

**IBM SPSS Collaboration and  
Deployment Services 5 PevServices  
Service Developer's Guide**



*Note:* Before using this information and the product it supports, read the general information under Notices on p. 57.

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

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

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

Licensed Materials - Property of IBM

**© Copyright IBM Corporation 2000, 2012.**

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# 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

<b>1</b>	<b><i>Introduction to web services</i></b>	<b>1</b>
	What are web services? . . . . .	1
	Web service system architecture . . . . .	1
	Web service protocol stack . . . . .	2
	Simple Object Access Protocol . . . . .	3
	Web Service Description Language . . . . .	3
	Proxies . . . . .	6
<b>2</b>	<b><i>PevServices Service overview</i></b>	<b>8</b>
	Accessing the PevServices Service . . . . .	8
	Calling PevServices Service operations . . . . .	8
<b>3</b>	<b><i>IBM SPSS Collaboration and Deployment Services Enterprise View concepts</i></b>	<b>10</b>
	Enterprise View . . . . .	10
	Data type . . . . .	11
	Application View . . . . .	11
	Environment . . . . .	12
	Direction . . . . .	12
	Data Provider Definition - Real Time . . . . .	13
	Variables . . . . .	13
	Uniform Resource Identifiers . . . . .	14
<b>4</b>	<b><i>Operation reference</i></b>	<b>16</b>
	Manage port type . . . . .	16
	The getAvMergeConflicts operation . . . . .	16
	The getEvMergeConflicts operation . . . . .	20
	The getVersion operation . . . . .	23
	The performAvMerge operation . . . . .	24
	The performEvMerge operation . . . . .	28
	The validateObject operation . . . . .	30

Metadata port type . . . . .	31
The getCategories operation . . . . .	31
The getCustomAttributes operation . . . . .	33
The getVariableMetadata operation. . . . .	36
The getVersion operation. . . . .	39
Search port type. . . . .	40
The getCompatibleAv operation. . . . .	41
The getCompatibleRtDpd operation . . . . .	44
The getRtDpdInputDescription operation. . . . .	47
The getVersion operation. . . . .	50

## ***Appendices***

### ***A Microsoft® .NET Framework-based clients*** **52**

Adding a service reference. . . . .	52
Service reference modifications . . . . .	53
Configuring the web service endpoint. . . . .	54
Configuring endpoint behaviors . . . . .	55
Exercising the service. . . . .	55

### ***B Notices*** **57**

### ***Index*** **60**



# 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.

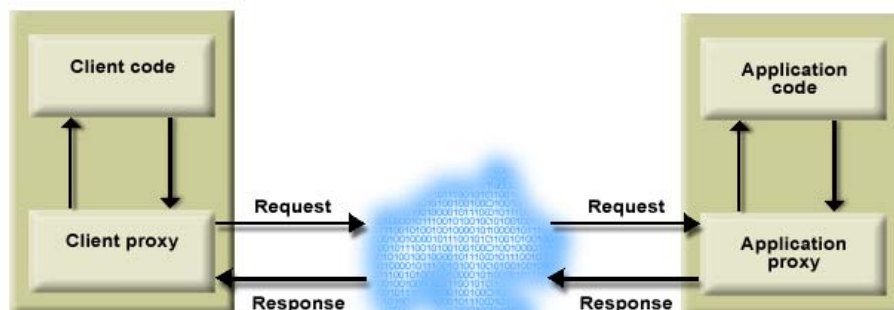
Advantages to implementing application functionality as web services include the following:

- 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*



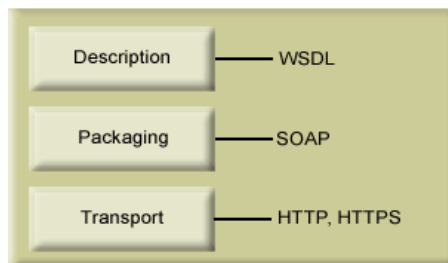
The client code supplies input to an operation offered by a proxy class. The proxy class generates a request containing a standardized representation of the input and sends it across the network to the application. A proxy class on the server receives the request and unmarshals the contents into objects 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.](#)



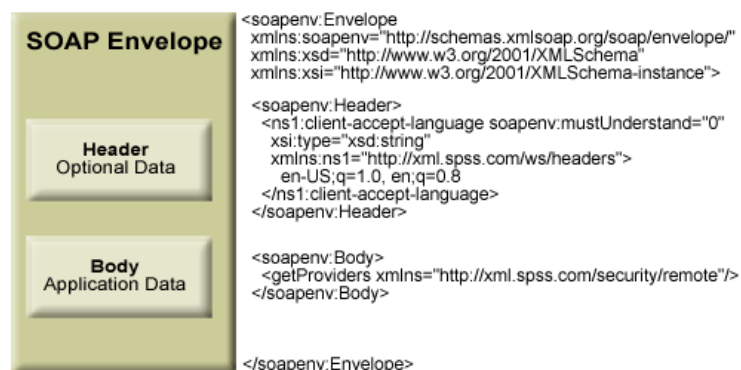
## Simple Object Access Protocol

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 \(http://www.w3.org/TR/wsdl\)](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="tns: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="tns:getProvidersResponse" name="parameters" />
</wsdl:message>
<wsdl:message name="getProvidersRequest">
  <wsdl:part element="tns: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:operation>
</wsdl:binding>
```

```
</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 `getProvidersResponse` 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.

## ***PevServices Service overview***

The PevServices Service provides functionality used when working with the input data sources for analytic and scoring tasks. In general, the service provides the ability to perform the following tasks:

- Manage Enterprise View and Application View resource versions, allowing two versions of a view to be merged into a new composite version
- Retrieve metadata about the variables available in data sources
- Search for Application View and Data Provider Definition - Real Time resources that are compatible with specified criteria

The PevServices Service is often used in conjunction with the Scoring Service. Use the PevServices Service to discover data sources that are compatible with a particular scoring configuration. The metadata for variables in those data sources can be used to construct user interfaces that prompt for scoring input and to validate any supplied data values.

### ***Accessing the PevServices Service***

To access the functionality offered by the PevServices Service, create a client application using the proxy classes generated by your preferred web service tool. The service includes three port types having the following endpoints:

```
http://<host-name>:<port-number>/pev/services/Manage  
http://<host-name>:<port-number>/pev/services/Metadata  
http://<host-name>:<port-number>/pev/services/Search
```

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/pev/services/Manage?wsdl  
http://cads_server:80/pev/services/Metadata?wsdl  
http://cads_server:80/pev/services/Search?wsdl
```

### ***Calling PevServices 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 = "/pev/services/Search";  
URL url = new URL("http", "cads_server", 80, context);
```

```
pevService service = new pevServiceLocator();  
stub = service.getStatus(url);
```

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

```
stub.getRtDpdInputDescription(type);
```

# ***IBM SPSS Collaboration and Deployment Services Enterprise View concepts***

## ***Enterprise View***

The Enterprise View is an abstracted dictionary of the enterprise data used in predictive analytic applications. It provides a simplified business-oriented picture of the enterprise data available for data mining and reporting. The Enterprise View is constructed and maintained by a data expert (a database administrator, for example). The Enterprise View editor in IBM® SPSS® Collaboration and Deployment Services Deployment Manager provides the data expert with a graphical user interface to build, maintain, and validate global information about enterprise data. Taken together, the Enterprise View editor, the Application View editor, the Data Provider Definition editor, and the Data Provider Definition - Real Time editor constitute the IBM® SPSS® Collaboration and Deployment Services Enterprise View builder user interface used by data experts in constructing, validating, and managing the IBM SPSS Collaboration and Deployment Services Enterprise View.

Upon installation, the Enterprise View is automatically created as an empty singleton object alongside the IBM® SPSS® Collaboration and Deployment Services Repository. The data expert populates the empty Enterprise View with table and column metadata used in predictive analytic applications. Changes to the Enterprise View are handled in the same manner as all repository objects; a new version is created and all previous versions continue to exist, unless they are deleted manually.

It is not possible to delete the Enterprise View from the repository completely. At least one version is required. Only users who are members of the **Administrators** role, and those assigned to a role with the **Manage Enterprise View** action, are allowed to work with the Enterprise View.

Each column in an Enterprise View table has a name that is unique within the table. In addition, columns are characterized using the following metadata:

- **Type.** For more information, see the topic [Data type on p. 11](#).
- **Category information.** The categories attribute defines categorical values for a column. The category attributes are used by consuming applications to populate the values for selection fields or menus. Categorical values can be used for any column, regardless of its type.
- **Minimum/maximum.** The minimum/maximum attributes function as user aids by providing defined limits for each column. For example, the column *INCOME* might have a minimum value of 20000 and a maximum value of 75000.
- **Description.** The Enterprise View column description is intended to provide high-level information about the column.

For more information on working with the Enterprise View, see the Deployment Manager documentation.

The PevServices Service includes operations for merging two versions of the Enterprise View.



## Data type

The type attribute specifies the type of data stored in a column. Each column can store data consisting of only a single data type.

