropertyValuesInBulkSpecific instead of updateCustomPropertyValuesInBulk.

Input fields

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

Table 4-100
Fields for updateCustomPropertyValuesInBulkSpecific

Field	Type/Valid Values	Description
QualifiedSpecificCustomPropertyValuesSpecifier[]	QualifiedSpecificCustomPropertyValuesSpecifier[]	Wrapper for an identification specifier, resource specifier and its custom properties

Return information

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

Table 4-101
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.

Java example

To update custom properties for multiple resources:

1. Create an array of IdentificationSpecifier objects for the resources to be updated. Set the identification and version information for each resource.
2. Create an array of SpecificResourceSpecifier objects to contain the updated information for the resources. Set the properties for each resource.
3. Retrieve the existing custom properties for the resources using the getCustomPropertyValues operation.
4. Update the custom property values as desired.
5. Create an array of QualifiedSpecificCustomPropertyValuesSpecifier objects for the custom properties to be updated. Set the identification, resource, and new custom property for each item in the array.
6. Supply the updateCustomPropertyValuesInBulkSpecific operation with the QualifiedCustomPropertyValuesSpecifier array.

   The following code updates a boolean property for two files.

```
IdentificationSpecifier[] is = new IdentificationSpecifier[2];
ResourceID id = new ResourceID();
id.setValue("0a0a4a35c72b39630000010f588e773d8046");
is[0].setIdentifier(id);
Version vsion = new Version();
vsion.setMarker("0:2006-12-06 10:30:13.338");
is[0].setVersion(vsion);

SpecificResourceSpecifier[] rs = new SpecificResourceSpecifier[2];
File file = new File();
id.setValue("0a0a4a35c72b39630000010f588e773d8046");
file.setResourceID(id);
ResourcePath rp = new ResourcePath();
rp.setValue("/StatisticsLibrary/tree_model.sps");
rp.setHierarchyType(HierarchyType.FOLDER);
file.setResourcePath(rp);
rs[0].setFile(file);

id.setValue("0a0a4a35c72b39630000010f588e773d8108");
is[1].setIdentifier(id);
vsion.setMarker("0:2006-12-06 11:10:17.643");
is[1].setVersion(vsion);

file.setResourceID(id);
rp.setValue("/SASSyntaxLibrary/frequencies.sas");
rp.setHierarchyType(HierarchyType.FOLDER);
file.setResourcePath(rp);
rs[1].setFile(file);

CustomPropertyValue[] cpv = new CustomPropertyValue[2];
```

```
FreeformValue ffv = new FreeformValue();
ffv.setBooleanValue(true);
for (int i = 0; i < cpv.length; i++)
  cpv[i] = stub.getCustomPropertyValues(is[i]);
  cpv[i].setFreeform(ffv);
}

QualifiedSpecificCustomPropertyValuesSpecifier[] cpvSpec =
  new QualifiedSpecificCustomPropertyValuesSpecifier[2];
for (int i = 0; i < cpvSpec.length; i++)
  cpvSpec[i].setIdentificationSpecifier(is[i]);
  cpvSpec[i].setResourceSpecifier(rs[i]);
  cpvSpec[i].setCustomPropertyValue(cpv[i]);
}

IdentificationSpecifier[] idSpec = stub.updateCustomPropertyValuesInBulkSpecific(cpvSpec);
```

SOAP request example

Client invocation of the updateCustomPropertyValuesInBulkSpecific 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>
  <updateCustomPropertyValuesInBulkSpecific xmlns="http://xml.spss.com/repository/remote">
   <ns2:QualifiedSpecificCustomPropertyValuesSpecifier xmlns:ns2="http://xml.spss.com/repository">
    <ns2:IdentificationSpecifier>
     <ns2:identifier value="0a0a4a35c72b39630000010f588e773d8046" xsi:type="ResourceID"/>
     <ns2:Version marker="0:2006-12-06 10:30:13.338"/>
    </ns2:IdentificationSpecifier>
    <ns2:SpecificResourceSpecifier>
     <ns2:File>
      <ns2:ResourceID value="0a0a4a35c72b39630000010f588e773d8046"/>
      <ns2:ResourcePath value="/StatisticsLibrary/tree_model.sps" hierarchyType="folder"/>
      <ns2:associatedTopicList/>
     </ns2:File>
    </ns2:SpecificResourceSpecifier>
    <ns2:CustomPropertyValue label="Reviewed" identifier="0a0a4a3567b4a2cc0000010fed740baa800b">
```

