IBM® Analytical Decision Management
Version 18 Release 0

Response Service Developer's Guide

Note

Before you use this information and the product it supports, read the information in “Notices” on page 23.

Product Information

This edition applies to version 18, release 0, modification 0 of IBM Analytical Decision Management and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Introduction to web services

What are web services?

At a high level, a web service is a set of functionality distributed across a network (LAN or the Internet) using a common communication protocol. The web service serves as an intermediary between an application and its clients, providing both a standardized information structure and a standardized communication protocol for interaction between the two.

Where other methods of distributed application architecture rely on a single programming language being used on both the application and its clients, a web service allows the use of loosely coupled services between non-homogenous platforms and languages. This provides a non-architecture-specific approach allowing, for example, Java services to communicate with C# clients, or vice versa.

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

Web service system architecture

Web services are deployed and made publicly available using an application server, such as WebSphere®, JBoss Application Server, 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.

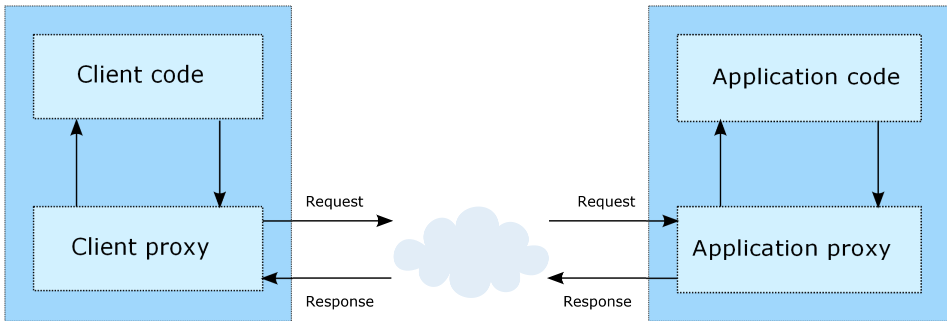


Figure 1. Web service architecture

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

Standardizing the format of the information passing between the client and the application allows a client written in one programming language to communicate with an application written in another. The proxy classes, which are automatically generated from a web service description by a variety of toolkits, handle the translation between native programming objects and the standardized representation. See the topic "Proxies" on page 5 for more information.

Web service protocol stack

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

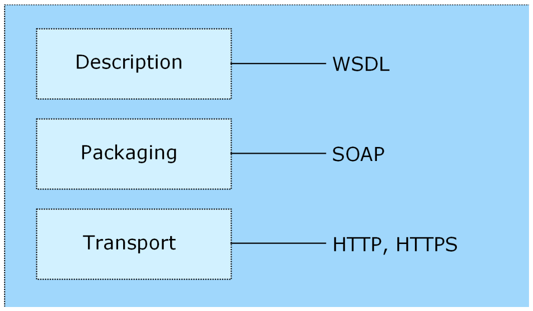


Figure 2. Web service protocol stack

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

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

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

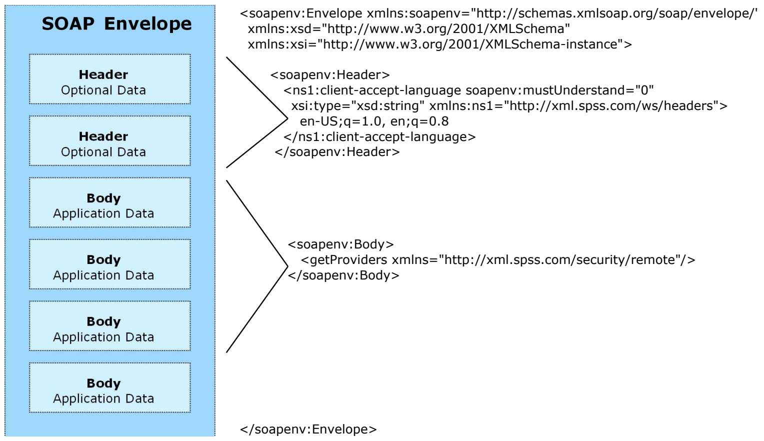
Simple Object Access Protocol

The Simple Object Access Protocol (SOAP) is a way to pass information between applications in an XML format.