Table 3-1  
Data types

Type	Description
Boolean	Used to store binary digits (0 or 1). These can be used to denote Boolean values, such as Yes/No, True/False, or On/Off.
Date	Values must have the form CCYY-MM-DD, where CCYY denotes the year, MM denotes the number of the month (January = 1, February = 2, and so on), and DD denotes the day (starting with 1). Optionally, the year can be preceded by + or -. Leading zeros for the year, month, and day are optional. Spaces are not allowed before or after the - separator.
Daytime	Values must have the form HH:MM:SS.xxx, where HH denotes the hours (ranging from 0 to 23), MM denotes the minutes (0 to 59), SS denotes the seconds (0 to 59), and xxx, the milliseconds (0 to 999). First, note that only the 24-hour representation is allowed. Furthermore, the seconds and milliseconds are optional. Leading zeros for the hours, minutes, and seconds are optional. Spaces are not allowed before or after the : separator.
Decimal	Used to store exact numbers. The scale and maximum precision can be configured. The precision denotes the maximum number of digits each value may consist of. The scale indicates how many of those digits may occur to the right of the decimal point (that is, the fractional part). The value of this parameter must be greater than or equal to zero and less than or equal to the precision.
Double	Represents signed, approximate numerical values. Double permits much larger numbers than Float.
Float	Represents signed, approximate numerical values. Double permits much larger numbers than Float.
Integer	Used for whole numbers (for example, 5 or 110).
Long	Used for character data of variable length up to two gigabytes.
String	Used to store a sequence of ISO-8859-Latin-1 characters. The maximum length of such a sequence is configurable. Furthermore, the string may be empty and cannot contain the newline (ASCII 10) or null (ASCII 0) characters.
Timestamp	Allows values of the form CCYY-MM-DDTHH:MM:SS.xxx in accordance with the ISO 8601 standard. It is easy to see that it is nothing more than the concatenation of the Date and the Time types, using a delimiter of T to indicate the beginning of the Time portion. Consequently, the rules for the Date and the Time types are also applicable to the Timestamp type.

## Application View

An Application View collects a subset of tables and columns from the Enterprise View that relate to a specific application. It captures additional information about how tables and columns are used in the context of an application (for example, call center or fraud detection). There can be more than one Application View associated with the Enterprise View.

The Application View serves two purposes. First, it provides a means to constrain the information displayed to a user in a tool or application based on the IBM® SPSS® Collaboration and Deployment Services Enterprise View. For example, a user may be interested only in call center information. By identifying which IBM SPSS Collaboration and Deployment Services Enterprise View objects are used in the call center application, user interface logic can limit results

to only those relevant objects. Second, the Application View provides a means for a system administrator or data expert to view the repository from the perspective of an application. This is especially useful in assessing the effect changes might have on a specific application.

Each column in an Application View table has a name that is unique within the table. In addition, columns are characterized using the following metadata:

- **Type.** For more information, see the topic [Data type](#) on p. 11.
- **Environment.** For more information, see the topic [Environment](#) on p. 12.
- **Direction.** For more information, see the topic [Direction](#) on p. 12.
- **Description.** The Application View column description is intended to provide high-level information about the column.

For more information on working with the Application View, see the IBM® SPSS® Collaboration and Deployment Services Deployment Manager documentation.

The PevServices Service includes operations for merging two versions of the Application View and for .

## **Environment**

The environment settings provide a means of identifying which particular columns should be associated with defined business segments. The business segments include:

- **Analytical.** Columns associated with an Analytic environment typically define the transaction data needed to run analytic and optimization tasks.
- **Operational.** Columns associated with an Operational environment are typically used to deploy the resultant analytics to operational channels.
- **Reporting.** Columns associated with an Reporting environment are typically used to provide the resultant analytics to reporting channels.

The column environment determines the availability of the column in data provider definitions. The data provider definition is associated with a particular environment for the Application View. Only columns belonging to that environment will be available to the data provider definition. For example, an *Analytic* data provider definition has access only to columns in the associated Application View that are defined as *Analytical*.

Columns can be defined as belonging to multiple environments. For example, the column *CUST\_ID* can be assigned to both the *Analytical* and *Operational* roles. In this case, the column would be available in both *Analytic* and *Operational* data provider definitions.

## **Direction**

The direction setting specifies whether columns should be treated as predictors or targets. Predictors are used as an input to machine learning (predictor fields); targets are used as output for machine learning (predicted fields). The direction setting does not affect how columns are used or presented. The setting serves as a aid, allowing users to ascertain each column's intended role. IBM® SPSS® Modeler is one example of an application that utilizes the direction setting.

## Data Provider Definition - Real Time

The Data Provider Definition - Real Time is used exclusively for real time interactions with applications. A Data Provider Definition - Real Time links data sets directly to an Application View and indirectly to Enterprise View tables and columns. A single Application View table may have more than one associated Data Provider Definition - Real Time, each of which may point to a different data source.

A data set represents a source of data. In its simplest form, a data set maps to a single data source table and abstracts the physical implementation of the data source. In addition to mapping to a data source table, a data set can map to other data sets. Using this mechanism, data from various data sources can be linked together.

The Data Provider Definition - Real Time retrieves data sets, one row at a time, from the following sources:

- JDBC data source defined in the Data Provider Definition - Real Time
- Application server defined data source
- User context data defined in the Data Provider Definition - Real Time
- User created data supplier (implemented externally)

*Note:* Additional data columns can be derived from existing columns.

The PevServices Service includes operations for finding Data Provider Definition - Real Time items that are compatible with an Application View and for accessing the input metadata for a specific Data Provider Definition - Real Time.

## Variables

Variables serve as input to analytical and scoring procedures, and are characterized by a set of metadata values. The metadata attributes used for variables depend on the data source and may include:

- **Name.** Short name of variable that is unique within the data source and can be used as a variable identifier.
- **Variable type.** The variable type indicates whether the variable is simple or category based. Category types include group, multiple category, and multiple dichotomy.
- **Display name.** Descriptive name of the variable typically displayed in user interfaces.
- **Data type.** The type of data stored in the variable. [For more information, see the topic Data type on p. 11.](#)
- **Measurement level.** The IBM® SPSS® Statistics measurement level. Valid values include nominal, ordinal and scale.
- **Size.** Variable size.
- **Number of existing categories.** A variable referred to as nominal, ordinal, or more generally categorical allows only a discrete set of values to be used. The number of existing categories reports the number of category values found in the data-source for the variable.

- **Category information.** Obtaining the set of permissible values for categorical variables allows elements of user interfaces, such as selection fields and menus, to present only valid values for those variables. In addition, variable values supplied by the user can be validated against the allowed category set. Each value in the category set is characterized by an id representing the value and a label, if any, associated with the value.
- **Multiple response set information.** Multiple-response variables correspond to questions that can have more than one value for each case. For example, the respondent may have been asked to circle all magazines read within the last month in a list of magazines. Multiple-response data can be organized in one of two ways. In the first approach, each possible response corresponds to a variable that can have one of two values, such as 1 for no and 2 for yes; this is the **multiple-dichotomy** method. Alternatively, the maximum number of possible answers from a respondent may be used to create that number of variables, each of which can have a value representing one of the possible answers, such as 1 for Time, 2 for Newsweek, and 3 for PC Week. If an individual did not give the maximum number of answers, the extra variables receive a missing-value code. This is the multiple-response or **multiple-category** method of coding answers. In both approaches, the variables representing the multiple responses are grouped for subsequent analysis. For a multiple-dichotomy group, each component variable with at least one yes value across cases becomes a category of the group variable. For a multiple-category group, each value becomes a category.
- **Custom attribute information.** In addition to standard attributes, variables in SPSS Statistics data files may have custom attributes defined. For example, there may be a variable attribute that identifies the type of response for survey questions (for example, single selection, multiple selection, fill-in-the-blank) or the formulas used for computed variables. Each attribute is characterized by an attribute name and a set of permitted values.

The PevServices Service includes operations for retrieving the full set of metadata, the categorical information, and the custom attributes for variables.

## ***Uniform Resource Identifiers***

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

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

The URI has the following format:

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

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

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

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

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

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

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

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

`spsscr://localhost/campaign2`

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

# Operation reference

## Manage port type

The Manage port type includes operations used for merging views.

### The *getAvMergeConflicts* operation

Identifies differences in the definitions for two specified versions of an Application View that would result in conflicts if the versions were merged. Conflicts result from columns in two versions of the same table having different properties. For example, if both versions of the view have a table named *Demographics* that contains a variable named *Gender* but the variable is defined for the Analytic environment in one version and the Operational environment in the other, a conflict in the environment type would occur should the versions be merged. Column properties that may have conflicting definitions include the following:

- Column type
- Environment
- Direction
- Description

The results from this operation can be used to define resolutions for conflicts used in an actual version merge.

#### **Input fields**

The following table lists the input fields for the *getAvMergeConflicts* operation.

Table 4-1  
*Fields for getAvMergeConflicts*

Field	Type/Valid Values	Description
preferredVersionUri	string	
mergeVersionUri	string	

#### **Return information**

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

Table 4-2  
*Return Value*

Type	Description
avTableConflict[]	This element describes merge conflicts for one table of the Application View.

**Java example**

To identify definitions that conflict in two different versions of an Application View, supply the `getAvMergeConflicts` operation with two strings corresponding to the uniform resource identifiers for the view versions.

```
String prefVersionURI =
    "spsscr:///?id=0a010a07e80375f70000011effb874f9ab38#m.4:2009-02-02 14:34:50.407";
String mergeVersionURI =
    "spsscr:///?id=0a010a07e80375f70000011effb874f9ab38#m.1:2009-02-02 14:32:13.917";
AvTableConflict[] avConflicts = stub.getAvMergeConflicts(prefVersionURI, mergeVersionURI);
```

The operation returns an array of `AvTableConflict` objects containing conflict information organized by tables in the view. Use the `getName` method for any array entry to access the name of the table containing conflicts.

The `getAvColumnConflict` method for a table conflict object returns an array of `AvColumnConflict` objects corresponding to conflicts between columns in the table. For any entry in this array, the `getAvColumnMeta` method returns an array of `AvColumnMeta` objects containing metadata for the individual columns that conflict.

The `getVersionUri` method reports the URI of the view containing the particular column definition. Use the `getColumnMeta` method for a column metadata object to obtain an `AvColumnMetaType` object from which the individual metadata values can be accessed. The `getName` method returns the name of the column and the `getType` method returns its type.

