

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



Note: Before using this information and the product it supports, read the general information under Notices on p. 102.

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

1	<i>Introduction to web services</i>	1
	What are web services?	1
	Web service system architecture	1
	Web service protocol stack	2
	Simple Object Access Protocol	3
	Web Service Description Language	3
	Proxies	6
2	<i>Scoring Service overview</i>	8
	Accessing the Scoring Service	8
	Calling Scoring Service operations	8
3	<i>Scoring concepts</i>	10
	Score providers	10
	Scoring configurations	10
	Scores	11
	Scoring metrics	12
4	<i>Operation reference</i>	15
	The buildConfigurationDetails operation	15
	The changeConfigurationRunningState operation	19
	The getConfigurationDetails operation	21
	The getConfigurations operation	26
	The getMetadata operation	28
	The getMetricItems operation	31
	The getMetricValue operation	34
	The getScore operation	37
	The getServiceDetails operation	42
	The getVersion operation	44
	The ping operation	45
	The removeConfiguration operation	46

The setConfigurationDetails operation	47
The updateConfigurationDetails operation	51

5 Scoring Service logging 57

Database objects	57
Request log table	57
Database views	58
XML schema	61
SQL for creating database objects	67
MS SQL Server (scoring_logging_sql_server.sql)	67
Oracle (scoring_logging_ora.sql)	76
DB2 (scoring_logging_db2.sql)	85
Customizing logging	88
Custom MDB	89
Custom JMS settings	90
Logging all requests to a different table	90
Logging specific scoring configurations to a different table	90

6 Scoring Service access using the Java Message Service 91

Using JMS to access the Scoring Service	91
Java example for JMS	93
Environment configuration	95
Configuring a Websphere environment	95
Configuring a JBoss environment	96
Configuring a WebLogic environment	96

Appendices

A Microsoft® .NET Framework-based clients 97

Adding a service reference.	97
Service reference modifications	98
Configuring the web service endpoint.	99

Configuring endpoint behaviors	100
Exercising the service.....	100
<i>B Notices</i>	102
<i>Index</i>	105

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.

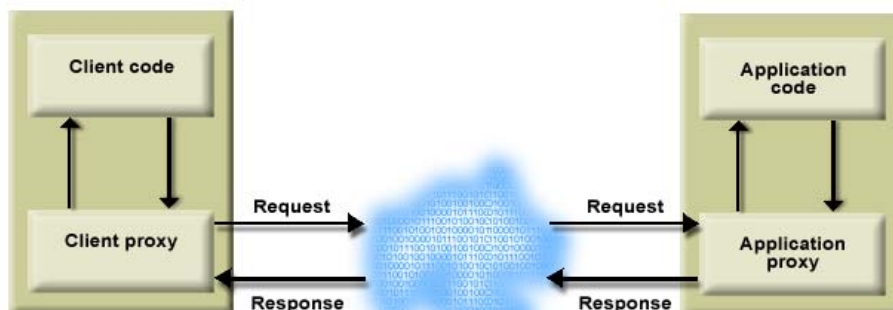
Advantages to implementing application functionality as web services include the following:

- 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



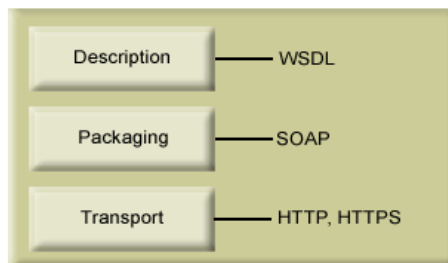
The client code supplies input to an operation offered by a proxy class. The proxy class generates a request containing a standardized representation of the input and sends it across the network to the application. A proxy class on the server receives the request and unmarshals the contents into objects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

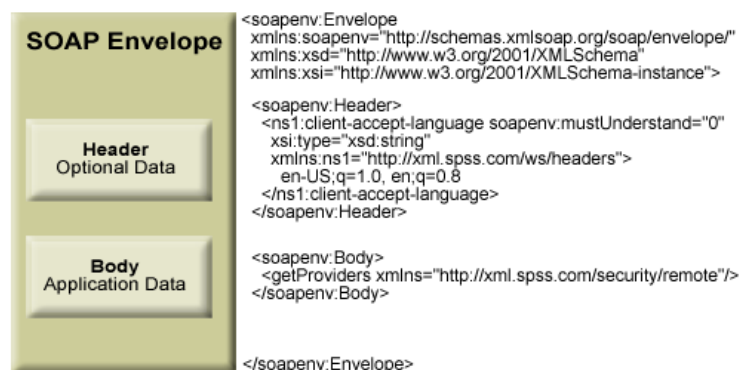
Simple Object Access Protocol

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

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

Port types

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

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

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

Bindings

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

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

Scoring Service overview

The Scoring Service allows client applications to employ real-time scores derived from predictive models developed in IBM® SPSS® Modeler, IBM® SPSS® Statistics, or third party tools. The service fetches a specified model, loads it, invokes the correct scoring implementation, and returns the result to the client. Supported models include regression (linear and logistic), decision trees, decision lists, neural networks, and naïve Bayes defined in SPSS Modeler streams or in PMML from SPSS Statistics.

Scoring can be either synchronous or asynchronous, depending on whether the client needs to wait for a score before proceeding or not. The service can load multiple models simultaneously for scoring and can be virtualized across multiple servers in a cluster configuration to handle large processing loads. The service logs all scoring activity for regulatory audit purposes. Configuring models for scoring and monitoring the service performance can be done using the IBM® SPSS® Collaboration and Deployment Services Deployment Manager.

The Scoring Service is often used in conjunction with the Data Services Service to access the data for scoring configurations. For more information, see Data Services Service documentation.

Accessing the Scoring 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>/scoring/services/Scoring.HttpV2
```

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/scoring/services/Scoring.HttpV2?wsdl
```

Note that previous versions of this service used a different endpoint. Although that endpoint is still functional, it is deprecated and does not include new features introduced in this release. If possible, you should update any references in existing clients to use the new endpoint. New clients should use the endpoint above to take advantage of the latest functionality.