```
      <ns2:freeform type="boolean">
       <ns2:booleanValue>false</ns2:booleanValue>
      </ns2:freeform>
     </ns2:CustomPropertyValue>
    </ns2:QualifiedSpecificCustomPropertyValuesSpecifier>
    <ns3:QualifiedSpecificCustomPropertyValuesSpecifier xmlns:ns3="http://xml.spss.com/repository">
     <ns3:IdentificationSpecifier>
      <ns3:identifier value="0a0a4a35c72b39630000010f588e773d8108" xsi:type="ResourceID"/>
      <ns3:Version marker="0:2006-12-06 11:10:17.643"/>
     </ns3:IdentificationSpecifier>
     <ns3:SpecificResourceSpecifier>
      <ns3:File>
       <ns3:ResourceID value="0a0a4a35c72b39630000010f588e773d8108"/>
       <ns3:ResourcePath value="/SASSyntaxLibrary/frequencies.sas" hierarchyType="folder"/>
       <ns3:associatedTopicList/>
      </ns3:File>
     </ns3:SpecificResourceSpecifier>
     <ns3:CustomPropertyValue label="Reviewed" identifier="0a0a4a3567b4a2cc0000010fed740baa800b">
      <ns3:freeform type="boolean">
       <ns3:booleanValue>false</ns3:booleanValue>
      </ns3:freeform>
     </ns3:CustomPropertyValue>
    </ns3:QualifiedSpecificCustomPropertyValuesSpecifier>
   </updateCustomPropertyValuesInBulkSpecific>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a updateCustomPropertyValuesInBulkSpecific 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>
  <updateCustomPropertyValuesInBulkSpecificResponse xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a35c72b39630000010f588e773d8046" xsi:type="ResourceID"/>
   </IdentificationSpecifier>
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a35c72b39630000010f588e773d8108" xsi:type="ResourceID"/>
   </IdentificationSpecifier>
  </updateCustomPropertyValuesInBulkSpecificResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The updateCustomPropertyValuesSpecific operation

Updates the values for custom properties defined for a resource.

The difference between this operation and the updateCustomPropertyValues operation is the structure of the wrapper for the resource input. The updateCustomPropertyValues operation accepts a ResourceSpecifier object defining the resource. Different resource types are handled through the use of XMLSchema instance types. In contrast, the updateCustomPropertyValuesSpecific operation accepts a SpecificResourceSpecifier object defining the resource. Different resource types are handled through the use of different objects for the types. For example, instead of handling a Resource object of the *File* type, updateCustomPropertyValuesSpecific processes a File object. Clients that cannot properly handle instance types should use updateCustomPropertyValuesSpecific instead of updateCustomPropertyValues.

Input fields

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

Table 4-102
Fields for updateCustomPropertyValuesSpecific

Field	Type/Valid Values	Description
IdentificationSpecifierIn	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.
CustomPropertyValue	CustomPropertyValue[]	The value assigned by a user to an object for a particular custom property.
SpecificResourceSpecifier	SpecificResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Return information

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

Table 4-103
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.

Java example

To update the value for a custom property:

1. Create an IdentificationSpecifier object for the resource to be updated. Set the identification and version information for the resource.
2. Retrieve the existing property values for the resource using the getCustomPropertyValues operation.
3. Update the values as needed.
4. Create a SpecificResourceSpecifier object for the resource being updated and specify the properties for the resource.
5. Supply the updateCustomPropertyValues operation with the identification specifier, the updated custom property values, and the specific resource specifier.

   The following code updates the values for a single-select custom property for a file.

```
IdentificationSpecifier is = new IdentificationSpecifier;
ResourceID id = new ResourceID();
id.setValue("0a0a4a35bad1b0750000010f3a50fe4980e6");

CustomPropertyValue[] val = stub.getCustomPropertyValues(id);
SelectionValue[] selVal = new SelectionValue[3];
selVal[0].setValue("Steve Bennett");
selVal[0].setIsSelected(true);
selVal[1].setValue("Lois Sanborn");
selVal[1].setIsSelected(false);
val[0].setSelect(selVal);

SpecificResourceSpecifier rs = new SpecificResourceSpecifier;
File file = new File();
id.setValue("0a0a4a35bad1b0750000010f3a50fe4980e6");
file.setResourceID(id);
ResourcePath rp = new ResourcePath();
rp.setValue("/ModelerStreamLibrary/drugplot.str");
rp.setHierarchyType(HierarchyType.FOLDER);
file.setResourcePath(rp);
rs.setFile(file);

IdentificationSpecifier id = stub.updateCustomPropertyValues(id, val, rs);
```