Additional metadata for view columns can also be accessed. The `getSupportedEnvironments` method returns an array of `PevEnvironmentType` objects indicating the environments in which the column is used. Moreover, the `getDirection` method returns a `ColumnDirectionType` object from which the defined column directions can be obtained.

```
for (int i = 0; i < avConflicts.length; i++) {
    System.out.println(avConflicts[i].getName() +
        " table has the following conflicts:");
    AvColumnConflict[] colConflicts = avConflicts[i].getAvColumnConflict();
    for (int j = 0; j < colConflicts.length; j++) {
        AvColumnMeta[] colMeta = colConflicts[j].getAvColumnMeta();
        for (int k = 0; k < colMeta.length; k++) {
            AvColumnMetaType colMetaType = colMeta[k].getColumnMeta();
            System.out.println(colMeta[k].getVersionUri() +
                " has a column named " + colMetaType.getName() +
                " of type " + colMetaType.getType().toString());
            PevEnvironmentType[] envTypes = colMetaType.getSupportedEnvironments().getValue();
            System.out.println("Supported environments include:");
            for (int m = 0; m < envTypes.length; m++) {
                System.out.println(envTypes[m].toString());
            }
            Direction[] direct = colMetaType.getDirection().getValue();
            System.out.println("The column is used as:");
            for (int n = 0; n < direct.length; n++) {
                System.out.println(direct[n].toString());
            }
        }
    }
}
```

```
}

```

For web service clients based on JAX-WS, replace the arrays in the sample with List collections and update the array processing accordingly. For example:

```
String prefVersionURI =
    "spsscr:///?id=0a010a07e80375f70000011effb874f9ab38#m.4:2009-02-02 14:34:50.407";
String mergeVersionURI =
    "spsscr:///?id=0a010a07e80375f70000011effb874f9ab38#m.1:2009-02-02 14:32:13.917";
List<AvTableConflict> avConflictList = stub.getAvMergeConflicts(prefVersionURI, mergeVersionURI);
for (AvTableConflict avConflict : avConflictList)
{
    System.out.println(avConflict.getName() + " table has the following conflicts:");
    List<AvColumnConflict> colConflictList = avConflict.getAvColumnConflict();
    for (AvColumnConflict colConflict : colConflictList)
    {
        List<AvColumnMeta> colMetaList = colConflict.getAvColumnMeta();
        for (AvColumnMeta colMeta : colMetaList)
        {
            AvColumnMetaType colMetaType = colMeta.getColumnMeta();
            System.out.println(colMeta.getVersionUri() +
                " has a column named " + colMetaType.getName() +
                " of type " + colMetaType.getType().toString());
            List<PevEnvironmentType> envTypeList = colMetaType.getSupportedEnvironments().getValue();
            System.out.println("Supported environments include:");
            for (PevEnvironmentType envType : envTypeList)
            {
                System.out.println(envType.toString());
            }
            List<Direction> directList = colMetaType.getDirection().getValue();
            System.out.println("The column is used as:");
            for (Direction direct : directList)
            {
                System.out.println(direct.toString());
            }
        }
    }
}
}
}
```

### **SOAP request example**

Client invocation of the `getAvMergeConflicts` 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
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <wsse:Username>Native/admin</wsse:Username>
  <wsse:Password
    wsse:Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
      #PasswordText">pass</wsse:Password>
  <wsse:Nonce>of0ShsZMIgHcdD0o6A8PkQ==</wsse:Nonce>
  <wsu:Created
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    >2009-01-08T20:36:10Z</wsu:Created>
  </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>
  <getAvMergeConflicts xmlns="http://xml.spss.com/pev/remote">
    <preferredVersionUri>spsscr:///id=0a010a07e80375f70000011effb874f9ab38#m.4:2009-02-02
      14:34:50.407</preferredVersionUri>
    <mergeVersionUri>spsscr:///id=0a010a07e80375f70000011effb874f9ab38#m.1:2009-02-02
      14:32:13.917</mergeVersionUri>
  </getAvMergeConflicts>
</soapenv:Body>
</soapenv:Envelope>

```

### **SOAP response example**

The server responds to a `getAvMergeConflicts` 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>
    <getAvMergeConflictsResponse xmlns="http://xml.spss.com/pev/remote">
      <ns1:avTableConflict name="Table1" xmlns:ns1="http://xml.spss.com/pev">
        <ns1:avColumnConflict>
          <ns1:avColumnMeta>
            <ns1:versionUri>spsscr:///id=0a010a07e80375f70000011effb874f9ab38#m.4:2009-02-02
              14:34:50.407</ns1:versionUri>
            <ns1:columnMeta name="Col1" type="boolean">
              <ns1:supportedEnvironments>
                <ns1:value>analytic</ns1:value>
                <ns1:value>operational</ns1:value>
                <ns1:value>reporting</ns1:value>
              </ns1:supportedEnvironments>
              <ns1:direction>
                <ns1:value>target</ns1:value>
              </ns1:direction>
            </ns1:columnMeta>
          </ns1:avColumnMeta>
        </ns1:avColumnConflict>
      </ns1:avTableConflict>
    </getAvMergeConflictsResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

```

        </ns1:direction>
      </ns1:columnMeta>
    </ns1:avColumnMeta>
  <ns1:avColumnMeta>
    <ns1:versionUri>spsscr:///?id=0a010a07e80375f70000011effb874f9ab38#m.1:2009-02-02
      14:32:13.917</ns1:versionUri>
    <ns1:columnMeta name="Col1" type="boolean">
      <ns1:supportedEnvironments>
        <ns1:value>analytic</ns1:value>
        <ns1:value>operational</ns1:value>
        <ns1:value>reporting</ns1:value>
      </ns1:supportedEnvironments>
      <ns1:direction>
        <ns1:value>predictor</ns1:value>
      </ns1:direction>
    </ns1:columnMeta>
  </ns1:avColumnMeta>
</ns1:avColumnConflict>
</ns1:avTableConflict>
</getAvMergeConflictsResponse>
</soapenv:Body>
</soapenv:Envelope>

```

### ***The getEvMergeConflicts operation***

Identifies differences in the definitions for two specified versions of an Enterprise View that would result in conflicts if the versions were merged. Conflicts result from columns in two versions of the same table having different properties. For example, if both versions of the view have a table named *Demographics* that contains a variable named *Gender* but the variable is defined as an integer in one version and a string in the other, a conflict in the column type would occur should the versions be merged. Column properties that may have conflicting definitions include the following:

- Column type
- Minimum value
- Maximum value
- Category values
- Description

The results from this operation can be used to define resolutions for conflicts used in an actual version merge.

#### ***Input fields***

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

Table 4-3  
Fields for `getEvMergeConflicts`

Field	Type/Valid Values	Description
<code>preferredVersionMarker</code>	string	
<code>mergeVersionMarker</code>	string	

### Return information

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

Table 4-4  
Return Value

Type	Description
<code>evTableConflict[]</code>	This element describes merge conflicts for one table of the Enterprise View.

### Java example

To identify definitions that conflict in two different versions of an Enterprise View, supply the `getEvMergeConflicts` operation with two strings corresponding to the markers for the view versions.

```
String prefVersion = "3:2009-02-02 14:16:33.28";
String mergeVersion = "2:2009-02-02 14:15:39.57";
EvTableConflict[] evConflicts = stub.getEvMergeConflicts(prefVersion, mergeVersion);
```

The operation returns an array of `EvTableConflict` objects containing conflict information organized by tables in the view. Use the `getName` method for any array entry to access the name of the table containing conflicts.

The `getEvColumnConflict` method for a table conflict object returns an array of `EvColumnConflict` objects corresponding to conflicts between columns in the table. For any entry in this array, the `getEvColumnMeta` method returns an array of `EvColumnMeta` objects containing metadata for the individual columns that conflict.

The `getMarker` method reports the version of the view containing the particular column definition. Use the `getColumnMeta` method for a column metadata object to obtain an `EvColumnMetaType` object from which the individual metadata values can be accessed. The `getName` method returns the name of the column and the `getType` method returns its type.

```
for (int i = 0; i < evConflicts.length; i++) {
    System.out.println(evConflicts[i].getName() +
        " table has the following conflicts:");
    EvColumnConflict[] colConflicts = evConflicts[i].getEvColumnConflict();
    for (int j = 0; j < colConflicts.length; j++) {
        EvColumnMeta[] colMeta = colConflicts[j].getEvColumnMeta();
        for (int k = 0; k < colMeta.length; k++) {
            EvColumnMetaType colMetaType = colMeta[k].getColumnMeta();
            System.out.println("Version " + colMeta[k].getMarker() +
                " has a column named " + colMetaType.getName() +
                " of type " + colMetaType.getType().toString());
        }
    }
}
```

```

    }
}

```

For web service clients based on JAX-WS, replace the arrays in the sample with List collections and update the array processing accordingly. For example:

```

String prefVersion = "3:2009-02-02 14:16:33.28";
String mergeVersion = "2:2009-02-02 14:15:39.57";
List<EvTableConflict> evConflictList = stub.getEvMergeConflicts(prefVersion, mergeVersion);
for (EvTableConflict evConflict : evConflictList)
{
    System.out.println(evConflict.getName() + " table has the following conflicts:");
    List<EvColumnConflict> colConflictList = evConflict.getEvColumnConflict();
    for (EvColumnConflict colConflict : colConflictList)
    {
        List<EvColumnMeta> colMetaList = colConflict.getEvColumnMeta();
        for (EvColumnMeta colMeta : colMetaList)
        {
            EvColumnMetaType colMetaType = colMeta.getColumnMeta();
            System.out.println("Version " + colMeta.getMarker() +
                " has a column named " + colMetaType.getName() +
                " of type " + colMetaType.getType().toString());
        }
    }
}
}
}

```

### **SOAP request example**

Client invocation of the `getEvMergeConflicts` 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
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>Native/admin</wsse:Username>
        <wsse:Password
          wsse:Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
            #PasswordText">pass</wsse:Password>
        <wsse:Nonce>of0ShsZMIgHcdD0o6A8PkQ==</wsse:Nonce>
        <wsu:Created
          xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
          >2009-01-08T20:36:10Z</wsu:Created>
        </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>
  <getEvMergeConflicts xmlns="http://xml.spss.com/pev/remote">
    <preferredVersionMarker>3:2009-02-02 14:16:33.28</preferredVersionMarker>
    <mergeVersionMarker>2:2009-02-02 14:15:39.57</mergeVersionMarker>
  </getEvMergeConflicts>
</soapenv:Body>
</soapenv:Envelope>