SOAP messages are transmitted from the sending application to the receiving application, typically over an HTTP session. The actual SOAP message is made up of the Envelope element, which contains a Body element and an optional Header element.

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

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



```
<soapenv:Envelope xmlns:soapenv="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>
      <ns1:client-accept-language soapenv:mustUnderstand="0"
        xsi:type="xsd:string" xmlns:ns1="http://xml.spss.com/ws/headers">
          en-US;q=1.0, en;q=0.8
      </ns1:client-accept-language>
    </soapenv:Header>

      <soapenv:Body>
        <getProviders xmlns="http://xml.spss.com/security/remote"/>
      </soapenv:Body>

</soapenv:Envelope>
```

Figure 3. An example SOAP packet

Web Service Description Language

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

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

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

In accordance with the WSDL specification adopted by the World Wide Web Consortium, information in the WSDL is organized into the following sections:

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

Types

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

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

```
        <sequence>
          <element name="providerInfo[unbounded]" type="tns1:providerInfo" />
        </sequence>
      </complexType>
    </element>
  </schema>
</wsdl:types>
```

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

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

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

Messages

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

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

The *getProvidersResponse* message contains a single part, corresponding to the *getProvidersResponse* element defined in the types section of the WSDL file. Similarly, the *getProvidersRequest* message also contains a single part, as defined by the *getProviders* element in the types section. See the topic “Types” on page 3 for more information.

Port types

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

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

The *ProviderInformation* port type consists of a single operation, *getProviders*. Input to this operation corresponds to the *getProvidersRequest* message. The operation returns information in the structure defined by the *getProvidersResponse* message. See the topic “Messages” for more information.

Bindings

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

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

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

In this case, the transport attribute of the `wsdlsoap:binding` element defines HTTP as the transport protocol. The `getProviders` operation in the interface is bound to the SOAP messaging protocol.

Services

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

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

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

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

Proxies

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

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

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

An alternative Java proxy tool is `wsimport`, which is part of JAX-WS. The general structure of the generated classes is similar to that created by the Axis tool, but there are some differences. For example, instead of using arrays for input fields and returned items, the code generated from the `wsimport` tool uses `List` collections. In addition, if an input type matches an output type for a method, the `wsimport` tool uses a `Holder` class for the parameter.

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

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

Chapter 2. Response Service Overview

The Response Service supplements the Scoring Service. It's a web service allowing client applications such as call center interfaces to send responses to the service to be logged. For example, a bank might have a call center interface that presents specific offers to the call center agent. The agent can then make the appropriate offer to the bank customer, and the customer's answer (response) is sent to the Response Service and logged. The following figure presents the flow of a complete example.