SOAP request example

Client invocation of the updateCustomPropertyValuesSpecific 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>
 <updateCustomPropertyValuesSpecific xmlns="http://xml.spss.com/repository/remote">
  <IdentificationSpecifierIn xmlns="http://xml.spss.com/repository">
   <identifier value="0a0a4a35bad1b0750000010f3a50fe4980e6"/>
  </IdentificationSpecifierIn>
  <ns3:CustomPropertyValue label="Reviewed by" identifier="0a0a4a35ee6ff9b20000010f4db6f82f814f"
   xmlns:ns3="http://xml.spss.com/repository">
   <ns3:select multipleSelect="false">
    <ns3:selectionValue isSelected="true" value="Steve Bennett"/>
    <ns3:selectionValue isSelected="false" value="Lois Sanborn"/>
   </ns3:select>
  </ns3:CustomPropertyValue>
  <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
   <File>
    <ResourceID value="0a0a4a35bad1b0750000010f3a50fe4980e6"/>
    <ResourcePath value="/ModelerStreamLibrary/drugplot.str"
     hierarchyType="folder"/>
   </File>
  </SpecificResourceSpecifier>
 </updateCustomPropertyValuesSpecific>
</soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a updateCustomPropertyValuesSpecific 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>
  <updateCustomPropertyValuesSpecificResponse xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifierOut xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a35bad1b0750000010f3a50fe4980e6" xsi:type="ResourceID"/>
   </IdentificationSpecifierOut>
  </updateCustomPropertyValuesSpecificResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The updateLabel operation

Updates the security for a specified label. Typically the security for a label is returned by the getLabelSecurity operation and modified as needed before calling this operation.

This operation cannot be used to create a label. The label must already be applied to a resource version in the IBM® SPSS® Collaboration and Deployment Services Repository.

Input fields

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

Table 4-104
Fields for updateLabel

Field	Type/Valid Values	Description
SecureLabel	SecureLabel	Wrapper for a secure label, contains the label and it's Access Control List (ACL).

Java example

To update the security definition for a label:

1. Use the getLabelSecurity operation to return the SecureLabel object for the label being modified.
2. Modify the SecureLabel object settings as desired. For example, the permissions for principals may be altered or a new principal may be added to the access control list.
3. Supply the updateLabel operation with the revised label object.

   The following sample adds a new principal having the *Read* permission to the *Production* label.

```
String myLabel = "Production";
SecureLabel labelSec = stub.getLabelSecurity(myLabel);

AccessControlEntry ace = new AccessControlEntry();
ace.setPermission(PermissionType.READ);
Principal princ = new Principal();
princ.setID("//uAD/spss/gsap");
princ.setDisplayName("gsap (spss)");
princ.setName("gsap (spss)");
princ.setIsGroup(false);
ace.setPrincipal(princ);

AccessControlList acl = labelSec.getAcl();
acl.addAccessControlEntry(ace);

stub.updateLabel(labelSec);
```

SOAP request example

Client invocation of the updateLabel 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>
    <updateLabel xmlns="http://xml.spss.com/repository/remote">
      <SecureLabel label="Production" canRead="true" canModify="true"
        xmlns="http://xml.spss.com/repository">
        <acl>
          <AccessControlEntry Permission="READ">
            <Principal ID="//uAD/spss/aessa" DisplayName="aessa (spss)"
              Name="aessa (spss)" IsGroup="false"/>
          </AccessControlEntry>
          <AccessControlEntry Permission="READ">
            <Principal ID="//uAD/spss/gsap" DisplayName="gsap (spss)"
              Name="gsap (spss)" IsGroup="false"/>
          </AccessControlEntry>
          <AccessControlEntry Permission="READ">
            <Principal ID="//uAD/spss/kkruger" DisplayName="kkruger (spss)"
              Name="kkruger" IsGroup="false"/>
          </AccessControlEntry>
          <AccessControlEntry Permission="MODIFY">
            <Principal ID="//uAD/spss/kkruger" DisplayName="kkruger (spss)"
              Name="kkruger" IsGroup="false"/>
          </AccessControlEntry>
        </acl>
      </SecureLabel>
    </updateLabel>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

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

The updateLabels operation

Updates the security for multiple labels in the system. Typically the security for labels is returned by the getLabelSecurity or getAllLabelSecurity operations and modified as needed before calling this operation.

This operation cannot be used to create labels. The labels must already be applied to resource versions in the IBM® SPSS® Collaboration and Deployment Services Repository.

Input fields

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

Table 4-105
Fields for updateLabels

Field	Type/Valid Values	Description
SecureLabel	SecureLabel[]	Wrapper for a secure label, contains the label and it's Access Control List (ACL).

Java example

To update the security definition for multiple labels:

1. Use the getAllLabelSecurity operation to return an array of SecureLabel objects in the system.
2. Modify the settings for the SecureLabel objects as desired. For example, the permissions for principals may be altered or a new principal may be added to the access control list.
3. Supply the updateLabels operation with the revised label objects.