```

### **SOAP response example**

The server responds to a `getEvMergeConflicts` 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>
    <getEvMergeConflictsResponse xmlns="http://xml.spss.com/pev/remote">
      <ns1:evTableConflict name="Table1" xmlns:ns1="http://xml.spss.com/pev">
        <ns1:evColumnConflict>
          <ns1:evColumnMeta>
            <ns1:marker>3:2009-02-02 14:16:33.28</ns1:marker>
            <ns1:columnMeta name="Col1" type="string"/>
          </ns1:evColumnMeta>
          <ns1:evColumnMeta>
            <ns1:marker>2:2009-02-02 14:15:39.57</ns1:marker>
            <ns1:columnMeta name="Col1" type="boolean"/>
          </ns1:evColumnMeta>
        </ns1:evColumnConflict>
      </ns1:evTableConflict>
    </getEvMergeConflictsResponse>
  </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 4-5  
Return Value

Type	Description
string	

### **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/pev/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/pev/remote">
      <version>4.20.000</version>
    </getVersionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

## **The performAvMerge operation**

Combines two specified versions of an Application View into a new composite version. If the tables in the two versions have conflicting definitions, resolutions for the conflicts can be supplied when calling the operation. Conflicts can be identified using the [getAvMergeConflicts](#) operation. For any conflict that does not have a specified resolution, the definition from the preferred version of the view is used in the resulting merged version.

**Input fields**

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

Table 4-6  
Fields for `performAvMerge`

Field	Type/Valid Values	Description
<code>preferredVersionUri</code>	string	
<code>mergeVersionUri</code>	string	
<code>avTableResolution</code>	<code>avTableResolution[]</code>	This element describes user specified merge resolutions for one table of the Application View.

**Return information**

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

Table 4-7  
Return Value

Type	Description
<code>mergeResult</code>	This element describes the result of an EV or AV merge.

**Java example**

To merge two different versions of an Application View,:

1. Use the `getAvMergeConflicts` operation to identify conflicts between the versions being merged. [For more information, see the topic `The getAvMergeConflicts` operation on p. 16.](#)
2. Create an array of `AvTableResolution` objects to define the resolutions for any table conflicts.
3. Use the `setName` method to define the name of the table for a resolution object.
4. Create an array of `AvColumnMetaType` objects to define the table columns for the conflict resolution.
5. Assign properties for the columns as needed. For example, use the `setName` and `setType` methods to define the name and type for the column.
6. Create a `ColumnEnvironmentType` object and define the environments for the column. Use the `setSupportedEnvironments` method to assign the object to the column metadata object.
7. Create a `ColumnDirectionType` object and define the directions for the column. Use the `setDirection` method to assign the object to the column metadata object.
8. Use the `setColumnMeta` method to assign the column definition to a resolution object.
9. Supply the `performAvMerge` operation with two strings corresponding to the URIs for the view versions and the table resolution object array.

The operation returns a `MergeResult` object containing the URI for merged version of the view. Use the `getNewVersionUri` method to access this value.

The following sample creates a new merged version of a view in which the conflict for the column *Col1* in table *Table1* is resolved by defining the desired type, environments, and direction for the column.

```
String prefVersionURI =
    "spsscr:///?id=0a010a07e80375f70000011effb874f9ab38#m.4:2009-02-02 14:34:50.407";
String mergeVersionURI =
    "spsscr:///?id=0a010a07e80375f70000011effb874f9ab38#m.1:2009-02-02 14:32:13.917";

AvTableResolution tableResolution = new AvTableResolution();
tableResolution.setName("Table1");
AvColumnMetaType colMeta = new AvColumnMetaType();
colMeta.setName("Col1");
colMeta.setType(PevDataType.BOOLEAN);

ColumnEnvironmentType envType = new ColumnEnvironmentType();
envType.addValue(PevEnvironmentType.ANALYTIC);
envType.addValue(PevEnvironmentType.OPERATIONAL);
envType.addValue(PevEnvironmentType.REPORTING);
colMeta.setSupportedEnvironments(envType);

ColumnDirectionType directionType = new ColumnDirectionType();
directionType.addValue(Direction.PREDICTOR);
colMeta.setDirection(directionType);

tableResolution.setColumnMeta(colMeta);
MergeResult result = stub.performAvMerge(prefVersionURI, mergeVersionURI, tableResolution);

System.out.println("URI for the merged version is: " + result.getNewVersionUri());
```

For web service clients based on JAX-WS, replace the arrays in the sample with List collections and update the array processing accordingly. For example:

```
String prefVersionURI =
    "spsscr:///?id=0a010a07e80375f70000011effb874f9ab38#m.4:2009-02-02 14:34:50.407";
String mergeVersionURI =
    "spsscr:///?id=0a010a07e80375f70000011effb874f9ab38#m.1:2009-02-02 14:32:13.917";

AvTableResolution tableResolution = new AvTableResolution();
tableResolution.setName("Table1");
AvColumnMetaType colMeta = new AvColumnMetaType();
colMeta.setName("Col1");
colMeta.setType(PevDataType.BOOLEAN);

ColumnEnvironmentType envType = new ColumnEnvironmentType();
envType.getValue().add(PevEnvironmentType.ANALYTIC);
envType.getValue().add(PevEnvironmentType.OPERATIONAL);
envType.getValue().add(PevEnvironmentType.REPORTING);
colMeta.setSupportedEnvironments(envType);

ColumnDirectionType directionType = new ColumnDirectionType();
```



```

directionType.getValue().add(Direction.PREDICTOR);
colMeta.setDirection(directionType);

tableResolution.getColumnMeta().add(colMeta);
MergeResult result = stub.performAvMerge(prefVersionURI, mergeVersionURI, tableResolution);

System.out.println("URI for the merged version is: " + result.getNewVersionUri());

```

### **SOAP request example**

Client invocation of the `performAvMerge` 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
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>Native/admin</wsse:Username>
        <wsse:Password
          wsse:Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
            #PasswordText">pass</wsse:Password>
        <wsse:Nonce>of0ShsZMIgHcdD0o6A8PkQ==</wsse:Nonce>
        <wsu:Created
          xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
          >2009-01-08T20:36:10Z</wsu:Created>
        </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>
    <performAvMerge xmlns="http://xml.spss.com/pev/remote">
      <preferredVersionUri>spsscr:///id=0a010a07e80375f70000011effb874f9ab38#m.4:2009-02-02
        14:34:50.407</preferredVersionUri>
      <mergeVersionUri>spsscr:///id=0a010a07e80375f70000011effb874f9ab38#m.1:2009-02-02
        14:32:13.917</mergeVersionUri>
      <ns1:avTableResolution name="Table1" xmlns:ns1="http://xml.spss.com/pev">
        <ns1:columnMeta name="Col1" type="boolean">
          <ns1:supportedEnvironments>
            <ns1:value>analytic</ns1:value>
            <ns1:value>operational</ns1:value>
            <ns1:value>reporting</ns1:value>
          </ns1:supportedEnvironments>
          <ns1:direction>
            <ns1:value>predictor</ns1:value>
          </ns1:direction>
        </ns1:columnMeta>
      </ns1:avTableResolution>
    </performAvMerge>
  </soapenv:Body>
</soapenv:Envelope>

```

```

        </ns1:direction>
        <ns1:description/>
        </ns1:columnMeta>
    </ns1:avTableResolution>
</performAvMerge>
</soapenv:Body>
</soapenv:Envelope>

```

## The performEvMerge operation

Combines two specified versions of an Enterprise View into a new composite version. If the tables in the two versions have conflicting definitions, resolutions for the conflicts can be supplied when calling the operation. Conflicts can be identified using the [getEvMergeConflicts](#) operation. For any conflict that does not have a specified resolution, the definition from the preferred version of the view is used in the resulting merged version.

### Input fields

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

Table 4-8  
Fields for performEvMerge

Field	Type/Valid Values	Description
preferredVersionMarker	string	
mergeVersionMarker	string	
evTableResolution	evTableResolution[]	This element describes user specified merge resolutions for one table of the Enterprise View.

### Return information

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

Table 4-9  
Return Value

Type	Description
mergeResult	This element describes the result of an EV or AV merge.

### Java example

To merge two different versions of an Enterprise View:

1. Use the `getEvMergeConflicts` operation to identify conflicts between the versions being merged. [For more information, see the topic The getEvMergeConflicts operation on p. 20.](#)
2. Create an array of `EvTableResolution` objects to define the resolutions for any table conflicts.
3. Use the `setName` method to define the name of the table for a resolution object.

4. Create an array of `EvColumnMetaType` objects to define the table columns for the conflict resolution.
5. Assign properties for the columns as needed. For example, use the `setName` and `setType` methods to define the name and type for the column.
6. Use the `setColumnMeta` method to assign the column definition to a resolution object.
7. Supply the `performEvMerge` operation with two strings corresponding to the markers for the view versions and the table resolution object array.

The operation returns a `MergeResult` object containing the URI for merged version of the view. Use the `getNewVersionUri` method to access this value.

The following sample creates a new merged version of a view in which the conflict for the column `Col1` in table `Table1` is resolved by setting the type to boolean.