Calling Scoring 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 = "/scoring/services/Scoring.HttpV2";  
URL url = new URL("http", "cads_server", 80, context);
```

```
ScoringService service = new ScoringServiceLocator();  
stub = service.getStatus(url);
```

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

```
stub.getServiceDetails();
```

Scoring concepts

Score providers

Score providers provide the Scoring Service with the internal processing instructions necessary for specific model types. For the service to be able to create a scoring configuration for a model, a score provider for that type of model must be available. The provider registers itself with the service, indicating which MIME types it supports. Each provider has a unique name and identifier to distinguish it from other providers, and each is versioned separately. Providers available to the scoring service include:

- *SmartScore Score Provider*, for processing PMML files
- *Modeler Score Provider*, for processing IBM® SPSS® Modeler streams and scenario files

The Scoring Service includes the `getServiceDetails` operation for determining which providers are available in the system.

Scoring configurations

Before a model can be used for scoring, supplemental information must be defined. For example, a IBM® SPSS® Modeler stream containing a IBM® SPSS® Collaboration and Deployment Services Enterprise View source node needs an associated real-time data provider definition specifying how to retrieve the required data. Such information constitutes a **scoring configuration** for the model, and defines scoring parameters such as the following:

- Identification information for the configuration itself
- Identification information for the model used for scoring
- The data provider for the input
- Settings for logging
- The order of the input attributes
- Cache size used for scoring models

A single model may be used in a variety of scoring situations that require different scoring parameters. For example, scores may be based on a test data provider for internal purposes and on a different data provider for production usage. Alternatively, the information being logged as result of scoring may depend on the scoring situation. To allow a model to be used in differing scoring circumstances, any model may be associated with multiple scoring configurations.

Scoring configurations can be suspended to temporarily prevent the processing of score requests. A suspended configuration must be reactivated before it can be used to generate scores.

The Scoring Service includes operations for defining, retrieving, updating, suspending, and deleting configurations, as well as for loading a scoring model associated with a configuration.

Scores

Applying a predictive model to a set of data can produce a variety of scores, such as predicted values, predicted probabilities, and other values based on that model. The type of score produced is referred to as the **scoring function**. The following scoring functions are available:

PREDICT	<i>Returns the predicted value of the target variable.</i>
STDDEV	<i>Standard deviation.</i>
PROBABILITY	<i>Probability associated with a particular category of a target variable. Applies only to categorical variables. In the absence of the optional third parameter, <i>category</i>, this is the probability that the predicted category is the correct one for the target variable. If a particular category is specified, then this is the probability that the specified category is the correct one for the target variable.</i>
CONFIDENCE	<i>A probability measure associated with the predicted value of a categorical target variable. Applies only to categorical variables.</i>
NODEID	<i>The terminal node number. Applies only to tree models.</i>
CUMHAZARD	<i>Cumulative hazard value. Applies only to Cox regression models.</i>
NEIGHBOR	<i>The ID of the <i>k</i>th nearest neighbor. Applies only to nearest neighbor models. In the absence of the optional third parameter, <i>k</i>, this is the ID of the nearest neighbor. The ID is the value of the case labels variable, if supplied, and otherwise the case number.</i>
DISTANCE	<i>The distance to the <i>k</i>th nearest neighbor. Applies only to nearest neighbor models. In the absence of the optional third parameter, <i>k</i>, this is the distance to the nearest neighbor. Depending on the model, either Euclidean or City Block distance will be used.</i>

The following table lists the set of scoring functions available for each type of model that supports scoring. The function type denoted as PROBABILITY (*category*) refers to specification of a particular category for the PROBABILITY function.

Table 3-1
Supported functions by model type

Model type	Supported functions
Tree (categorical target)	PREDICT, PROBABILITY, PROBABILITY (<i>category</i>), CONFIDENCE, NODEID
Tree (scale target)	PREDICT, NODEID, STDDEV
Boosted Tree (C5.0)	PREDICT, CONFIDENCE
Linear Regression	PREDICT, STDDEV
Automatic Linear Models	PREDICT
Binary Logistic Regression	PREDICT, PROBABILITY, PROBABILITY (<i>category</i>), CONFIDENCE
Conditional Logistic Regression	PREDICT
Multinomial Logistic Regression	PREDICT, PROBABILITY, PROBABILITY (<i>category</i>), CONFIDENCE
General Linear Model	PREDICT, STDDEV
Discriminant	PREDICT, PROBABILITY, PROBABILITY (<i>category</i>), CONFIDENCE
TwoStep Cluster	PREDICT
K-Means Cluster	PREDICT, CONFIDENCE
Kohonen	PREDICT

Model type	Supported functions
Neural Net (categorical target)	PREDICT, PROBABILITY, PROBABILITY (category), CONFIDENCE
Neural Net (scale target)	PREDICT
Naive Bayes	PREDICT, PROBABILITY, PROBABILITY (category), CONFIDENCE
Anomaly Detection	PREDICT
Ruleset	PREDICT, CONFIDENCE
Generalized Linear Model (categorical target)	PREDICT, PROBABILITY, PROBABILITY (category), CONFIDENCE
Generalized Linear Model (scale target)	PREDICT, STDDEV
Generalized Linear Mixed Model (categorical target)	PREDICT, PROBABILITY, PROBABILITY (category), CONFIDENCE
Generalized Linear Mixed Model (scale target)	PREDICT
Ordinal Multinomial Regression	PREDICT, PROBABILITY, PROBABILITY (category), CONFIDENCE
Cox Regression	PREDICT, CUMHAZARD
Nearest Neighbor (scale target)	PREDICT, NEIGHBOR, NEIGHBOR(K), DISTANCE, DISTANCE(K)
Nearest Neighbor (categorical target)	PREDICT, PROBABILITY, PROBABILITY (category), CONFIDENCE, NEIGHBOR, NEIGHBOR(K), DISTANCE, DISTANCE(K)
Bayesian Network	PREDICT, PROBABILITY, PROBABILITY (category), CONFIDENCE
Support Vector Machine (categorical target)	PREDICT, PROBABILITY, PROBABILITY (category), CONFIDENCE
Support Vector Machine (scale target)	PREDICT, STDDEV

- For the Binary Logistic Regression, Multinomial Logistic Regression, and Naive Bayes models, the value returned by the CONFIDENCE function is identical to that returned by the PROBABILITY function.
- For the K-Means model, the value returned by the CONFIDENCE function is the least distance.
- For tree and ruleset models, the confidence can be interpreted as an adjusted probability of the predicted category and is always less than the value given by PROBABILITY. For these models, the confidence value is more reliable than the value given by PROBABILITY.
- For neural network models, the confidence provides a measure of whether the predicted category is much more likely than the second-best predicted category.
- For Ordinal Multinomial Regression and Generalized Linear Model, the PROBABILITY function is supported when the target variable is binary.
- For nearest neighbor models without a target variable, the available functions are NEIGHBOR and DISTANCE.

Scoring metrics

Scoring metrics are measurements that reflect the performance of a scoring configuration or the service itself. Available metric items include the following:

- **Service Scores.** Total number of scores produced by the service.

- **Service Uptime.** Amount of time, measured in seconds, that the service has been running.
- **Average Latency.** Average amount of time, measured in milliseconds, between the request for a score and the generation of the score.
- **Minimum Latency.** Smallest amount of time, measured in milliseconds, between the request for a score and the generation of the score.
- **Maximum Latency.** Largest amount of time, measured in milliseconds, between the request for a score and the generation of the score.
- **Score Data Initialization Time.** Amount of time, measured in milliseconds, that the data service takes to be initialized for a score request.
- **Average Data Initialization Time.** Average amount of time, measured in milliseconds, that the data service takes to be initialized for a score request.
- **Minimum Data Initialization Time.** Smallest amount of time, measured in milliseconds, that the data service takes to be initialized for a score request.
- **Maximum Data Initialization Time.** Largest amount of time, measured in milliseconds, that the data service takes to be initialized for a score request.
- **Score Data Access Time.** Amount of time, measured in milliseconds, that the data service takes to access the data.
- **Average Data Access Time.** Average amount of time, measured in milliseconds, that the data service takes to access the data.
- **Minimum Data Access Time.** Smallest amount of time, measured in milliseconds, that the data service takes to access the data.
- **Maximum Data Access Time.** Largest amount of time, measured in milliseconds, that the data service takes to access the data.
- **Score Computation Wait Time.** Amount of time, measured in milliseconds, that the score provider worker spends waiting for the data service.
- **Average Computation Wait Time.** Average amount of time, measured in milliseconds, that the score provider worker spends waiting for the data service.
- **Minimum Computation Wait Time.** Smallest amount of time, measured in milliseconds, that the score provider worker spends waiting for the data service.
- **Maximum Computation Wait Time.** Largest amount of time, measured in milliseconds, that the score provider worker spends waiting for the data service.
- **Score Computation Time.** Amount of time, measured in milliseconds, that the score provider worker spends computing the score.
- **Average Computation Time.** Average amount of time, measured in milliseconds, that the score provider worker spends computing the score.
- **Minimum Computation Time.** Smallest amount of time, measured in milliseconds, that the score provider worker spends computing the score.
- **Maximum Computation Time.** Largest amount of time, measured in milliseconds, that the score provider worker spends computing the score.
- **Average Log Serialization Time.** Average amount of time, measured in milliseconds, to create a log entry in XML format.

- **Minimum Log Serialization Time.** Smallest amount of time, measured in milliseconds, to create a log entry in XML format.
- **Maximum Log Serialization Time.** Largest amount of time, measured in milliseconds, to create a log entry in XML format.
- **Average Log Queue Time.** Average amount of time, measured in milliseconds, to place the XML logging data onto the JMS queue.
- **Minimum Log Queue Time.** Smallest amount of time, measured in milliseconds, to place the XML logging data onto the JMS queue.
- **Maximum Log Queue Time.** Largest amount of time, measured in milliseconds, to place the XML logging data onto the JMS queue.
- **Configuration Scores.** Total number of scores produced by a specific scoring configuration.
- **Score Elapsed Time.** Amount of time, measured in milliseconds, since the previous score generation.
- **Configuration Uptime.** Amount of time, measured in seconds, that the scoring configuration has been available for scoring.
- **Cache Hits.** Number of successful attempts to retrieve data from the memory cache for a scoring configuration.
- **Cache Misses.** Number of failed attempts to retrieve data from the memory cache for a scoring configuration. Each failed attempt results in a new service call to retrieve the necessary data.

The scale for a particular metric item determines the number of decimal points included in the measurement.

The Scoring Service includes operations for retrieving the list of metric items and the value for a specific metric item for a configured model.

Operation reference

The *buildConfigurationDetails* operation

Creates a new, default scoring configuration for a specified model. The score provider associated with the model type generates the appropriate configuration parameters for the model. This operation returns the configuration for any desired modifications. A subsequent call to the *setConfigurationDetails* operation stores the configuration in the system.

Input fields

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

Table 4-1
Fields for buildConfigurationDetails

Field	Type/Valid Values	Description
modelReference	modelReference	This element provides all the fields necessary to locate a specific version of a model file.

Return information

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

Table 4-2
Return Value

Type	Description
configurationDetails	This element contains the information that's exchanged to view or adjust the configuration settings.

Java example

To create a default scoring configuration:

1. Create a `ModelReference` object.
2. Provide the `setId` method with a string corresponding to the identifier for the model.
3. Supply the `setResourcePath` method with a string denoting the repository path to the model.
4. Provide the `setLabel` method with a string identifying the label associated with the version of the model to use for scoring.
5. Supply the *buildConfigurationDetails* operation with the model reference object.

The following sample builds a default scoring configuration for the latest version of the model defined by the *naivebayes.xml* PMML file.