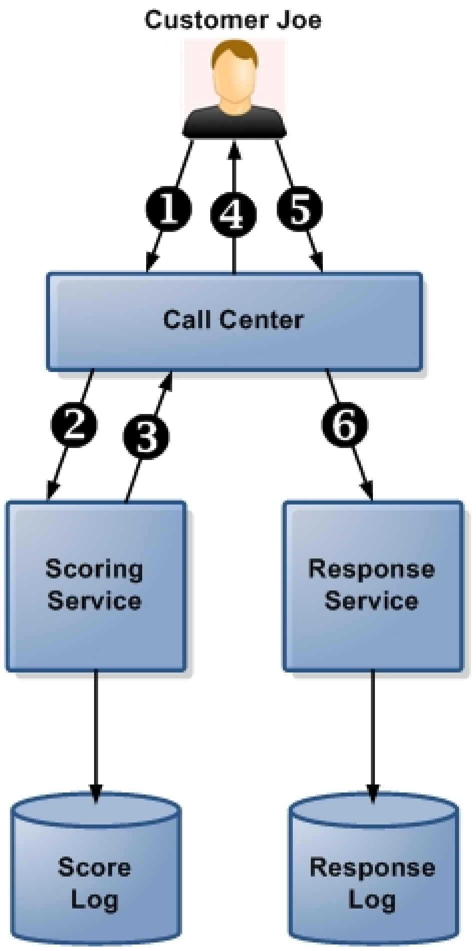


Figure 4. Example use of Scoring Service and Response Service

Table 1. Scoring Service and Response Service example.

Figure label	Description
❶	Customer Joe calls.
❷	The call center sends Joe's customer ID to the Scoring Service. If logging is turned on in the scoring configuration (optional), this information is sent to the score log. Note that score logging is distinct from response logging.
❸	The Scoring Service determines the best offer for customer Joe (*Gold Card*, for example) and sends the offer back to the call center. This information is also written to the score log (if enabled); Views and Queries can be written against the score log.

Table 1. Scoring Service and Response Service example (continued).

Figure label	Description
❹	The call center operator presents the *Gold Card* offer to customer Joe.
❺	Joe says yes to the offer.
❻	The call center sends Joe's "yes" response to the Response Service and this response is logged. Queries can be written against the response log, or against both logs.

The Response Service compliments the Scoring Service and has one primary method:

`public void logRequest(java.lang.String id, com.ibm.spss.dm.logging.schema.jaxb.info)` throws `java.rmi.RemoteException`

Where:

- `id` is the ID of the request to be logged. This should correspond with the ID returned from the scoring service (step 3 in the diagram above). This ID is specific to the scoring transaction; it would not be the same as the Customer ID in this example.
- `info` is the response information to be logged. See the topic "response-logging.xsd Schema" on page 11 for more information.

For a given score request, the scoring service may return 0 or more scores. Depending on the application, the scores may represent offers such as *Credit Card* or *Personal Loan*. Similarly, the `info` to be logged via the Response Service can contain 0 or more responses, which would indicate whether the customer accepted each offer.

A given response typically corresponds to a specific score returned from the Scoring Service. Each score returned can be identified by the `ModelOutput` value for that score, and correlated to the `ModelOutput` value of the response. Much like the `id`, these model outputs can be used to correlate a specific response to a given score output.

Each response can contain 0 or more `OutputResponse` items. For example, an `OutputResponse` may contain the name/value pairs `"Purchased"/"True"` and `"NegotiatedPrice"/"250.00"`.

For the `info` and for each response, the caller can specify additional properties. Again, these are name/value pairs to be used at the caller's discretion.

Accessing the Response Service

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

```
http://<host-name>:<port-number>/DM/services/ResponseService
```

The value of <*host-name*> corresponds to the machine on which the IBM® SPSS® Collaboration and Deployment Services Repository is installed, with <*port-number*> indicating the port number on which it is running.

Use the WSDL and schema files provided by IBM Corp. for proxy generation. For the Response Service, these files are:

- *response-logging.xsd*
- *dm-response-remote.xsd*
- *responseService.wsdl*

Calling Response Service Operations

The file *com.ibm.spss.dm.logging.client.ws.jaxws.ResponseLogger* contains a helper class that can be used to call the Response Service.

Alternatively, clients can 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 JAX-WS defined methods:

```
ResponseServices responseServices = new ResponseServices();
ResponseService responseService = responseServices.getResponseService();
String url = "http://pes_server:80/DM/services/ResponseService";
((BindingProvider) responseService).getRequestContext().put(
    BindingProvider.ENDPOINT_ADDRESS_PROPERTY, url);
UserNameToken userNameToken = new UserNameToken("admin", "spss");
userNameToken.updateServiceSecurity(responseService);
```

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

```
service.logRequest(id, info);
```

Security consideration

When calling the Response Service on the main IBM SPSS Collaboration and Deployment Services service, it is secured by IBM SPSS Collaboration and Deployment Services security. But when calling the Response Service that is deployed on a remote scoring server, the Response Service is secured by the application server security. See the IBM SPSS Collaboration and Deployment Services documentation for more information.