```
String prefVersion = "3:2009-02-02 14:16:33.28";
String mergeVersion = "2:2009-02-02 14:15:39.57";

EvTableResolution tableResolution = new EvTableResolution();
tableResolution.setName("Table1");
EvColumnMetaType colMeta = new EvColumnMetaType();
colMeta.setName("Col1");
colMeta.setType(PevDataType.BOOLEAN);
tableResolution.setColumnMeta(colMeta);

MergeResult result = stub.performEvMerge(prefVersion, mergeVersion, tableResolution);

System.out.println("URI for the merged version is: " + result.getNewVersionUri());
```

For web service clients based on JAX-WS, replace the arrays in the sample with `List` collections and update the array processing accordingly. For example:

```
String prefVersion = "3:2009-02-02 14:16:33.28";
String mergeVersion = "2:2009-02-02 14:15:39.57";

EvTableResolution tableResolution = new EvTableResolution();
tableResolution.setName("Table1");
EvColumnMetaType colMeta = new EvColumnMetaType();
colMeta.setName("Col1");
colMeta.setType(PevDataType.BOOLEAN);
tableResolution.getColumnMeta().add(colMeta);

MergeResult result = stub.performEvMerge(prefVersion, mergeVersion, tableResolution);

System.out.println("URI for the merged version is: " + result.getNewVersionUri());
```

### ***SOAP request example***

Client invocation of the `performEvMerge` 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
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>Native//admin</wsse:Username>
        <wsse:Password
          wsse:Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
            #PasswordText">pass</wsse:Password>
        <wsse:Nonce>of0ShsZMIgHcdD0o6A8PkQ==</wsse:Nonce>
        <wsu:Created
          xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
          >2009-01-08T20:36:10Z</wsu:Created>
        </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>
      <performEvMerge xmlns="http://xml.spss.com/pev/remote">
        <preferredVersionMarker>3:2009-02-02 14:16:33.28</preferredVersionMarker>
        <mergeVersionMarker>2:2009-02-02 14:15:39.57</mergeVersionMarker>
        <ns1:evTableResolution name="Table1" xmlns:ns1="http://xml.spss.com/pev">
          <ns1:columnMeta name="Col1" type="boolean">
            <ns1:minimum/>
            <ns1:maximum/>
            <ns1:description/>
          </ns1:columnMeta>
        </ns1:evTableResolution>
      </performEvMerge>
    </soapenv:Body>
  </soapenv:Envelope>

```

## The validateObject operation

### Input fields

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

Table 4-10  
Fields for validateObject

Field	Type/Valid Values	Description
objectReference	labeledUriType	
realTimeUsage	realTimeUsage	

**Return information**

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

Table 4-11  
Return Value

Type	Description
validationResult	This element describes the result of a PEV object validation.

**Metadata port type**

The Metadata port type includes operations used for working with information about variables.

**The `getCategories` operation**

Retrieves the category values and labels for a specified variable in a data source.

**Input fields**

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

Table 4-12  
Fields for `getCategories`

Field	Type/Valid Values	Description
categoryQuery	categoryQuery	This type represents a query used to look up a list of categories for a specific variable.

**Return information**

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

Table 4-13  
Return Value

Type	Description
categoryType[]	

**Java example**

To retrieve information about the categories for a variable:

1. Create a `CategoryQuery` object.
2. Provide the `setLocation` method with a string corresponding to the location of the IBM® SPSS® Statistics data file as a URI.
3. Supply the `setVariableName` method with a string denoting the variable in the data file.
4. Supply the `getCategories` operation with the query object.

The operation returns an array of `CategoryType` objects. Each object in the array corresponds to a category for the variable. Use the `getId` and `getLabel` methods to obtain the category values and associated labels.

The following sample retrieves the category information for the *region* variable in a data file.

```
CategoryQuery catQuery = new CategoryQuery();
catQuery.setLocation("spsscr:///id=0a010a07e123f4280000011f5babe020845b");
catQuery.setVariableName("region");
CategoryType[] categories = stub.getCategories(catQuery);

for (int i = 0; i < categories.length; i++) {
    System.out.println("Category " + categories[i].getId() +
        " = " + categories[i].getLabel());
}
```

For web service clients based on JAX-WS, replace the arrays in the sample with `List` collections and update the array processing accordingly. For example:

```
CategoryQuery catQuery = new CategoryQuery();
catQuery.setLocation("spsscr:///id=0a010a07e123f4280000011f5babe020845b");
catQuery.setVariableName("region");
List<CategoryType> categoryList = stub.getCategories(catQuery);

for (CategoryType category : categoryList)
{
    System.out.println("Category " + category.getId() +
        " = " + category.getLabel());
}
```

### **SOAP request example**

Client invocation of the `getCategories` 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
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>Native/admin</wsse:Username>
        <wsse:Password
          wsse:Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
            #PasswordText">pass</wsse:Password>
        <wsse:Nonce>of0ShsZMIgHcdD0o6A8PkQ==</wsse:Nonce>
        <wsu:Created
          xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
```

```

    >2009-01-08T20:36:10Z</wsu:Created>
  </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>
  <getCategories xmlns="http://xml.spss.com/pev/remote">
    <categoryQuery>
      <location xmlns="http://xml.spss.com/pev"
        >spsscr:///?id=0a010a07e123f4280000011f5babe020845b</location>
      <variableName xmlns="http://xml.spss.com/pev">region</variableName>
    </categoryQuery>
  </getCategories>
</soapenv:Body>
</soapenv:Envelope>

```

### ***SOAP response example***

The server responds to a `getCategories` 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>
    <getCategoriesResponse xmlns="http://xml.spss.com/pev/remote">
      <ns1:category xmlns:ns1="http://xml.spss.com/pev">
        <ns1:id>1.0</ns1:id>
        <ns1:label>North East</ns1:label>
      </ns1:category>
      <ns2:category xmlns:ns2="http://xml.spss.com/pev">
        <ns2:id>2.0</ns2:id>
        <ns2:label>South East</ns2:label>
      </ns2:category>
      <ns3:category xmlns:ns3="http://xml.spss.com/pev">
        <ns3:id>3.0</ns3:id>
        <ns3:label>West</ns3:label>
      </ns3:category>
    </getCategoriesResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

### ***The getCustomAttributes operation***

Retrieves any custom attributes defined for a specified variable in a IBM® SPSS® Statistics data source.

**Input fields**

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

Table 4-14  
*Fields for `getCustomAttributes`*

Field	Type/Valid Values	Description
<code>customAttributesQuery</code>	<code>customAttributesQuery</code>	This type represents a query used to look up a list of custom attributes for a specific variable.

**Return information**

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

Table 4-15  
*Return Value*

Type	Description
<code>customAttributesType[]</code>	

**Java example**

To retrieve information about the custom attributes for a variable:

1. Create a `CustomAttributesQuery` object.
2. Provide the `setLocation` method with a string corresponding to the location of the SPSS Statistics data file as a URI.
3. Supply the `setVariableName` method with a string denoting the variable in the data file.
4. Supply the `getCustomAttributes` operation with the query object.

The operation returns an array of `CustomAttributesType` objects. Each object in the array corresponds to a custom attribute for the variable. Use the `getName` method to obtain the attribute name. The `getValues` method returns an array of strings corresponding to the attribute values.

The following sample retrieves the custom attribute information for the `region` variable in a data file.

```
CustomAttributesQuery custAttQuery = new CustomAttributesQuery();
custAttQuery.setLocation("spsscr:///id=0a010a07e123f4280000011f5babe020845b");
custAttQuery.setVariableName("region");
CustomAttributesType[] custAtts = stub.getCustomAttributes(custAttQuery);
```

```
for (int i = 0; i < custAtts.length; i++) {
    System.out.println("Attribute " + custAtts[i].getName() +
        " has the following values:");
    String[] valArray = custAtts[i].getValues();
    for (int j = 0; j < valArray.length; j++) {
        System.out.println(valArray[j]);
    }
}
```



For web service clients based on JAX-WS, replace the arrays in the sample with List collections and update the array processing accordingly. For example:

```

CustomAttributesQuery custAttQuery = new CustomAttributesQuery();
custAttQuery.setLocation("spsscr:///?id=0a010a07e123f4280000011f5babe020845b");
custAttQuery.setVariableName("region");
List<CustomAttributesType> custAttList = stub.getCustomAttributes(custAttQuery);

for (CustomAttributesType custAtt : custAttList)
{
    System.out.println("Attribute " + custAtt.getName() +
        " has the following values:");
    List<String> valList = custAtt.getValues();
    for (String val : valList)
    {
        System.out.println(val);
    }
}

```

### **SOAP request example**

Client invocation of the `getCustomAttributes` 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
                xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
                <wsse:Username>Native/admin</wsse:Username>
                <wsse:Password
                    wsse:Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
                    #PasswordText">pass</wsse:Password>
                <wsse:Nonce>of0ShsZMIgHcdD0o6A8PkQ==</wsse:Nonce>
                <wsu:Created
                    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                    >2009-01-08T20:36:10Z</wsu:Created>
                </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>
        <getCustomAttributes xmlns="http://xml.spss.com/pev/remote">
            <customAttributesQuery>

```

```

    <location xmlns="http://xml.spss.com/pev"
      >spsscr:///id=0a010a07e123f4280000011f5babe020845b</location>
    <variableName xmlns="http://xml.spss.com/pev">region</variableName>
  </customAttributesQuery>
</getCustomAttributes>
</soapenv:Body>
</soapenv:Envelope>

```

### **SOAP response example**

The server responds to a `getCustomAttributes` 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>
    <getCustomAttributesResponse xmlns="http://xml.spss.com/pev/remote">
      <ns1:customAttributes xmlns:ns1="http://xml.spss.com/pev">
        <ns1:name>myAtt1</ns1:name>
        <ns1:values>Value1</ns1:values>
      </ns1:customAttributes>
      <ns2:customAttributes xmlns:ns2="http://xml.spss.com/pev">
        <ns2:name>myAtt2</ns2:name>
        <ns2:values>a</ns2:values>
        <ns2:values>b</ns2:values>
        <ns2:values>c</ns2:values>
      </ns2:customAttributes>
    </getCustomAttributesResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

## **The `getVariableMetadata` operation**

Retrieves the metadata information for all variables in a specified data source.