   The following sample adds a new principal having the *Modify* permission to the first label object and changes the permission for the first principal in the access control list for the second label to *Read*.

```
SecureLabel[] labelSec = stub.getAllLabelSecurity(myLabel);

AccessControlEntry ace = new AccessControlEntry();
ace.setPermission(PermissionType.MODIFY);
Principal princ = new Principal();
princ.setID("//gNative//$$security/everyoneGroup");
princ.setDisplayName("–everyone –");
princ.setName("$$security/everyoneGroup");
```

```
princ.setIsGroup(true);
ace.setPrincipal(princ);
AccessControlList acl = labelSec[0].getAcl();
acl.addAccessControlEntry(ace);
labelSec[0].setAcl(acl);

AccessControlList changeAcl = labelSec[1].getAcl();
AccessControlEntry[] editAce = changeAcl.getAccessControlEntry();
editAce[0].setPermission(PermissionType.READ);
changeAcl.setAccessControlEntry(editAce);
labelSec[1].setAcl(changeAcl);

stub.updateLabels(labelSec);
```

SOAP request example

Client invocation of the updateLabels 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>
    <updateLabels xmlns="http://xml.spss.com/repository/remote">
      <SecureLabel label="Draft" canRead="true" canModify="true"
        xmlns="http://xml.spss.com/repository">
        <acl>
          <AccessControlEntry Permission="MODIFY">
            <Principal ID="//gNative//$$security/everyoneGroup"
              DisplayName="–everyone –" Name="$$security/everyoneGroup"
              IsGroup="true"/>
          </AccessControlEntry>
          <AccessControlEntry Permission="READ">
            <Principal ID="//gNative//$$security/everyoneGroup"
              DisplayName="–everyone –" Name="$$security/everyoneGroup"
              IsGroup="true"/>
          </AccessControlEntry>
          <owner ID="" Name=""/>
        </acl>
      </SecureLabel>
```

```
        <SecureLabel label="Production" canRead="true" canModify="true"
          xmlns="http://xml.spss.com/repository">
          <acl>
            <AccessControlEntry Permission="READ">
          <Principal ID="//uAD/spss/aessa" DisplayName="aessa (spss)"
            Name="aessa (spss)" IsGroup="false"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="MODIFY">
              <Principal ID="//uAD/spss/kkruger" DisplayName="kkruger (spss)"
                Name="kkruger" IsGroup="false"/>
            </AccessControlEntry>
            <AccessControlEntry Permission="READ">
              <Principal ID="//uAD/spss/kkruger" DisplayName="kkruger (spss)"
                Name="kkruger" IsGroup="false"/>
            </AccessControlEntry>
            <owner ID="" Name=""/>
          </acl>
        </SecureLabel>
      </updateLabels>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a updateLabels 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>
    <updateLabelsResponse xmlns="http://xml.spss.com/repository/remote"/>
  </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. The client must supply at least an identifier and the modification date, as well as the metadata to be updated. 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 4-106
Fields for updateResource

Field	Type/Valid Values	Description
TargetIdentificationSpecifier	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.
ResourceSpecifier	ResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Return information

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

Table 4-107
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.

Java example

The following sample updates a file resource by changing the author and associating the file with an existing topic.

```
ResourcePath rp = new ResourcePath();
rp.setValue("/" + getFolderName() + "/myNewFile.xls");
IdentificationSpecifier is = new IdentificationSpecifier();
is.setIdentifier(rp);
ResourceSpecifier rs = stub.getResource(is);
File f = (File)rs.getResource();
f.getAuthor().setValue("J.Q. Public");
ResourcePath rpt = new ResourcePath();
rpt.setValue("/myTopicHierarchy/myFirstTopic");
f.addAssociatedTopic(rpt);
stub.updateResource(getResourceSpecifier(f));
```

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: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>
  <updateResource xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106a7a10b13803c" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
   <ResourceSpecifier xmlns="http://xml.spss.com/repository">
    <Resource xsi:type="File" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <ResourceID value="0a0a4aac00072ffb00000106a7a10b13803c"/>
     <ResourcePath value="/ModelerStreamLibrary/Joining Age.html" hierarchyType="folder"/>
    </Resource>
   </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">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4aac00072ffb00000106a7a10b13803c" xsi:type="ResourceID"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </IdentificationSpecifier>
  </updateResourceResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The updateResourceMaintenanceProviders operation

This operation is for internal use only.

Input fields

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

Table 4-108
Fields for updateResourceMaintenanceProviders

Field	Type/Valid Values	Description
ResourceMaintenanceProviderSpecifiers	ResourceMaintenanceProviderSpecifierWrapper[]	Wrapper for a resource maintenance provider. Used to define service interface parameter to preserve polymorphism.