Response Log

The Response Service will log to the *SPSSDMRESPONSE_LOG* table in the IBM SPSS Collaboration and Deployment Services Repository. There will be one row logged for each call to `logRequest`.

This table consists of the following columns:

- **SERIAL**. This is `id` that is passed on the `logRequest`.
- **STAMP**. This is the time when the `logRequest` was called.
- **INFO**. This is the `info` that is passed on the `logRequest`.

As long as the Scoring Service `id` is passed as the `id` to the `logRequest`, the *SPSSDMRESPONSE_LOG* can be joined to the *SPSSSCORE_LOG* table using the *SERIAL* column.

The *INFO* column of the *SPSSDMRESPONSE_LOG* table follows the *response-logging.xsd* schema that is recorded in the database. Much like the *SPSSSCORE_LOG* used by the Scoring Service, custom SQL views containing XML queries can be written over this table. Note that while the XML schema of the Response Service and Scoring Service may complement each other, they are not the same schema.

It is entirely up to the caller of the Response Service to make sure appropriate information is logged. The Response Service makes no attempt to validate any information. For example, it does not validate that the `id` matches a scoring service `id`. Nor does it validate that a `ModelOutput` matches an output of the Scoring Service. The ability to join the *SPSSDMRESPONSE_LOG* and the *SPSSSCORE_LOG* is dependent on the information the caller provides.

Chapter 3. Schema Reference

response-logging.xsd Schema

This section provides a reference for all the elements in the Response Service schema (*response-logging.xsd*). Each topic lists the valid attributes for an element and its parent and child elements. The *response-logging.xsd* schema file is included in the *.zip* archive with this guide.

The Response Service WSDL file (*responseService.wsdl*) uses a separate schema named *dm-response-remote.xsd*.

Elements

Info Element

XML Representation

```
<xs:element name="Info">
  <xs:sequence>
    <xs:element name="Response" type="spss_dms_logging:ResponseType" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="ModelOutput" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
        </xs:element>
        <xs:element name="OutputResponse" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
        </xs:element>
        <xs:element name="Property" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="Property" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:element>
```

Child Elements

Property, Response

Response Element: Decision Management has the ability to generate multiple outputs (multiple offers). There will be one OutputRow for each output (for each offer).

XML Representation

```
<xs:element name="Response" type="spss_dms_logging:ResponseType" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="ModelOutput" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
    <xs:element name="OutputResponse" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
    <xs:element name="Property" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:element>
```

Parent Elements

Info

Child Elements

ModelOutput, OutputResponse, Property

ModelOutput Element: A name value pair.

Table 2. Attributes for ModelOutput

Attribute	Use	Description	Valid Values
name	**required**		*string*
value	optional	A value, in string representation. If this attribute is not specified, the value is considered to be null. The text representation of the numeric types is obvious, but several types are not. The format of the non-numeric types must be as follows: boolean="true"(case insensitive) or "1" or "false"(case insensitive) or "0", date="yyyy-MM-dd", daytime="HH:mm:ss", and timestamp="yyyy-MM-ddTHH:mm:ss".	*string*

XML Representation

```
<xs:element name="ModelOutput" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="optional"/>
</xs:element>
```

Parent Elements

Response

OutputResponse Element: A name value pair.