### **Input fields**

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

Table 4-16  
Fields for `getVariableMetadata`

Field	Type/Valid Values	Description
metadataQuery	metadataQuery	This type represents a query used to specify a location (URI) and table.

**Return information**

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

Table 4-17  
Return Value

Type	Description
<code>pevMetaDataType</code>	This type represents the result returned from execution of a meta-data query

**Java example**

To retrieve the metadata information for variables:

1. Create a `MetadataQuery` object.
2. Provide the `setLocation` method with a string corresponding to the location of the data source as a URI.
3. If the data source is a data provider definition, use the optional `setTableName` method to specify the name of the table containing the variables of interest.
4. Supply the `getVariableMetadata` operation with the query object.

Use the `getQualifier` method for the returned `PevMetaDataType` object to obtain an array of `QualifierMetaType` objects containing the variable metadata. For any entry in the array, the `getVariableMetaType` method returns the metadata as an array of `VariableMetaType` objects.

A `VariableMetaType` object provides a variety of accessor methods for retrieving specific metadata values. For example, the `getName` method returns the variable name. The `getDataType` method returns the variable data type. Use the accessors as needed to obtain the desired metadata information.

The following sample retrieves the variable metadata for the *CQ-DPD* data provider definition.

```
MetadataQuery mdQuery = new MetadataQuery();
mdQuery.setLocation("spsscr:///CQ-DPD");
PevMetaDataType mdType = stub.getVariableMetadata(mdQuery);
```

```
QualifierMetaType[] qualifier = mdType.getQualifier();
for (int i = 0; i < qualifier.length; i++) {
    System.out.println(qualifier[i].getName() +
        " has the following variables:");
    VariableMetaType[] varArray = qualifier[i].getVariableMetaType();
    System.out.println("NAME\tVAR TYPE\tDATA TYPE\n");
    for (int j = 0; j < varArray.length; j++) {
        System.out.println(varArray[j].getName() + "\t" +
            varArray[j].getVarType().toString() + "\t" +
            varArray[j].getDataType().toString() + "\n");
    }
}
```

For web service clients based on JAX-WS, replace the arrays in the sample with List collections and update the array processing accordingly. For example:

```
List<QualifierMetaType> qualifierList = mdType.getQualifier();
```

### **SOAP request example**

Client invocation of the `getVariableMetadata` 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
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>Native//admin</wsse:Username>
        <wsse:Password
          wsse:Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
            #PasswordText">pass</wsse:Password>
        <wsse:Nonce>of0ShsZMIgHcdD0o6A8PkQ==</wsse:Nonce>
        <wsu:Created
          xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
          >2009-01-08T20:36:10Z</wsu:Created>
        </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>
      <getVariableMetadata xmlns="http://xml.spss.com/pev/remote">
        <metadataQuery>
          <location xmlns="http://xml.spss.com/pev">spsscr:///CQ-DPD</location>
        </metadataQuery>
      </getVariableMetadata>
    </soapenv:Body>
  </soapenv:Envelope>
```

### **SOAP response example**

The server responds to a `getVariableMetadata` 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>
  <getVariableMetadataResponse xmlns="http://xml.spss.com/pev/remote">
    <ns1:return xmlns:ns1="http://xml.spss.com/pev">
      <ns1:location>spsscr:///CQ-DPD</ns1:location>
      <ns1:qualifier>
        <ns1:name>CQ defects</ns1:name>
        <ns1:description>CQ defects</ns1:description>
        <ns1:variableMetaData>
          <ns1:name>overall_status</ns1:name>
          <ns1:varType>simple_variable</ns1:varType>
          <ns1:displayName>overall_status</ns1:displayName>
          <ns1:dataType>string</ns1:dataType>
          <ns1:size>0</ns1:size>
        </ns1:variableMetaData>
        <ns1:variableMetaData>
          <ns1:name>component_feature</ns1:name>
          <ns1:varType>simple_variable</ns1:varType>
          <ns1:displayName>component_feature</ns1:displayName>
          <ns1:dataType>string</ns1:dataType>
          <ns1:size>0</ns1:size>
        </ns1:variableMetaData>
        <ns1:variableMetaData>
          <ns1:name>userimpact</ns1:name>
          <ns1:varType>simple_variable</ns1:varType>
          <ns1:displayName>userimpact</ns1:displayName>
          <ns1:dataType>string</ns1:dataType>
          <ns1:size>0</ns1:size>
        </ns1:variableMetaData>
        <ns1:variableMetaData>
          <ns1:name>project</ns1:name>
          <ns1:varType>simple_variable</ns1:varType>
          <ns1:displayName>project</ns1:displayName>
          <ns1:dataType>integer</ns1:dataType>
          <ns1:size>0</ns1:size>
        </ns1:variableMetaData>
      </ns1:qualifier>
    </ns1:return>
  </getVariableMetadataResponse>
</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 4-18  
Return Value

Type	Description
string	

### **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/pev/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/pev/remote">
      <version>4.20.000</version>
    </getVersionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

## **Search port type**

The Search port type includes operations used for identifying Application View and real-time Data Provider Definition resources that match specified criteria.

## The `getCompatibleAv` operation

Returns a list of all Application View resources in the system that match specified criteria. The criteria that must be met by the returned views includes the following:

- Support for specified environments
- Presence of columns having specified names and types

### Input fields

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

Table 4-19  
Fields for `getCompatibleAv`

Field	Type/Valid Values	Description
environment	supportedEnvironment	
dataItem	pevCatalogColumn[]	This represents the information in the PEV Catalog Column entry. The catalog is managed internally by PEV. Entries are created and deleted as Enterprise View columns are created and deleted. The fields of this object uniquely define (primary composite key) a column entry within a table of the catalog. The catalog is global to all versions of the Enterprise View. One can not change information in a catalog entry. One can only create or delete EV Tables (PEV will manage the entries). In other words, one can not rename an Enterprise View column. Instead, a new column must be created. This limitation is reasonable since any objects (Clem streams, etc) that reference a table name in a SELECT statement would no longer be valid anyway and must also be updated.

### Return information

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

Table 4-20  
Return Value

Type	Description
resourceTablesByLabel[]	

**Java example**

To retrieve Application View resources in the system that are compatible with specified criteria:

1. Create a `SupportedEnvironment` object.
2. Provide the `setPevEnvironment` method with a `PevEnvironmentType` value denoting the Application View environment of interest.
3. Create an array of `PevCatalogColumn` objects for the data items that must exist in any matching Application View resources. For each object, use the `setName` and `setType` methods to specify the name and data type for the item.
4. Supply the `getCompatibleAv` operation with the environment and data item objects.

The operation returns an array of `ResourceTablesByLabel` objects containing information about Application View resources matching the input criterion. The `getName`, `getPath`, and `getBaseUri` methods return the name, resource path, and URI for a specific resource in the array.

Different versions of Application View resources may contain different tables. Consequently, the table information in a `ResourceTablesByLabel` object is organized by label. Use the `getTablesByLabel` method to obtain an array of `TablesByLabel` objects for tables in different Application View versions. For any entry in the array, the `getLabel` method returns the label of the Application View version containing the table. The names of the tables in the version that contain the data items of interest are returned as a string array using the `getTable` method.

The following sample retrieves Application View resources for the *analytic* environment that contain an integer field named *component*.

```
SupportedEnvironment env = new SupportedEnvironment();
env.setPevEnvironment(PevEnvironmentType.ANALYTIC);

PevCatalogColumn dataItem = new PevCatalogColumn();
dataItem.setName("component");
dataItem.setType(PevDataType.INTEGER);
ResourceTablesByLabel[] resTables = stub.getCompatibleAv(env, dataItem);

for (int i = 0; i < resTables.length; i++) {
    System.out.println("View " + resTables[i].getName() +
        "has a path of " + resTables[i].getPath() +
        "and contains the following tables:");
    TablesByLabel[] tabByLab = resTables[i].getTablesByLabel();
    for (int j = 0; j < tabByLab.length; j++) {
        System.out.println("LABEL = " + tabByLab[j].getLabel());
        String[] tables = tabByLab[j].getTable();
        for (int k = 0; k < tables.length; k++) {
            System.out.println(tables[k]);
        }
    }
}
```



For web service clients based on JAX-WS, replace the arrays in the sample with List collections and update the array processing accordingly. For example:

```
SupportedEnvironment env = new SupportedEnvironment();
env.setPevEnvironment(PevEnvironmentType.ANALYTIC);