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 IdentificationSpecifier 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 IdentificationSpecifier corresponds to the first ResourceSpecifier.

Input fields

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

Table 4-109
Fields for updateResources

Field	Type/Valid Values	Description
IdentificationSpecifier	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.
ResourceSpecifier	ResourceSpecifier[]	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Return information

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

Table 4-110
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.

Java example

To update multiple resources:

1. Create an array of IdentificationSpecifier objects for the resources to be updated. Set the identification and version information for each resource.
2. Create an array of ResourceSpecifier objects to contain the updated information for the resources. Set the new properties for each resource.
3. Supply the updateResources operation with both arrays.

   The following code updates the expiration dates for two files.

```
IdentificationSpecifier[] is = new IdentificationSpecifier[2];

ResourceID id = new ResourceID();
id.setValue("0a0a4a35bad1b0750000010f3a50fe4980e6");
is[0].setIdentifier(id);
Version vsion = new Version();
vsion.setMarker("0:2006-11-30 14:19:20.769");
is[0].setVersion(vsion);
```

```
id.setValue("0a0a4a353da969fa0000010ef0fcc01f8649");
is[1].setIdentifier(id);
vsion.setMarker("0:2006-11-17 09:59:42.666");
vsion.setLabel("Production");
is[1].setVersion(vsion);

ResourceSpecifier[] rs = new ResourceSpecifier[2];
File file = new File();
id.setValue("0a0a4a35bad1b0750000010f3a50fe4980e6");
file.setResourceID(id);
ResourcePath rp = new ResourcePath();
rp.setValue("/ModelerStreamLibrary/drugplot.str");
rp.setHierarchyType(HierarchyType.FOLDER);
file.setResourcePath(rp);
ExpirationDate exp = new ExpirationDate();
Calendar date = new Calendar();
date.set(2007, 01, 31, 0, 0, 0);
exp.setValue(date);
file.setExpirationDate(exp);
rs[0].setResource(file);

id.setValue("0a0a4a353da969fa0000010ef0fcc01f8649");
file.setResourceID(id);
rp.setValue("/StatisticsLibrary/tree_model.sps");
rp.setHierarchyType(HierarchyType.FOLDER);
file.setResourcePath(rp);
date.set(2007, 01, 31, 0, 0, 0);
exp.setValue(date);
file.setExpirationDate(exp);
rs[1].setResource(file);

IdentificationSpecifier[] idspec = stub.updateResources(is, rs);
```

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: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>
  <updateResources xmlns="http://xml.spss.com/repository/remote">
   <ns2:IdentificationSpecifier xmlns:ns2="http://xml.spss.com/repository">
    <ns2:identifier value="0a0a4a35bad1b0750000010f3a50fe4980e6" xsi:type="ResourceID"/>
    <ns2:Version marker="0:2006-11-30 14:19:20.769"/>
   </ns2:IdentificationSpecifier>
   <ns3:IdentificationSpecifier xmlns:ns3="http://xml.spss.com/repository">
    <ns3:identifier value="0a0a4a353da969fa0000010ef0fcc01f8649" xsi:type="ResourceID"/>
    <ns3:Version marker="0:2006-11-17 09:59:42.666">
     <ns3:label>Production</ns3:label>
    </ns3:Version>
   </ns3:IdentificationSpecifier>
   <ns4:ResourceSpecifier xmlns:ns4="http://xml.spss.com/repository">
    <ns4:Resource xsi:type="File">
     <ns4:ResourceID value="0a0a4a35bad1b0750000010f3a50fe4980e6"/>
     <ns4:ResourcePath value="/ModelerStreamLibrary/drugplot.str" hierarchyType="folder"/>
     <ns4:ExpirationDate value="2007-01-31T00:00:00.000-06:00"/>
     <ns4:associatedTopicList/>
    </ns4:Resource>
   </ns4:ResourceSpecifier>
   <ns5:ResourceSpecifier xmlns:ns5="http://xml.spss.com/repository">
    <ns5:Resource xsi:type="File">
     <ns5:ResourceID value="0a0a4a353da969fa0000010ef0fcc01f8649"/>
     <ns5:ResourcePath value="/StatisticsLibrary/tree_model.sps" hierarchyType="folder"/>
     <ns5:ExpirationDate value="2007-01-31T00:00:00.000-06:00"/>
     <ns5:associatedTopicList/>
    </ns5:Resource>
   </ns5: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">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a35bad1b0750000010f3a50fe4980e6" xsi:type="ResourceID"/>
   </IdentificationSpecifier>
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a353da969fa0000010ef0fcc01f8649" xsi:type="ResourceID"/>
   </IdentificationSpecifier>
  </updateResourcesResponse>
```

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

The updateResourceSpecific operation

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

The difference between this operation and the updateResource operation is the structure of the wrapper for the resource input. The updateResource operation accepts a ResourceSpecifier object defining the resource. Different resource types are handled through the use of XMLSchema instance types. In contrast, the updateResourceSpecific operation accepts a SpecificResourceSpecifier object defining the resource. Different resource types are handled through the use of different objects for the types. For example, instead of handling a Resource object of the *File* type, updateResourceSpecific processes a File object. Clients that cannot properly handle instance types should use updateResourceSpecific instead of updateResource.

Input fields

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

Table 4-111
Fields for updateResourceSpecific

Field	Type/Valid Values	Description
TargetIdentificationSpecifier	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.
SpecificResourceSpecifier	SpecificResourceSpecifier	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Return information

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

Table 4-112
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.

Java example

To update a resource:

1. Create an IdentificationSpecifier object for the resource to be updated. Set the identification and version information for the resource.
2. Create a SpecificResourceSpecifier object to contain the updated information for the resource. Set the new properties for the resource.
3. Supply the updateResourceSpecific operation with both specifiers.

   The following code updates the label for a file.