```

ModelReference model = new ModelReference();
modelRef.setId("0a010a07d3dc251c000011eac84bfd18771");
modelRef.setResourcePath("/PMML/naivebayes.xml");
modelRef.setLabel("LATEST");
ConfigurationDetails details = stub.buildConfigurationDetails(model);

ModelReference modelRef = details.getModelReference();
System.out.println("Associated model: " + modelRef.getResourcePath());
System.out.println("Version label of model: " + modelRef.getLabel());
ModelInputMetadata inputMetadata = details.getModelInputMetadata();
ModelInputMetadataField[] inputMetadataField = inputMetadata.getModelInputMetadataField();
System.out.println("Inputs:\n");
System.out.println("NAME\tTYPE\tDESCRIPTION\tRETURNED\n");
for (int i = 0; i < inputMetadataField.length; i++) {
    System.out.println(inputMetadataField[i].getName() + "\t" +
        inputMetadataField[i].getType() + "\t" +
        inputMetadataField[i].getDescription() + "\t" +
        inputMetadataField[i].getIsReturned());
}
LogSettings ISettings = details.getLogSettings();
System.out.println("Log Destination: " + ISettings.getLogDestination());
LoggableItemGroup[] Igroups = ISettings.getLoggableItemGroup();
for (int i = 0; i < Igroups.length; i++) {
    System.out.println(Igroups[i].getName() + " (" + Igroups[i].getId() + ")");
    LoggableItem[] logItem = Igroups[i].getLoggableItem();
    if (logItem.length > 0) {
        System.out.println("NAME\tID\tENABLED\n");
        for (int j = 0; j < logItem.length; j++) {
            System.out.println(logItem[j].getName() + "\t" +
                logItem[j].getId() + "\t" +
                logItem[j].getIsEnabled());
        }
    }
}
InputAttributeOrder inputOrder = details.getInputAttributeOrder();
String[] inputAttributes = inputOrder.getAttributes();
System.out.println("The order of the input attributes is:\n");
for (int i = 0; i < inputAttributes.length; i++) {
    System.out.println(inputAttributes[i]);
}
ConfigurationItem[] configItem = details.getConfigurationItem();
for (int i = 0; i < configItem.length; i++) {
    System.out.println("Configuration item for this configuration is:\n" +
        configItem[i].getXML());
}

```

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<ModelInputMetadataField> inputMetadataFieldList = inputMetadata.getModelInputMetadataField();
```

For information on working with ConfigurationDetails objects, see the java sample for the [getConfigurationDetails](#) operation.