PevCatalogColumn dataItem = new PevCatalogColumn();
dataItem.setName("component");
dataItem.setType(PevDataType.INTEGER);
List<ResourceTablesByLabel> resTablesList = stub.getCompatibleAv(env, dataItem);

for (ResourceTablesByLabel resTables : resTablesList)
{
    System.out.println("View " + resTables.getName() +
        " has a path of " + resTables.getPath() +
        " and contains the following tables:");
    List<TablesByLabel> tabByLabList = resTables.getTablesByLabel();
    for (TablesByLabel tabByLab : tabByLabList)
    {
        System.out.println("LABEL = " + tabByLab.getLabel());
        List<String> tablesList = tabByLab.getTable();
        for (String tables : tablesList)
        {
            System.out.println(tables);
        }
    }
}
}
```

### **SOAP request example**

Client invocation of the `getCompatibleAv` 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
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>Native/admin</wsse:Username>
        <wsse:Password
          wsse:Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
            #PasswordText">pass</wsse:Password>
        <wsse:Nonce>of0ShsZMIgHcdD0o6A8PkQ==</wsse:Nonce>
        <wsu:Created
          xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
          >2009-01-08T20:36:10Z</wsu:Created>
```

```

    </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>
  <getCompatibleAv xmlns="http://xml.spss.com/pev/remote">
    <environment pevEnvironment="analytic"/>
    <dataItem name="component" type="integer"/>
  </getCompatibleAv>
</soapenv:Body>
</soapenv:Envelope>

```

### ***SOAP response example***

The server responds to a `getCompatibleAv` 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>
    <getCompatibleAvResponse xmlns="http://xml.spss.com/pev/remote">
      <ns1:resourceTablesByLabel baseUri="spsscr:///id=0a010a07e123f4280000011f5babe0208305"
        name="CQ" path="/CQ" xmlns:ns1="http://xml.spss.com/pev">
        <ns1:tablesByLabel label="LATEST">
          <ns1:table>ClearQuest defects</ns1:table>
        </ns1:tablesByLabel>
      </ns1:resourceTablesByLabel>
    </getCompatibleAvResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

### ***The getCompatibleRtDpd operation***

Returns a list of all Data Provider Definition - Real Time resources in the system that meet specified criteria, including the following:

- Resources associated with a specified Application View
- Resources that include a specified table

#### ***Input fields***

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

Table 4-21  
Fields for `getCompatibleRtDpd`

Field	Type/Valid Values	Description
avReference	labeledUriType	
table	pevCatalogTable	This represents the information in a PEV Catalog Table entry. The catalog is managed internally by PEV. Entries are created and deleted as Enterprise View tables are created and deleted. The fields of this object uniquely define (primary composite key) a table entry. The catalog is global to all versions of the Enterprise View. One can not change information in a catalog entry. One can only create or delete EV Tables (PEV will manage the entries). In other words, one can not rename an Enterprise View table. Instead, a new table must be created. This limitation is reasonable since any consumers (Clem streams, etc) that reference a table name in a SELECT statement would no longer be valid anyway and must also be updated.

### Return information

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

Table 4-22  
Return Value

Type	Description
labeledResourceType[]	

### Java example

To retrieve real-time data provider definitions in the system that are compatible with an application view:

1. Create a `LabeledUriType` object.
2. Provide the `setBaseUri` method with a string corresponding to the location of the application view as a URI. Label or version markers should not be included in the URI. If a version other than *LATEST* is needed, use the `setLabel` method to define the label.
3. Create a `PevCatalogTable` object.
4. Provide the `setName` method with a string denoting the name of a table that must be included in any real-time data provider definition returned as a match.
5. Supply the `getCompatibleRtDpd` operation with the URI and table objects.

The operation returns an array of `LabeledResourceType` objects containing information about real-time data providers matching the input criterion. The `getName`, `getPath`, and `getLabel` methods return the name, resource path, and label for a specific provider in the array.

The following sample retrieves real-time data provider definitions for an application view that contain a *CQdefects* table.

```
LabeledUriType type = new LabeledUriType();
type.setBaseUri("spsscr:///id=0a010a077e08b4a40000011f5cabdad38098")

PevCatalogTable table = new PevCatalogTable();
table.setName("CQdefects");

LabeledResourceType[] resType = stub.getCompatibleRtDpd(type, table);

System.out.println("NAME\tPATH\tLABEL\n");
for (int i = 0; i < resType.length; i++) {
    System.out.println(resType[i].getName() + "\t" +
        resType[i].getPath() + "\t" + resType[i].getLabel() + "\n");
}
```

For web service clients based on JAX-WS, replace the arrays in the sample with `List` collections and update the array processing accordingly. For example:

```
LabeledUriType type = new LabeledUriType();
type.setBaseUri("spsscr:///id=0a010a077e08b4a40000011f5cabdad38098")

PevCatalogTable table = new PevCatalogTable();
table.setName("CQdefects");

List<LabeledResourceType> resTypeList = stub.getCompatibleRtDpd(type, table);
System.out.println("NAME\tPATH\tLABEL\n");
for (LabeledResourceType resType : resTypeList)
{
    System.out.println(resType.getName() + "\t" +
        resType.getPath() + "\t" + resType.getLabel() + "\n");
}
```

### **SOAP request example**

Client invocation of the `getCompatibleRtDpd` 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
```

```

xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
<wsse:Username>Native/admin</wsse:Username>
<wsse:Password
  wsse:Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
  #PasswordText">pass</wsse:Password>
<wsse:Nonce>of0ShsZMIgHcdD0o6A8PkQ==</wsse:Nonce>
<wsu:Created
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  >2009-01-08T20:36:10Z</wsu:Created>
</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>
  <getCompatibleRtDpd xmlns="http://xml.spss.com/pev/remote">
    <avReference baseUri="spsscr:///id=0a010a077e08b4a40000011f5cabdad38098"/>
    <table name="CQdefects"/>
  </getCompatibleRtDpd>
</soapenv:Body>
</soapenv:Envelope>

```

### ***SOAP response example***

The server responds to a `getCompatibleRtDpd` 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>
    <getCompatibleRtDpdResponse xmlns="http://xml.spss.com/pev/remote">
      <ns1:compatibleDpd baseUri="spsscr:///id=0a010a079039a7090000011f664c9e2a83b2"
        label="LATEST" name="myRTDPD" path="/myRTDPD" xmlns:ns1="http://xml.spss.com/pev"/>
    </getCompatibleRtDpdResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

## ***The getRtDpdInputDescription operation***

Returns information about the inputs for a specified Data Provider Definition - Real Time, including table keys, column names, and column types.

### ***Input fields***

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

Table 4-23  
Fields for `getRtDpdInputDescription`

Field	Type/Valid Values	Description
<code>rtDpdReference</code>	<code>labeledUriType</code>	

### Return information

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

Table 4-24  
Return Value

Type	Description
<code>rtDpdInputDescription</code>	This element describes all the possible inputs to the Scoring Service. It contains information about the Real Time Data Provider key(s) and Application View columns for the table which the Data Provider Definition is based, as well as the “relational” context data.

### Java example

To retrieve information about the inputs for a real-time data provider definition:

1. Create a `LabeledUriType` object.
2. Provide the `setBaseUri` method with a string corresponding to the location of the real-time data provider definition as a URI. Label or version markers should not be included in the URI. If a version other than *LATEST* is needed, use the `setLabel` method to define the label.
3. Supply the `getRtDpdInputDescription` operation with the URI object.

The operation returns a `RtDpdInputDescription` object containing information organized by tables in the real-time data provider definition. The `getRtDpdTableDescription` method returns an array of `RtDpdTableDescription` objects corresponding to the tables.

The table description objects contain information about the columns and the keys for the tables. Use the `getAvColumn` method for a table description object to obtain an array of `PevCatalogColumn` objects corresponding to the columns in the table. The `getName` and `getType` methods return the name and type for any entry in the column array.

Use the `getKey` method for a table description object to obtain an array of `Key` objects corresponding to the keys for the table. The `getName` method reports the name for a key in the array. The `getKeyColumn` method returns an array of `Column` objects containing the columns associated with a particular key. The `getName` and `getType` methods return the name and type for any entry in the column array.

The following sample retrieves the column and key information for a real-time data provider definition.

```
LabeledUriType type = new LabeledUriType();
type.setBaseUri("spsscr:///id=0a010a079039a7090000011f664c9e2a83b2")
RtDpdInputDescription inputDesc = stub.getRtDpdInputDescription(type);

RtDpdTableDescription[] tableDesc = inputDesc.getRtDpdTableDescription();
```

```

for (int i = 0; i < tableDesc.length; i++) {
    System.out.println("Table " + tableDesc[i].getName() +
        " contains the following columns in the application view:");
    PevCatalogColumn[] catColumn = tableDesc[i].getAvColumn();
    for (int j = 0; j < catColumn.length; j++) {
        System.out.println(catColumn[j].getName() +
            " (type = " + catColumn[j].getType().toString() + ")");
    }

    System.out.println("Table " + tableDesc[i].getName() +
        " contains the following keys:");
    Key[] keyArray = tableDesc[i].getKey();
    for (int k = 0; k < keyArray.length; k++) {
        System.out.println(keyArray[k].getName() +
            " having key column(s):");
        Column[] cols = keyArray[k].getKeyColumn();
        for (int m = 0; m < keyArray.length; m++) {
            System.out.println(cols[m].getName() +
                " (type = " + cols[m].getType().toString() + ")");
        }
    }
}

```

For web service clients based on JAX-WS, replace the arrays in the sample with List collections and update the array processing accordingly. For example:

```
List<RtDpdTableDescription> tableDesc = inputDesc.getRtDpdTableDescription();
```

### **SOAP request example**

Client invocation of the `getRtDpdInputDescription` 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
                xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
                <wsse:Username>Native/admin</wsse:Username>
                <wsse:Password
                    wsse:Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
                    #PasswordText">pass</wsse:Password>
                <wsse:Nonce>of0ShsZMIgHcdD0o6A8PkQ==</wsse:Nonce>
                <wsu:Created
                    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                    >2009-01-08T20:36:10Z</wsu:Created>
            </wsse:UsernameToken>
        </wsse:Security>
    </soapenv:Header>