```
IdentificationSpecifier is = new IdentificationSpecifier;
ResourceID id = new ResourceID();
id.setValue("0a0a4a353da969fa0000010ef0fcc01f8649");
is.setIdentifier(id);

SpecificResourceSpecifier rs = new SpecificResourceSpecifier;
File file = new File();
Version vsion = new Version();
vsion.setLabel("Production");
file.setVersion(vsion);
rs.setFile(file);

IdentificationSpecifier idspec = stub.updateResourceSpecific(is, rs);
```

SOAP request example

Client invocation of the updateResourceSpecific 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>
  <updateResourceSpecific xmlns="http://xml.spss.com/repository/remote">
   <TargetIdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a353da969fa0000010ef0fcc01f8649"/>
   </TargetIdentificationSpecifier>
```

```
    <SpecificResourceSpecifier xmlns="http://xml.spss.com/repository">
     <File>
      <Version>
       <label>Production</label>
      </Version>
     </File>
    </SpecificResourceSpecifier>
   </updateResourceSpecific>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a updateResourceSpecific 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>
  <updateResourceSpecificResponse xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a353da969fa0000010ef0fcc01f8649" xsi:type="ResourceID"/>
    <Version>
     <label>Production</label>
    </Version>
   </IdentificationSpecifier>
  </updateResourceSpecificResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

The updateResourcesSpecific 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 updateResourceSpecific operation for each resource to be updated. However, using updateResourcesSpecific, only one web service call is needed.

The difference between this operation and the updateResources operation is the structure of the wrapper for the resource input. The updateResources operation accepts an array of ResourceSpecifier objects defining the resources. Different resource types are handled through the use of XMLSchema instance types. In contrast, the updateResourcesSpecific operation accepts an array of SpecificResourceSpecifier objects defining the resources. Different resource types are handled through the use of different objects for the types. For example, instead of handling a Resource object of the *File* type, updateResourcesSpecific processes a File object. Clients that cannot properly handle instance types should use updateResourcesSpecific instead of updateResources.

Input fields

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

Table 4-113
Fields for updateResourcesSpecific

Field	Type/Valid Values	Description
TargetIdentificationSpecifier	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.
SpecificResourceSpecifier	SpecificResourceSpecifier[]	Wrapper for a resource object. Used to define service interface parameter to preserve polymorphism.

Return information

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

Table 4-114
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.

Java example

To update multiple resources:

1. Create an array of IdentificationSpecifier objects for the resources to be updated. Set the identification and version information for each resource.
2. Create an array of SpecificResourceSpecifier objects to contain the updated information for the resources. Set the new properties for each resource.
3. Supply the updateResourcesSpecific operation with both arrays.

   The following code updates the expiration dates for two files.