SOAP request example

Client invocation of the buildConfigurationDetails 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>
    <buildConfigurationDetails xmlns="http://xml.spss.com/scoring-v2/remote">
      <modelReference label="LATEST" resourcePath="/PMML/naivebayes.xml"
        id="0a010a07d3dc251c0000011eac84bfd18771" xmlns="http://xml.spss.com/scoring-v2"/>
    </buildConfigurationDetails>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a buildConfigurationDetails 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>
    <d:buildConfigurationDetailsResponse xmlns:a="http://xml.spss.com/data"
```

```

xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
xmlns:d="http://xml.spss.com/scoring-v2/remote"
xmlns:e="http://xml.spss.com/scoring/exception">
<c:configurationDetails cfgSerial="21c881e0-ddc4-11dd-aca9-dcb0b853b9a7" id="">
  <c:modelReference label="LATEST" resourcePath="/PMML/naivebayes.xml"
    id="0a010a07d3dc251c000011eac84bfd18771"/>
  <c:modelInputMetadata isReturnEnabled="false">
    <c:modelInputMetadataField name="age" type="double"
      description="Age of Respondent" isReturned="false"/>
    <c:modelInputMetadataField name="educ" type="double"
      description="Highest Year of School Completed" isReturned="false"/>
    <c:modelInputMetadataField name="prestg80" type="double"
      description="R's Occupational Prestige Score (1980)" isReturned="false"/>
    <c:modelInputMetadataField name="speduc" type="double"
      description="Highest Year School Completed, Spouse" isReturned="false"/>
  </c:modelInputMetadata>
  <c:logSettings isEnabled="false" logDestination="queue/PASWLog">
    <c:loggableItemGroup name="Context Data" id="INPUT_CONTEXT"/>
    <c:loggableItemGroup name="Model Inputs" id="INPUT_MODEL">
      <c:loggableItem isEnabled="false" name="age" id="age"/>
      <c:loggableItem isEnabled="false" name="educ" id="educ"/>
      <c:loggableItem isEnabled="false" name="prestg80" id="prestg80"/>
      <c:loggableItem isEnabled="false" name="speduc" id="speduc"/>
    </c:loggableItemGroup>
    <c:loggableItemGroup name="Model Outputs" id="OUTPUT">
      <c:loggableItem isEnabled="false" name="Prediction" id="Prediction"/>
      <c:loggableItem isEnabled="false" name="Prob" id="Prob"/>
      <c:loggableItem isEnabled="false" name="Prob-1.0" id="Prob-1.0"/>
      <c:loggableItem isEnabled="false" name="Prob-2.0" id="Prob-2.0"/>
      <c:loggableItem isEnabled="false" name="Confidence" id="Confidence"/>
    </c:loggableItemGroup>
    <c:loggableItemGroup name="Scoring Engine Properties" id="ENGINE_PROPERTY">
      <c:loggableItem isEnabled="false" name="Model Path" id="MODEL_PATH"/>
      <c:loggableItem isEnabled="false" name="Scoring Configuration Name"
        id="CONFIGURATION_NAME"/>
      <c:loggableItem isEnabled="false" name="Model Version Label" id="MODEL_LABEL"/>
      <c:loggableItem isEnabled="false" name="Model MIME type" id="MODEL_MIME_TYPE"/>
      <c:loggableItem isEnabled="false" name="Model Version Marker"
        id="MODEL_MARKER"/>
      <c:loggableItem isEnabled="false" name="Scoring Configuration Serial"
        id="CONFIGURATION_SERIAL"/>
      <c:loggableItem isEnabled="false" name="Model ID" id="MODEL_ID"/>
    </c:loggableItemGroup>
    <c:loggableItemGroup name="Score Provider Properties" id="PROVIDER_PROPERTY"/>
  </c:logSettings>
  <c:inputAttributeOrder>
    <c:attributes>age</c:attributes>
    <c:attributes>educ</c:attributes>
    <c:attributes>prestg80</c:attributes>
    <c:attributes>speduc</c:attributes>
  </c:inputAttributeOrder>
  <c:cacheSize value="1" min="1" max="100"/>

```



```

<c:batchEnabledFlag>>false</c:batchEnabledFlag>
<c:configurationItem>
  <c:xml>&lt;?xml version=&quot;1.0&quot;
    encoding=&quot;UTF-8&quot;?&gt; &lt;node
    name=&quot;Model Specific&quot; objectType=&quot;Model
    Specific&quot;&gt;&lt;boolean
    objectType=&quot;PredictionOnly&quot; name=&quot;A flag for
    prediction only&quot; required=&quot;true&quot;
    value=&quot;false&quot;/&gt;&lt;choice
    objectType=&quot;MissingValuePolicy&quot; name=&quot;Missing
    value handling policy for this model&quot;
    required=&quot;true&quot;&gt;&lt;item
    value=&quot;Substitute: Different model has different substitute
    policy&quot;/&gt;&lt;item value=&quot;Score as SYSMIS: Score
    result would be SYSMIS&quot;/&gt;&lt;item value=&quot;Throw
    exception: A SCMissingPredictorException is
    thrown&quot;/&gt;&lt;/choice&gt;&lt;/node&gt;</c:xml>
  </c:configurationItem>
</c:configurationDetails>
</d:buildConfigurationDetailsResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The *changeConfigurationRunningState* operation

Changes the running state of a scoring configuration to a specified value. Use this operation to temporarily suspend a configuration from processing requests or to reactivate a previously suspended configuration. Valid values for the running state include:

- **ACTIVE**. The configuration is consuming resources and can process score requests (if the configuration status code does not indicate an error).
- **SUSPENDED**. The configuration is not consuming resources and cannot process score requests.

The running state of a configuration is preserved when restarting the IBM® SPSS® Collaboration and Deployment Services Repository server. Configurations that are suspended cannot be activated by a server restart. However, a suspended configuration can be modified as needed.

Input fields

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

Table 4-3
Fields for *changeConfigurationRunningState*

Field	Type/Valid Values	Description
configurationRunningState	configurationRunningState	This element indicates the state that the configuration should be set to

Java example

To modify the running state of a scoring configuration:

1. Create a `ConfigurationRunningState` object.
2. Provide the `setId` method with a string corresponding to the identifier for the configuration.
3. Supply the `setState` method with a `RunningState` constant corresponding to the desired state.
4. Supply the `changeConfigurationRunningState` operation with the `ConfigurationRunningState` object.