Table 3. Attributes for OutputResponse

Attribute	Use	Description	Valid Values
name	**required**		*string*
value	optional	A value, in string representation. If this attribute is not specified, the value is considered to be null. The text representation of the numeric types is obvious, but several types are not. The format of the non-numeric types must be as follows: boolean="true"(case insensitive) or "1" or "false"(case insensitive) or "0", date="yyyy-MM-dd", daytime="HH:mm:ss", and timestamp="yyyy-MM-ddTHH:mm:ss".	*string*

XML Representation

```
<xs:element name="OutputResponse" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="optional"/>
</xs:element>
```

Parent Elements

Response

Property Element: A name value pair.

Table 4. Attributes for Property

Attribute	Use	Description	Valid Values
name	**required**		*string*
value	optional	A value, in string representation. If this attribute is not specified, the value is considered to be null. The text representation of the numeric types is obvious, but several types are not. The format of the non-numeric types must be as follows: boolean="true"(case insensitive) or "1" or "false"(case insensitive) or "0", date="yyyy-MM-dd", daytime="HH:mm:ss", and timestamp="yyyy-MM-ddTHH:mm:ss".	*string*

XML Representation

```
<xs:element name="Property" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="optional"/>
</xs:element>
```

Parent Elements

Response

Property Element: A name value pair.

Table 5. Attributes for Property

Attribute	Use	Description	Valid Values
name	**required**		*string*

Table 5. Attributes for Property (continued)

Attribute	Use	Description	Valid Values
value	optional	A value, in string representation. If this attribute is not specified, the value is considered to be null. The text representation of the numeric types is obvious, but several types are not. The format of the non-numeric types must be as follows: boolean="true"(case insensitive) or "1" or "false"(case insensitive) or "0", date="yyyy-MM-dd", daytime="HH:mm:ss", and timestamp="yyyy-MM-ddTHH:mm:ss".	*string*

XML Representation

```
<xs:element name="Property" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="optional"/>
</xs:element>
```

Parent Elements

Info

Chapter 4. WSDL Reference

This section describes the Web Service Description Language (WSDL) file and its associated schema.

WSDL File

Following is the contents of the Response Service WSDL file (`responseService.wsdl`) included in the `.zip` archive with this guide. Complete Javadoc is also included for the WSDL.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
    targetNamespace="http://xml.pasw.com/dmResponseLogging/wsdl"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://xml.pasw.com/dmResponseLogging/wsdl"
    xmlns:jaxws="http://java.sun.com/xml/ns/jaxws"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:spss_rsr="http://xml.pasw.com/dm_response/remote"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/">

    <wsdl:documentation>
        Licensed Materials - Property of IBM
        IBM SPSS Products: Decision Management
        (C) Copyright IBM Corp. 2010, 2016
        US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA
        ADP Schedule Contract with IBM Corp.
    </wsdl:documentation>


    <!-- ******************************************************************* -->
    <!-- IMPORTANT INFORMATION                                               -->
    <!-- See: http://www.w3.org/TR/wsdl                                      -->
    <!--                                                                     -->
    <!-- The names of the elements in this WSDL are *very sensitive* and     -->
    <!-- must follow the WSI-Interop standard.                               -->
    <!-- ******************************************************************* -->

    <jaxws:bindings>
        <jaxws:package name="com.ibm.spss.dm.logging.ws.jaxws"/>
    </jaxws:bindings>

    <!-- ******************************************************************* -->
    <!-- TYPES                                                               -->
    <!-- Reference: http://www.w3.org/TR/wsdl#_types                         -->
    <!-- ******************************************************************* -->
    <wsdl:types>
        <xs:schema>
            <xs:import namespace="http://xml.pasw.com/dm_response/remote" schemaLocation="dm-response-remote.xsd"/>
        </xs:schema>
    </wsdl:types>


    <!-- ******************************************************************* -->
    <!-- MESSAGES                                                            -->
    <!-- Reference: http://www.w3.org/TR/wsdl#_messages                      -->
    <!-- ******************************************************************* -->
    <!-- Exceptions -->
    <!-- wsdl:message name="responseException">
        <wsdl:part name="responseException" element="spss_rsr:responseException"/>
    </wsdl:message-->