```
IdentificationSpecifier[] is = new IdentificationSpecifier[2];

ResourceID id = new ResourceID();
id.setValue("0a0a4a35bad1b0750000010f3a50fe4980e6");
is[0].setIdentifier(id);
Version vsion = new Version();
vsion.setMarker("0:2006-11-30 14:19:20.769");
is[0].setVersion(vsion);
```

```
id.setValue("0a0a4a353da969fa0000010ef0fcc01f8649");
is[1].setIdentifier(id);
vsion.setMarker("0:2006-11-17 09:59:42.666");
vsion.setLabel("Production");
is[1].setVersion(vsion);

SpecificResourceSpecifier[] rs = new SpecificResourceSpecifier[2];
File file = new File();
id.setValue("0a0a4a35bad1b0750000010f3a50fe4980e6");
file.setResourceID(id);
ResourcePath rp = new ResourcePath();
rp.setValue("/ModelerStreamLibrary/drugplot.str");
rp.setHierarchyType(HierarchyType.FOLDER);
file.setResourcePath(rp);
ExpirationDate exp = new ExpirationDate();
Calendar date = new Calendar();
date.set(2007, 01, 31, 0, 0, 0);
exp.setValue(date);
file.setExpirationDate(exp);
rs[0].setFile(file);

id.setValue("0a0a4a353da969fa0000010ef0fcc01f8649");
file.setResourceID(id);
rp.setValue("/StatisticsLibrary/tree_model.sps");
rp.setHierarchyType(HierarchyType.FOLDER);
file.setResourcePath(rp);
date.set(2007, 01, 31, 0, 0, 0);
exp.setValue(date);
file.setExpirationDate(exp);
rs[1].setFile(file);

IdentificationSpecifier[] idspec = stub.updateResourcesSpecific(is, rs);
```

SOAP request example

Client invocation of the updateResourcesSpecific 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>
  <updateResourcesSpecific xmlns="http://xml.spss.com/repository/remote">
   <ns2:IdentificationSpecifier xmlns:ns2="http://xml.spss.com/repository">
    <ns2:identifier value="0a0a4a35bad1b0750000010f3a50fe4980e6" xsi:type="ResourceID"/>
    <ns2:Version marker="0:2006-11-30 14:19:20.769"/>
   </ns2:IdentificationSpecifier>
   <ns3:IdentificationSpecifier xmlns:ns3="http://xml.spss.com/repository">
    <ns3:identifier value="0a0a4a353da969fa0000010ef0fcc01f8649" xsi:type="ResourceID"/>
    <ns3:Version marker="0:2006-11-17 09:59:42.666">
     <ns3:label>Production</ns3:label>
    </ns3:Version>
   </ns3:IdentificationSpecifier>
   <ns4:SpecificResourceSpecifier xmlns:ns4="http://xml.spss.com/repository">
    <ns4:File>
     <ns4:ResourceID value="0a0a4a35bad1b0750000010f3a50fe4980e6"/>
     <ns4:ResourcePath value="/ModelerStreamLibrary/drugplot.str" hierarchyType="folder"/>
     <ns4:ExpirationDate value="2007-01-31T00:00:00.000-06:00"/>
     <ns4:associatedTopicList/>
    </ns4:File>
   </ns4:SpecificResourceSpecifier>
   <ns5:SpecificResourceSpecifier xmlns:ns5="http://xml.spss.com/repository">
    <ns5:File>
     <ns5:ResourceID value="0a0a4a353da969fa0000010ef0fcc01f8649"/>
     <ns5:ResourcePath value="/StatisticsLibrary/tree_model.sps" hierarchyType="folder"/>
     <ns5:ExpirationDate value="2007-01-31T00:00:00.000-06:00"/>
     <ns5:associatedTopicList/>
    </ns5:File>
   </ns5:SpecificResourceSpecifier>
  </updateResourcesSpecific>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a updateResourcesSpecific 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>
  <updateResourcesSpecificResponse xmlns="http://xml.spss.com/repository/remote">
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a35bad1b0750000010f3a50fe4980e6" xsi:type="ResourceID"/>
   </IdentificationSpecifier>
   <IdentificationSpecifier xmlns="http://xml.spss.com/repository">
    <identifier value="0a0a4a353da969fa0000010ef0fcc01f8649" xsi:type="ResourceID"/>
   </IdentificationSpecifier>
  </updateResourcesSpecificResponse>
```

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

Appendix

A

Content Repository and Scheduling Server interactions

The Scheduling Server Service stores objects in a native structure optimized for the execution of event clusters. That structure differs from the hierarchical folder system used by the Content Repository Service, which is optimized for organizing and retrieving file resources. However, clients often wish to make Scheduling Server Service objects available to the Content Repository Service to take advantage of its functionality, such as versioning.

To accommodate these differing storage architectures, the Content Repository Service includes two operations that map identifiers across systems. The getIdentificationSpecifier operation takes a single string parameter that represents a unique identifier of the resource in the external system and returns a fully qualified IdentificationSpecifier that includes all necessary information to locate the version of the resource in the Content Repository hierarchy. In contrast, the getResourceImplementationId operation returns information about the unique identifier in the external system for a given Content Repository identifier.

The following examples describe how the identifiers are used across the two services. In the first, an event cluster from the Scheduling Server Service is stored within the repository using the Content Repository Service. In the second, an existing event cluster is updated with a schedule.