The following sample changes the running state for the *nb-config* configuration to *suspended*.

```
ConfigurationRunningState cfgState = new ConfigurationRunningState();
cfgState.setId("nb-config");
cfgState.setState(RunningState.SUSPENDED);
stub.changeConfigurationRunningState(cfgState);
```

SOAP request example

Client invocation of the `changeConfigurationRunningState` 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>
    <changeConfigurationRunningState xmlns="http://xml.spss.com/scoring-v2/remote">
      <configurationRunningState xmlns="http://xml.spss.com/scoring-v2" state="SUSPENDED" id="nb-config"/>
    </changeConfigurationRunningState>
  </soapenv:Body>
</soapenv:Envelope>
```

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

SOAP response example

The server responds to a `changeConfigurationRunningState` 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>
    <d:changeConfigurationRunningStateResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
    </soapenv:Body>
  </soapenv:Envelope>
```

The `getConfigurationDetails` operation

Retrieves the parameter settings for a specified scoring configuration.

Input fields

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

Table 4-4
Fields for `getConfigurationDetails`

Field	Type/Valid Values	Description
id	string	Non-localized ID for the element. This should match the <code>spss_ss:idAttributeGroup</code> type.

Return information

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

Table 4-5
Return Value

Type	Description
configurationDetails	This element contains the information that's exchanged to view or adjust the configuration settings.

Java example

To retrieve information about a scoring configuration, supply the `getConfigurationDetails` operation with a string corresponding to the identifier for the configuration.

```
String configId = "nb-config";
ConfigurationDetails details = stub.getConfigurationDetails(configId);
```

The `getModelReference` method returns a `ModelReference` object containing information about the model associated with the configuration. Use the `getResourcePath` and `getLabel` methods to access the repository path and version label for the model.

```
ModelReference modelRef = details.getModelReference();
System.out.println("Associated model: " + modelRef.getResourcePath());
System.out.println("Version label of model: " + modelRef.getLabel());
```

The `getModelInputMetadata` method returns a `ModelInputMetadata` object containing information about the input fields for the model. Use the `getModelInputMetadataField` method to create an array containing an entry for each field. The `getName` and `getType` methods return the name and type for a field. The `getDescription` method returns the description of the field. Use the `getIsReturned` method to access a boolean indicating whether the model requires a value for the field or not.

```
ModelInputMetadata inputMetadata = details.getModelInputMetadata();
ModelInputMetadataField[] inputMetadataField = inputMetadata.getModelInputMetadataField();
System.out.println("Inputs:\n");
System.out.println("NAME\tTYPE\tDESCRIPTION\tRETURNED\n");
for (int i = 0; i < inputMetadataField.length; i++) {
    System.out.println(inputMetadataField[i].getName() + "\t" +
        inputMetadataField[i].getType() + "\t" +
        inputMetadataField[i].getDescription() + "\t" +
        inputMetadataField[i].getIsReturned());
}
```

The `getLogSettings` method returns a `LogSettings` object containing information about the settings for logging. Use the `getLogDestination` method to access the JMS destination of the information logged by the service.

The actual items being logged are organized into groups. Use the `getLoggableItemGroup` method to create an array containing an entry for each group. For each group, the `getLoggableItem` method returns an array containing the log items. The `getName`, `getId`, and `getIsEnabled` methods return details about a specific item.

```
LogSettings lSettings = details.getLogSettings();
System.out.println("Log Destination: " + lSettings.getLogDestination());
LoggableItemGroup[] lgroups = lSettings.getLoggableItemGroup();
for (int i = 0; i < lgroups.length; i++) {
    System.out.println(lgroups[i].getName() + " (" + lgroups[i].getId() + ")");
    LoggableItem[] logItem = lgroups[i].getLoggableItem();
    if (logItem.length > 0) {
        System.out.println("NAME\tID\tENABLED\n");
        for (int j = 0; j < logItem.length; j++) {
            System.out.println(logItem[j].getName() + "\t" +
```

```

        logItem[j].getId() + "\t" +
        logItem[j].getIsEnabled());
    }
}
}

```

The `getInputAttributeOrder` method returns an `InputAttributeOrder` object containing information about the order of the input fields for the model. Use the `getAttributes` method to create an array containing an entry for each field.

```

InputAttributeOrder inputOrder = details.getInputAttributeOrder();
String[] inputAttributes = inputOrder.getAttributes();
System.out.println("The order of the input attributes is:\n");
for (int i = 0; i < inputAttributes.length; i++) {
    System.out.println(inputAttributes[i]);
}

```

The `getConfigurationItem` method returns an array of `ConfigurationItem` objects containing information about model-specific configuration settings. The settings are defined using XML having a structure defined by the `schedParams.xsd` schema. Use the `getXML` method to access the XML definition for an item.

```

ConfigurationItem[] configItem = details.getConfigurationItem();
for (int i = 0; i < configItem.length; i++) {
    System.out.println("Configuration item for this configuration is:\n" +
        configItem[i].getXML());
}

```

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<ModelInputMetadataField> inputMetadataFieldList = inputMetadata.getModelInputMetadataField();
```

SOAP request example

Client invocation of the `getConfigurationDetails` 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>
  <getConfigurationDetails xmlns="http://xml.spss.com/scoring-v2/remote">
    <ns2:id xmlns:ns2="http://xml.spss.com/scoring-v2">nb-config</ns2:id>
  </getConfigurationDetails>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `getConfigurationDetails` 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>
    <d:getConfigurationDetailsResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
      <c:configurationDetails cfgSerial="0a010a07d3dc251c0000011eac84bfd18803" id="nb-config">
        <c:modelReference label="LATEST" resourcePath="/PMML/naivebayes.xml"
          id="0a010a07d3dc251c0000011eac84bfd18771"/>
        <c:modelInputMetadata isReturnEnabled="false">
          <c:modelInputMetadataField name="age" type="double"
            description="Age of Respondent" isReturned="false"/>
          <c:modelInputMetadataField name="educ" type="double"
            description="Highest Year of School Completed" isReturned="false"/>
          <c:modelInputMetadataField name="prestg80" type="double"
            description="R's Occupational Prestige Score (1980)" isReturned="false"/>
          <c:modelInputMetadataField name="speduc" type="double"
            description="Highest Year School Completed, Spouse" isReturned="false"/>
        </c:modelInputMetadata>
        <c:logSettings isEnabled="false" logDestination="queue/PASWLog">
          <c:loggableItemGroup name="Context Data" id="INPUT_CONTEXT"/>
          <c:loggableItemGroup name="Model Inputs" id="INPUT_MODEL">
            <c:loggableItem isEnabled="false" name="age" id="age"/>
            <c:loggableItem isEnabled="false" name="educ" id="educ"/>
            <c:loggableItem isEnabled="false" name="prestg80" id="prestg80"/>
            <c:loggableItem isEnabled="false" name="speduc" id="speduc"/>
          </c:loggableItemGroup>
        </c:logSettings>
      </c:configurationDetails>
    </d:getConfigurationDetailsResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