    <!-- ResponseService.ping -->
    <wsdl:message name="pingRequest">
        <wsdl:part name="parameters" element="spss_rsr:ping"/>
    </wsdl:message>
    <wsdl:message name="pingResponse">
        <wsdl:part name="parameters" element="spss_rsr:pingResponse"/>
    </wsdl:message>

    <!-- ResponseService.logResponseData -->
    <wsdl:message name="logRequest">
        <wsdl:part name="parameters" element="spss_rsr:logRequest"/>
    </wsdl:message>
    <wsdl:message name="logResponse">
```

```
        <wsdl:part name="parameters" element="spss_rsr:logResponse"/>
    </wsdl:message>

    <!-- ****************************************************************** -->
    <!-- PORT TYPES with associated operations                              -->
    <!-- Reference: http://www.w3.org/TR/wsdl#_porttypes                    -->
    <!-- ****************************************************************** -->
    <wsdl:portType name="ResponseService">

        <wsdl:operation name="ping">
            <wsdl:input name="ping" message="impl:pingRequest"/>
            <wsdl:output name="pingResponse" message="impl:pingResponse"/>
        </wsdl:operation>

        <wsdl:operation name="logRequest">
            <wsdl:input name="logRequest" message="impl:logRequest"/>
            <wsdl:output name="logResponse" message="impl:logResponse"/>
            <!-- wsdl:fault name="responseException" message="impl:responseException"/-->
        </wsdl:operation>

    </wsdl:portType>


    <!-- ****************************************************************** -->
    <!-- BINDINGS                                                           -->
    <!-- Reference: http://www.w3.org/TR/wsdl#_bindings                     -->
    <!-- ****************************************************************** -->
    <wsdl:binding name="ResponseDataSOAP" type="impl:ResponseService">
        <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="ping">
            <wsdlsoap:operation/>
            <wsdl:input name="ping">
                <wsdlsoap:body use="literal"/>
            </wsdl:input>
            <wsdl:output name="pingResponse">
                <wsdlsoap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>

        <wsdl:operation name="logRequest">
            <wsdlsoap:operation/>
            <wsdl:input name="logRequest">
                <wsdlsoap:body use="literal"/>
            </wsdl:input>
            <wsdl:output name="logResponse">
                <wsdlsoap:body use="literal"/>
            </wsdl:output>
            <!-- wsdl:fault name="responseException">
                <wsdlsoap:fault name="responseException" use="literal"/>
            </wsdl:fault-->
        </wsdl:operation>

    </wsdl:binding>


    <!-- ****************************************************************** -->
    <!-- SERVICES                                                           -->
    <!-- Reference: http://www.w3.org/TR/wsdl#_services                     -->
    <!-- ****************************************************************** -->
    <wsdl:service name="ResponseServices">
        <wsdl:port name="ResponseService" binding="impl:ResponseDataSOAP">
            <wsdlsoap:address location="http://localhost:8080/DM/services/ResponseService"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

WSDL Schema

This section provides a reference for all the elements in the *dm-response-remote.xsd* schema associated with the Response Service WSDL file. Each topic lists the valid attributes for an element and its parent and child elements. The *dm-response-remote.xsd* file is included in the *.zip* archive with this guide.

Elements

Info Element

XML Representation

```
<xs:element name="Info">
  <xs:sequence>
    <xs:element name="Response" type="spss_dms_logging:ResponseType" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="ModelOutput" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
        </xs:element>
        <xs:element name="OutputResponse" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
        </xs:element>
        <xs:element name="Property" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="Property" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:element>
```

Parent Elements

logRequest

Child Elements

Property, Response

Response Element: Decision Management has the ability to generate multiple outputs (multiple offers). There will be one OutputRow for each output (for each offer).