Storing event clusters

Storing an event cluster using the Content Repository Service involves the following general approach:

1. Create a new File instance.
2. Define the metadata for the file, such as the description and keywords.
3. Specify the appropriate MIME type for the file. For an event cluster, the MIME type must be *application/x-vnd.spss-prms-job*.
4. Using the Scheduling Server Service proxy classes, create an event cluster with the desired structure.
5. Attach the event cluster to the SOAP request as a MIME attachment.
6. Create a resource specifier for the file resource.
7. Access the identifier for the parent folder in which the file should be stored.
8. Use createResource to add the resource to the repository in the desired location.

The following Java code illustrates these steps:

```
File file = new File();
setupFileMetadata(file);
```

```
MimeType mimeType = new MimeType();
mimeType.setValue("application/x-vnd.spss-prms-job");
file.setMimeType(mimeType);
EventCluster eventCluster = createPRMSJob();
ContentRepository contentRepository = getContentRepository();
createSOAPAttachment(contentRepository, eventCluster, file);
ResourceSpecifier resourceSpecifier = new ResourceSpecifier();
resourceSpecifier.setResource(file);
IdentificationSpecifier parentIdentificationSpecifier = getParentIdentificationSpecifier();
IdentificationSpecifier identificationSpecifier =
  contentRepository.createResource(parentIdentificationSpecifier, resourceSpecifier);
```

Assigning schedules

Assigning a schedule to an event cluster stored in the repository involves the following general approach:

1. Create a schedule using the Scheduling Server Service proxy classes.
2. Add the schedule as an update for the cluster.
3. Retrieve the native identifier for the event cluster by supplying the getResourceImplementationId operation with the identification specifier for the resource in the repository.
4. Update the event cluster using the native identifier for the Scheduling Server Service.

The following Java code illustrates these steps:

```
DailySchedule dailySchedule = createDailySchedule();
AddObject addObject = new AddObject();
addObject.setObjectToBeAdded(dailySchedule);
Updates updates = new Updates();
updates.addUpdateCommand(addObject);
ResourceImplementationId resourceImplementationId =
  contentRepository.getResourceImplementationId(identificationSpecifier);
String eventClusterId = resourceImplementationId.getId().getValue();
SchedulingServer schedulingServer = getSchedulingServer();
schedulingServer.updateCluster(eventClusterId, updates, null, false);
```

Appendix

B

Microsoft® .NET Framework-based clients

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

1. Add a Service Reference. For more information, see the topic Adding a service reference on p. 177.

2. Configure the web service endpoint. For more information, see the topic Configuring the web service endpoint on p. 179.

3. Programmatically configure the necessary endpoint behaviors. For more information, see the topic Configuring endpoint behaviors on p. 180.

4. Exercise the web service. For more information, see the topic Exercising the service on p. 180.

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

Adding a service reference

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

1. In Visual Studio, right-click the folder's *References* folder and select Add Service Reference.

2. Type the URL of the service WSDL location in the Address field, and click Go. The value corresponds to the service endpoint appended with *?wsdl*.

3. Specify the desired namespace in the Namespace field.

4. Click OK.

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

Service reference modifications

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

To modify a service reference:

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

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

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

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

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

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

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

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

Configuring the web service endpoint

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

1. In Visual Studio, double-click the *app.config* file for the application (or *web.config* for a web-application).

2. Find the system.serviceModel element. Create it if it does not already exist.

3. Find the client element. Create it if it does not already exist.

4. Create a new endpoint element as a child of the client element.

5. Specify the appropriate service endpoint URL as the value of the *address* attribute.

6. Specify *basicHttpBinding* as the value of the *binding* attribute.

7. Specify the appropriate service contract as the value of the *contract* attribute. The service contract is the value of the service reference namespace appended with the service name.

8. Optionally specify a value for the *name* attribute that identifies a name for the endpoint configuration. If the *name* is blank, the configuration is used as the default for the service.

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

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

Configuring endpoint behaviors

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

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

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

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

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

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

Exercising the service

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

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

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

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

The corrected code is as follows:

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

Appendix C

Notices

This information was developed for products and services offered worldwide.

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

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

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

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

Intellectual Property Licensing, Legal and Intellectual Property Law, IBM Japan Ltd., 1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242-8502 Japan.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

IBM Software Group, Attention: Licensing, 233 S. Wacker Dr., Chicago, IL 60606, USA.

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

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

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

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

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

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

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

Trademarks

IBM, the IBM logo, ibm.com, and SPSS are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at *http://www.ibm.com/legal/copytrade.shtml*.

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

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

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

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

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

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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


Java

Index