```

</c:loggableViewGroup>
<c:loggableViewGroup name="Model Outputs" id="OUTPUT">
  <c:loggableViewItem isEnabled="false" name="Prediction" id="Prediction"/>
  <c:loggableViewItem isEnabled="false" name="Prob" id="Prob"/>
  <c:loggableViewItem isEnabled="false" name="Prob-1.0" id="Prob-1.0"/>
  <c:loggableViewItem isEnabled="false" name="Prob-2.0" id="Prob-2.0"/>
  <c:loggableViewItem isEnabled="false" name="Confidence" id="Confidence"/>
</c:loggableViewGroup>
<c:loggableViewGroup name="Scoring Engine Properties" id="ENGINE_PROPERTY">
  <c:loggableViewItem isEnabled="false" name="Model Path" id="MODEL_PATH"/>
  <c:loggableViewItem isEnabled="false" name="Scoring Configuration Name"
    id="CONFIGURATION_NAME"/>
  <c:loggableViewItem isEnabled="false" name="Model Version Label" id="MODEL_LABEL"/>
  <c:loggableViewItem isEnabled="false" name="Model MIME type" id="MODEL_MIME_TYPE"/>
  <c:loggableViewItem isEnabled="false" name="Model Version Marker"
    id="MODEL_MARKER"/>
  <c:loggableViewItem isEnabled="false" name="Scoring Configuration Serial"
    id="CONFIGURATION_SERIAL"/>
  <c:loggableViewItem isEnabled="false" name="Model ID" id="MODEL_ID"/>
</c:loggableViewGroup>
<c:loggableViewGroup name="Score Provider Properties" id="PROVIDER_PROPERTY"/>
</c:logSettings>
<c:inputAttributeOrder>
  <c:attributes>age</attributes>
  <c:attributes>educ</attributes>
  <c:attributes>prestg80</attributes>
  <c:attributes>speduc</attributes>
</c:inputAttributeOrder>
<c:cacheSize value="1" min="1" max="100"/>
<c:batchEnabledFlag>false</c:batchEnabledFlag>
<c:configurationItem>
  <c:xml>&lt;?xml version="1.0"
    encoding="UTF-8" ?&gt; &lt;node
    name="Model Specific" objectType="Model
    Specific"&gt;&lt;boolean
    objectType="PredictionOnly" name="A flag for
    prediction only" required="true"
    value="false"/&gt;&lt;choice
    objectType="MissingValuePolicy" name="Missing
    value handling policy for this model"
    required="true"&gt;&lt;item
    value="Substitute: Different model has different substitute
    policy"/&gt;&lt;item value="Score as SYSMIS: Score
    result would be SYSMIS"/&gt;&lt;item value="Throw
    exception: A SCMissingPredictorException is
    thrown"/&gt;&lt;value>Score as SYSMIS: Score result
    would be
    SYSMIS&lt;/value&gt;&lt;/choice&gt;&lt;/node&gt;</c:xml>
  </c:configurationItem>
</c:configurationDetails>
</d:getConfigurationDetailsResponse>
</soapenv:Body>

```

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

The *getConfigurations* operation

Returns information about all scoring configurations in the system. For each configuration, the operation reports the following:

- Configuration identifier
- Path, identifier, and label for the model associated with the configuration
- Current status of the configuration

Return information

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

Table 4-6
Return Value

Type	Description
<code>configurationReference[]</code>	This element defines the format of the array element that is returned by the <code>getConfigurations</code> service call. It contains a reference to the model which this configuration is based upon, and the identifier of the scoring configuration itself.

Java example

Each entry of the array of `ConfigurationReference` objects returned by the `getConfigurations` operation corresponds to a scoring configuration in the system. Use the `getId` method to return the identifier for the configuration.

Information about the model associated with the scoring configuration can be obtained from the `ModelReference` object returned by the `getModelReference` method. The `getResourcePath` method returns the repository path to the model. The `getLabel` method returns the label identifying the model version being used.

```
ConfigurationReference[] configRef = stub.getConfigurations();
for (int i = 0; i < configRef.length; i++) {
    System.out.println("Configuration ID: " + configRef[i].getId());

    ModelReference modelRef = configRef[i].getModelReference();
    System.out.println("Associated model: " + modelRef.getResourcePath());
    System.out.println("Version label of model: " + modelRef.getLabel());

    ConfigurationStatus configStat = configRef[i].getConfigurationStatus();
    System.out.println("Status: (" + configStat.getStatusCode().toString() +
        ") " + configStat.getMessage());
}
```


The `getConfigurationStatus` method returns a `ConfigurationStatus` object containing information about the current status for the configuration. This information consists of a status code and a status message. The code indicates whether the message represents an error, a warning, or informational text.

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<ConfigurationReference> configRefList = stub.getConfigurations();
for (ConfigurationReference configRef : configRefList)
{
    System.out.println("Configuration ID: " + configRef.getId());
    ModelReference modelRef = configRef.getModelReference();
    System.out.println("Associated model: " + modelRef.getResourcePath());
    System.out.println("Version label of model: " + modelRef.getLabel());
    ConfigurationStatus configStat = configRef.getConfigurationStatus();
    System.out.println("Status: (" + configStat.getStatusCode()).toString() +
        ") " + configStat.getMessage());
}
```

SOAP request example

Client invocation of the `getConfigurations` 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>
    <getConfigurations xmlns="http://xml.spss.com/scoring-v2/remote"/>
  </soapenv:Body>
```

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

SOAP response example

The server responds to a `getConfigurations` 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>
    <d:getConfigurationsResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
      <c:configurationReference id="nb-config">
        <c:modelReference label="LATEST" resourcePath="/PMML/naivebayes.xml"
          id="0a010a07d3dc251c0000011eac84bfd18771"/>
        <c:configurationStatus statusCode="INFORMATION" message="Started"/>
      </c:configurationReference>
      <c:configurationReference id="test">
        <c:modelReference label="LATEST" resourcePath="/CSGLM_model.xml"
          id="0a010a0774bd18050000011ea8b211d98040"/>
        <c:configurationStatus statusCode="INFORMATION" message="Started"/>
      </c:configurationReference>
    </d:getConfigurationsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The getMetadata operation

Retrieves information about the input and output fields for a specified scoring configuration. For each field, this information includes the following:

- Name
- Data type for the field
- An optional description

For input fields, an optional indicator of whether or not a value of the field is required for scoring may be included. In addition, for categorical fields, the metadata may include a list of valid values.

This operation is often used to provide information for interfaces that prompt a user for model input values on which scoring will be based.

Input fields

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

Table 4-7
Fields for `getMetadata`

Field	Type/Valid Values	Description
id	string	Non-localized ID for the element. This should match the <code>spss_ss:idAttributeGroup</code> type.

Return information

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

Table 4-8
Return Value

Type	Description
<code>metadataResult</code>	This element combines the metadata input and output data into one element, which is used by the <code>getMetadata</code> service call.

Java example

Supply the `getMetadata` operation with a string denoting the identifier for the scoring configuration. The resulting `MetadataResult` object contains two types of objects, `MetadataInputField` and `MetadataOutputField`. Use the `getMetadataInputField` method to return an array of the input objects. The `getMetadataOutputField` method returns an array of the output objects, which represent the scoring functions used for the model.

For both input and output objects, use the `getName`, `getType`, and `getDescription` methods to obtain information about the fields. In addition, for input objects only, the `getIsRequired` method returns a boolean indicating whether or not a value for the field is required.