XML Representation

```
<xs:element name="Response" type="spss_dms_logging:ResponseType" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="ModelOutput" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
    <xs:element name="OutputResponse" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
    <xs:element name="Property" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:element>
```

Parent Elements

Info

Child Elements

ModelOutput, OutputResponse, Property

ModelOutput Element: A name value pair.

Table 6. Attributes for ModelOutput

Attribute	Use	Description	Valid Values
name	**required**		*string*

Table 6. Attributes for ModelOutput (continued)

Attribute	Use	Description	Valid Values
value	optional	A value, in string representation. If this attribute is not specified, the value is considered to be null. The text representation of the numeric types is obvious, but several types are not. The format of the non-numeric types must be as follows: boolean="true"(case insensitive) or "1" or "false"(case insensitive) or "0", date="yyyy-MM-dd", daytime="HH:mm:ss", and timestamp="yyyy-MM-ddTHH:mm:ss".	*string*

XML Representation

```
<xs:element name="ModelOutput" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="optional"/>
</xs:element>
```

Parent Elements

Response

OutputResponse Element: A name value pair.

Table 7. Attributes for OutputResponse

Attribute	Use	Description	Valid Values
name	**required**		*string*
value	optional	A value, in string representation. If this attribute is not specified, the value is considered to be null. The text representation of the numeric types is obvious, but several types are not. The format of the non-numeric types must be as follows: boolean="true"(case insensitive) or "1" or "false"(case insensitive) or "0", date="yyyy-MM-dd", daytime="HH:mm:ss", and timestamp="yyyy-MM-ddTHH:mm:ss".	*string*

XML Representation

```
<xs:element name="OutputResponse" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="optional"/>
</xs:element>
```

Parent Elements

Response

Property Element: A name value pair.

Table 8. Attributes for Property

Attribute	Use	Description	Valid Values
name	**required**		*string*
value	optional	A value, in string representation. If this attribute is not specified, the value is considered to be null. The text representation of the numeric types is obvious, but several types are not. The format of the non-numeric types must be as follows: boolean="true"(case insensitive) or "1" or "false"(case insensitive) or "0", date="yyyy-MM-dd", daytime="HH:mm:ss", and timestamp="yyyy-MM-ddTHH:mm:ss".	*string*

XML Representation

```
<xs:element name="Property" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="optional"/>
</xs:element>
```

Parent Elements

Response

Property Element: A name value pair.

Table 9. Attributes for Property

Attribute	Use	Description	Valid Values
name	**required**		*string*

Table 9. Attributes for Property (continued)

Attribute	Use	Description	Valid Values
value	optional	A value, in string representation. If this attribute is not specified, the value is considered to be null. The text representation of the numeric types is obvious, but several types are not. The format of the non-numeric types must be as follows: boolean="true"(case insensitive) or "1" or "false"(case insensitive) or "0", date="yyyy-MM-dd", daytime="HH:mm:ss", and timestamp="yyyy-MM-ddTHH:mm:ss".	*string*

XML Representation

```
<xs:element name="Property" type="spss_dms_logging:nameValueType" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="optional"/>
</xs:element>
```

Parent Elements

Info

logRequest Element

Request for log response data service call

XML Representation

```
<xs:element name="logRequest">
  <xs:sequence>
    <xs:element name="id" type="xs:string"/>
    <xs:element ref="spss_rs:Info"/>
  </xs:sequence>
</xs:element>
```

Child Elements

Info, id

id Element:

XML Representation

```
<xs:element name="id" type="xs:string"/>
```

Parent Elements

logRequest

logResponse Element

Response from logResponseData service call

XML Representation

```
<xs:element name="logResponse"/>
```

ping Element

Request for ping service call

XML Representation

```
<xs:element name="ping"/>
```

pingResponse Element

Response from ping service call

XML Representation

```
<xs:element name="pingResponse"/>
```

responseException Element

Response service exception

XML Representation

```
<xs:element name="responseException"/>
```

Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

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

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

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

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

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

Terms and conditions for product documentation

Index