```

```

</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>
  <getRtDpdInputDescription xmlns="http://xml.spss.com/pev/remote">
    <rtDpdReference baseUri="spsscr://?id=0a010a079039a7090000011f664c9e2a83b2"/>
  </getRtDpdInputDescription>
</soapenv:Body>
</soapenv:Envelope>

```

### **SOAP response example**

The server responds to a `getRtDpdInputDescription` 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>
    <getRtDpdInputDescriptionResponse xmlns="http://xml.spss.com/pev/remote">
      <ns1:rtDpdInputDescription xmlns:ns1="http://xml.spss.com/pev">
        <ns1:rtDpdTableDescription name="CQdefects">
          <ns1:key name="custID" isUnique="true">
            <ns1:keyColumn name="customerid" type="string"/>
          </ns1:key>
          <ns1:avColumn name="component" type="integer"/>
          <ns1:avColumn name="customerid" type="string"/>
        </ns1:rtDpdTableDescription>
      </ns1:rtDpdInputDescription>
    </getRtDpdInputDescriptionResponse>
  </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 4-25  
Return Value

Type	Description
string	



**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/pev/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/pev/remote">
      <version>4.20.000</version>
    </getVersionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

# **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. 52.](#)
2. Configure the web service endpoint. [For more information, see the topic Configuring the web service endpoint on p. 54.](#)
3. Programmatically configure the necessary endpoint behaviors. [For more information, see the topic Configuring endpoint behaviors on p. 55.](#)
4. Exercise the web service. [For more information, see the topic Exercising the service on p. 55.](#)

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) (<http://support.microsoft.com/kb/891386>) and [326790](http://support.microsoft.com/kb/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 Services web services by WCF clients:

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

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

To use these classes, ensure that the *IBM.SPSS.WCF.Utilities* project containing these classes has been compiled and added as a 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;
```

# 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](http://www.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.





- app.config files
  - WCF clients, 54
- Application View, 41, 44
  - direction, 12
  - merge conflicts, 16
  - merging versions, 24
- attributes
  - for variables, 33
- AvColumnConflict objects, 17
- AvColumnMeta objects, 17
- AvColumnMetaType objects, 17, 25
- AvTableConflict objects, 17
- AvTableResolution objects, 25
  
- bindings
  - in WSDL files, 5
- body elements
  - in SOAP messages, 3
  
- categories
  - for variables, 31
- CategoryQuery objects, 31
- CategoryType objects, 32
- column attributes
  - direction, 12
  - environment, 12
  - type, 11
- Column objects, 48
- ColumnDirectionType objects, 25
- ColumnEnvironmentType objects, 25
- Content Repository service
  - WCF clients, 53
- Content Repository URI service
  - WCF clients, 53
- CustomAttributesQuery objects, 34
- CustomAttributesType objects, 34
  
- Data Provider Definition - Real Time, 44, 47
- direction
  - column attributes, 12
  
- Enterprise View, 10
  - merge conflicts, 20
  - merging versions, 28
- environment
  - column attributes, 12
- EvColumnConflict objects, 21
- EvColumnMeta objects, 21
- EvColumnMetaType objects, 21, 28
- EvTableConflict objects, 21
- EvTableResolution objects, 28
  
- getAvColumn method
  - for RtDpdTableDescription objects, 48
- getAvColumnConflict method
  - for AvTableConflict objects, 17
- getAvColumnMeta method
  - for AvColumnConflict objects, 17
- getAvMergeConflicts operation, 16
- getBaseUri method
  - for ResourceTablesByLabel objects, 42
- getCategories operation, 31
- getColumnMeta method
  - for AvColumnMeta objects, 17
  - for EvColumnMeta objects, 21
- getCompatibleAv operation, 41
- getCompatibleRtDpd operation, 44
- getCustomAttributes operation, 33
- getDataType method
  - for VariableMetaType objects, 37
- getDirection method
  - for AvColumnMetaType objects, 17
- getEvColumnConflict method
  - for EvTableConflict objects, 21
- getEvColumnMeta method
  - for EvColumnConflict objects, 21
- getEvMergeConflicts operation, 20
- getId method
  - for CategoryType objects, 32
- getKey method
  - for RtDpdTableDescription objects, 48
- getKeyColumn method
  - for Key objects, 48
- getLabel method
  - for CategoryType objects, 32
  - for TablesByLabel objects, 42
- getMarker method
  - for EvColumnMeta objects, 21
- getName method
  - for AvColumnMetaType objects, 17
  - for AvTableConflict objects, 17
  - for Column objects, 48
  - for CustomAttributesType objects, 34
  - for EvColumnMetaType objects, 21
  - for EvTableConflict objects, 21
  - for Key objects, 48
  - for PevCatalogColumn objects, 48
  - for ResourceTablesByLabel objects, 42
  - for VariableMetaType objects, 37
- getNewVersionUri method
  - for MergeResult objects, 25, 29
- getPath method
  - for ResourceTablesByLabel objects, 42
- getQualifier method
  - for PevMetaData objects, 37
- getRtDpdInputDescription operation, 47
- getRtDpdTableDescription method
  - for RtDpdInputDescription objects, 46, 48
- getSupportedEnvironments method
  - for AvColumnMetaType objects, 17

- getTable method
  - for TablesByLabel objects, 42
- getTablesByLabel method
  - for ResourceTablesByLabel objects, 42
- getType method
  - for AvColumnMetaType objects, 17
  - for Column objects, 48
  - for EvColumnMetaType objects, 21
  - for PevCatalogColumn objects, 48
- getValues method
  - for CustomAttributesType objects, 34
- getVariableMetadata method
  - for QualifierMetaType objects, 37
- getVariableMetadata operation, 36
- getVersion operation, 23, 39, 50
- getVersionUri method
  - for AvColumnMeta objects, 17
  
- header elements
  - in SOAP messages, 3
- Holder classes
  - in JAX-WS, 6
- HTTP, 2
- HTTPS, 2
  
- Java proxies, 6
- JAX-WS, 6, 18, 22, 26, 29, 32, 35, 38, 43, 46, 49
  
- Key objects, 48
- keys, 47
  
- LabeledUriType objects, 45, 48
- legal notices, 57
- List collections
  - in JAX-WS, 6
  
- MergeResult objects, 25, 29
- merging views
  - Application View, 16, 24
  - Enterprise View, 20, 28
- MessageBodyMemberAttribute
  - for WCF clients, 55
- messages
  - in WSDL files, 5
- metadata
  - for variables, 36
- MetadataQuery objects, 37
  
- .NET framework, 52
- .NET proxies, 7
  
- performAvMerge operation, 24
- performEvMerge operation, 28
- PevCatalogColumn objects, 42, 48
- PevCatalogTable objects, 45
- PevEnvironmentType objects, 17
- PevMetadataType objects, 37
- PevServices service
  - service endpoint, 8
  - stubs, 8
  - WCF clients, 53
- port types
  - in WSDL files, 5
- predictor, 12
- Process Management service
  - WCF clients, 53
- protocols
  - in web services, 2
- proxies, 6
  - Java, 6
  - .NET, 7
  
- QualifierMetaType objects, 37
  
- ResourceTablesByLabel objects, 42
- RtDpdInputDescription objects, 46, 48
- RtDpdTableDescription objects, 46, 48
  
- Scoring service
  - WCF clients, 53
- service endpoints
  - PevServices service, 8
- services
  - in WSDL files, 6
- setBaseUri method
  - for LabeledUriType objects, 45, 48
- setColumnMeta method
  - for AvTableResolution objects, 25
  - for EvTableResolution objects, 29
- setDirection method
  - for AvColumnMetaType objects, 25
- setLabel method
  - for LabeledUriType objects, 45, 48
- setLocation method
  - for CategoryQuery objects, 31
  - for CustomAttributesQuery objects, 34
  - for MetadataQuery objects, 37
- setName method
  - for AvColumnMetaType objects, 25
  - for AvTableResolution objects, 25
  - for EvColumnMetaType objects, 29
  - for EvTableResolution objects, 28
  - for PevCatalogColumn objects, 42
  - for PevCatalogTable objects, 45
- setPevEnvironment method
  - for SupportedEnvironment objects, 42
- setSupportedEnvironments method
  - for AvColumnMetaType objects, 25
- setTableName method
  - for MetadataQuery objects, 37

- setType method
  - for AvColumnMetaType objects, 25
  - for EvColumnMetaType objects, 29
  - for PevCatalogColumn objects, 42
- setVariableName method
  - for CategoryQuery objects, 31
  - for CustomAttributesQuery objects, 34
- single sign-on
  - WCF clients, 52
- SOAP, 2–3
- stubs
  - PevServices service, 8
- SupportedEnvironment objects, 42
  
- TablesByLabel objects, 42
- target, 12
- trademarks, 58
- type
  - column attributes, 11
- types
  - in WSDL files, 4
  
- uniform resource identifiers, 14
- URIs
  - syntax, 14
  
- validateObject operation, 30
- VariableMetaType objects, 37
- variables
  - attributes, 33
  - categories, 31
  - metadata, 36
- versions
  - in URIs, 15
- Visual Studio, 52
  
- WCF clients, 52, 55
  - endpoint behaviors, 55
  - endpoint configuration, 54
  - limitations, 52
  - service reference, 52–53
  - single sign-on, 52
- web services
  - introduction to web services, 1
  - protocol stack, 2
  - system architecture, 1
  - what are web services?, 1
- web.config files
  - WCF clients, 54
- Windows Communication Foundation, 52
- WSDL files, 2–3
  - accessing, 8
  - bindings, 5
  - messages, 5
  - port types, 5
  
- services, 6
  - types, 4
- wsdl.exe, 7
- wsdl2java, 6
- wsimport, 6
  
- XmlElementAttribute
  - for WCF clients, 55