```
String configId = "config-glm";
MetadataResult result = stub.getMetadata(configId);

MetadataInputField[] inputs = result.getMetadataInputField();
System.out.println("Inputs:\n");
System.out.println("NAME\tTYPE\tDESCRIPTION\tREQUIRED\n");
for (int i = 0; i < inputs.length; i++) {
    System.out.println(inputs[i].getName() + "\t" + inputs[i].getType() +
        "\t" + inputs[i].getDescription() + "\t" + inputs[i].getIsRequired());
}

MetadataOutputField[] outputs = result.getMetadataOutputField();
System.out.println("Outputs:\n");
System.out.println("NAME\tTYPE\tDESCRIPTION\n");
for (int i = 0; i < outputs.length; i++) {
    System.out.println(outputs[i].getName() + "\t" + outputs[i].getType() +
        "\t" + outputs[i].getDescription());
}
```

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<MetadataInputField> inputsList = result.getMetadataInputField();
```

SOAP request example

Client invocation of the `getMetadata` 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>
      <getMetadata xmlns="http://xml.spss.com/scoring-v2/remote">
        <ns2:id xmlns:ns2="http://xml.spss.com/scoring-v2">config-glm</ns2:id>
      </getMetadata>
    </soapenv:Body>
  </soapenv:Envelope>
```

SOAP response example

The server responds to a `getMetadata` 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>
```

```

<d:getMetadataResponse xmlns:a="http://xml.spss.com/data"
  xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
  xmlns:d="http://xml.spss.com/scoring-v2/remote"
  xmlns:e="http://xml.spss.com/scoring/exception">
  <c:metadataResult>
    <c:metadataInputField name="happy" type="double" description="General Happiness"
      isRequired="true"/>
    <c:metadataInputField name="prestg80" type="double"
      description="R's Occupational Prestige Score (1980)" isRequired="true"/>
    <c:metadataInputField name="tax" type="double" description="R's Federal Income Tax"
      isRequired="true"/>
    <c:metadataOutputField name="Prediction" type="double" description=""/>
    <c:metadataOutputField name="StdDev" type="double" description=""/>
  </c:metadataResult>
</d:getMetadataResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The *getMetricItems* operation

Returns all metric items available for a specified configuration. To retrieve the value for a particular metric, use the *getMetricValue* operation.

Input fields

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

Table 4-9
Fields for *getMetricItems*

Field	Type/Valid Values	Description
id	string	Non-localized ID for the element. This should match the <code>spss_ss:idAttributeGroup</code> type.

Return information

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

Table 4-10
Return Value

Type	Description
<code>metricItem[]</code>	This element describes a single metric entry (score performance, etc).

Java example

To access the metric items, supply the *getMetricItems* operation with a string corresponding to the identifier for the scoring configuration. The name, identifier, unit, and scale for any metric item in the returned array can be obtained using the *getName*, *getId*, *getUnit*, and *getScale* methods.

```
String configId = "test";
MetricItem[] item = stub.getMetricItems(configId);
System.out.println("NAME\tID\tUNIT\tSCALE");
for (int i = 0; i < item.length; i++) {
    System.out.println(item[i].getName() + "\t" + item[i].getId() +
        "\t" + item[i].getUnit() + "\t" + item[i].getScale());
}
```

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 configId = "test";
List itemList = stub.getMetricItems(configId);
System.out.println("NAME\tID\tUNIT\tSCALE");
for (MetricItem item : itemList)
{
    System.out.println(item.getName() + "\t" + item.getId() +
        "\t" + item.getUnit() + "\t" + item.getScale());
}
```

SOAP request example

Client invocation of the `getMetricItems` 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>
    <getMetricItems xmlns="http://xml.spss.com/scoring-v2/remote">
      <ns2:id xmlns:ns2="http://xml.spss.com/scoring-v2">test</ns2:id>
```

```

    </getMetricItems>
  </soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `getMetricItems` 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>
    <d:getMetricItemsResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
      <c:metricItem scale="3" unit="milliseconds" name="Minimum Computation Wait Time"
        id="CONFIGURATION_COMPUTATION_WAIT_TIME_MINIMUM"/>
      <c:metricItem scale="0" unit="hits" name="Cache Hits" id="CONFIGURATION_CACHE_HITS"/>
      <c:metricItem scale="0" unit="scores" name="Service Scores" id="SERVICE_TOTAL_SCORES"/>
      <c:metricItem scale="3" unit="milliseconds" name="Average Data Initialization Time"
        id="CONFIGURATION_DATA_INIT_TIME_AVERAGE"/>
      <c:metricItem scale="3" unit="milliseconds" name="Minimum Log Queue Time"
        id="CONFIGURATION_LOG_QUEUE_TIME_MINIMUM"/>
      <c:metricItem scale="3" unit="milliseconds" name="Maximum Data Initialization Time"
        id="CONFIGURATION_DATA_INIT_TIME_MAXIMUM"/>
      <c:metricItem scale="3" unit="milliseconds" name="Minimum Latency"
        id="CONFIGURATION_RESPONSE_TIME_MINIMUM"/>
      <c:metricItem scale="0" unit="seconds" name="Service Uptime" id="SERVICE_UPTIME"/>
      <c:metricItem scale="3" unit="milliseconds" name="Average Computation Wait Time"
        id="CONFIGURATION_COMPUTATION_WAIT_TIME_AVERAGE"/>
      <c:metricItem scale="3" unit="milliseconds" name="Maximum Computation Wait Time"
        id="CONFIGURATION_COMPUTATION_WAIT_TIME_MAXIMUM"/>
      <c:metricItem scale="3" unit="milliseconds" name="Average Log Queue Time"
        id="CONFIGURATION_LOG_QUEUE_TIME_AVERAGE"/>
      <c:metricItem scale="3" unit="milliseconds" name="Minimum Computation Time"
        id="CONFIGURATION_COMPUTATION_TIME_MINIMUM"/>
      <c:metricItem scale="3" unit="milliseconds" name="Minimum Log Serialization Time"
        id="CONFIGURATION_LOG_SERIALIZE_TIME_MINIMUM"/>
      <c:metricItem scale="3" unit="milliseconds" name="Average Latency"
        id="CONFIGURATION_RESPONSE_TIME_AVERAGE"/>
      <c:metricItem scale="3" unit="milliseconds" name="Maximum Log Queue Time"
        id="CONFIGURATION_LOG_QUEUE_TIME_MAXIMUM"/>
      <c:metricItem scale="3" unit="milliseconds" name="Maximum Latency"
        id="CONFIGURATION_RESPONSE_TIME_MAXIMUM"/>
      <c:metricItem scale="0" unit="scores" name="Configuration Scores"
        id="CONFIGURATION_TOTAL_SCORES"/>
      <c:metricItem scale="3" unit="milliseconds" name="Minimum Data Access Time"
        id="CONFIGURATION_DATA_ACCESS_TIME_MINIMUM"/>
    </d:getMetricItemsResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

```

<c:metricItem scale="3" unit="milliseconds" name="Average Computation Time"
  id="CONFIGURATION_COMPUTATION_TIME_AVERAGE"/>
<c:metricItem scale="3" unit="milliseconds" name="Average Log Serialization Time"
  id="CONFIGURATION_LOG_SERIALIZE_TIME_AVERAGE"/>
<c:metricItem scale="3" unit="milliseconds" name="Maximum Computation Time"
  id="CONFIGURATION_COMPUTATION_TIME_MAXIMUM"/>
<c:metricItem scale="0" unit="misses" name="Cache Misses"
  id="CONFIGURATION_CACHE_MISSES"/>
<c:metricItem scale="3" unit="milliseconds" name="Maximum Log Serialization Time"
  id="CONFIGURATION_LOG_SERIALIZE_TIME_MAXIMUM"/>
<c:metricItem scale="0" unit="seconds" name="Configuration Uptime"
  id="CONFIGURATION_UPTIME"/>
<c:metricItem scale="3" unit="milliseconds" name="Average Data Access Time"
  id="CONFIGURATION_DATA_ACCESS_TIME_AVERAGE"/>
<c:metricItem scale="3" unit="milliseconds" name="Maximum Data Access Time"
  id="CONFIGURATION_DATA_ACCESS_TIME_MAXIMUM"/>
<c:metricItem scale="3" unit="milliseconds" name="Minimum Data Initialization Time"
  id="CONFIGURATION_DATA_INIT_TIME_MINIMUM"/>
</d:getMetricItemsResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The getMetricValue operation

Returns the value for a specified metric of a scoring configuration. The list of metric items available for a configuration can be retrieved using the `getMetricItems` operation. Valid metric identifiers include:

- SERVICE_TOTAL_SCORES
- SERVICE_UPTIME
- CONFIGURATION_RESPONSE_TIME_AVERAGE
- CONFIGURATION_RESPONSE_TIME_MINIMUM
- CONFIGURATION_RESPONSE_TIME_MAXIMUM
- CONFIGURATION_DATA_INIT_TIME_AVERAGE
- CONFIGURATION_DATA_INIT_TIME_MINIMUM
- CONFIGURATION_DATA_INIT_TIME_MAXIMUM
- CONFIGURATION_DATA_ACCESS_TIME_AVERAGE
- CONFIGURATION_DATA_ACCESS_TIME_MINIMUM
- CONFIGURATION_DATA_ACCESS_TIME_MAXIMUM
- CONFIGURATION_COMPUTATION_WAIT_TIME_AVERAGE
- CONFIGURATION_COMPUTATION_WAIT_TIME_MINIMUM
- CONFIGURATION_COMPUTATION_WAIT_TIME_MAXIMUM
- CONFIGURATION_COMPUTATION_TIME_AVERAGE
- CONFIGURATION_COMPUTATION_TIME_MINIMUM

- CONFIGURATION_COMPUTATION_TIME_MAXIMUM
- CONFIGURATION_LOG_SERIALIZE_TIME_AVERAGE
- CONFIGURATION_LOG_SERIALIZE_TIME_MINIMUM
- CONFIGURATION_LOG_SERIALIZE_TIME_MAXIMUM
- CONFIGURATION_LOG_QUEUE_TIME_AVERAGE
- CONFIGURATION_LOG_QUEUE_TIME_MINIMUM
- CONFIGURATION_LOG_QUEUE_TIME_MAXIMUM
- CONFIGURATION_TOTAL_SCORES
- CONFIGURATION_UPTIME
- CONFIGURATION_CACHE_HITS
- CONFIGURATION_CACHE_MISSES

Input fields

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

Table 4-11
Fields for getMetricValue

Field	Type/Valid Values	Description
id	string	Non-localized ID for the element. This should match the <code>spss_ss:idAttributeGroup</code> type.
metricId	string	Metric item ID

Return information

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

Table 4-12
Return Value

Type	Description
metricValue	This element contains a single metric value.

Java example

To obtain a value of a metric item for a configuration, supply the `getMetricValue` operation with strings denoting the configuration and metric identifiers. Use the `getValue` method of the returned `MetricValue` object to access the value.

The following sample reports the uptime for the `test` configuration.

```
String configId = "test";
String metricId = "CONFIGURATION_UPTIME";
MetricValue val = stub.getMetricValue(configId, metricId);
System.out.format("%s Configuration\n%s = %f", configId, metricId, val.getValue());
```

SOAP request example

Client invocation of the `getMetricValue` 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>
    <getMetricValue xmlns="http://xml.spss.com/scoring-v2/remote">
      <ns2:id xmlns:ns2="http://xml.spss.com/scoring-v2">test</ns2:id>
      <metricId>CONFIGURATION_UPTIME</metricId>
    </getMetricValue>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `getMetricValue` 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>
    <d:getMetricValueResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
```

```

    <c:metricValue value="0.0"/>
  <d:/getMetricValueResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The *getScore* operation

Generates scores based on supplied inputs for a specified scoring configuration. The types of scores returned depend on the model being used for scoring.

If the configuration being used for scoring does not specify a data provider, the `getScore` call must provide all required input data in the scoring request. This data can be specified as a set of name/value pairs for the input fields used by the model or as tabular context data in which the columns correspond to the input fields and the row entries identify the input values. The format for input values must match the data type, as specified in the configuration details and metadata. For non-numeric data types, the format must be as follows:

- boolean = *true* (case insensitive) or *1*, or *false* (case insensitive) or *0*
- date = *yyyy-MM-dd*
- daytime = *HH:mm:ss*
- timestamp = *yyyy-MM-dd'T'HH:mm:ss*

The results returned by the operation include the following information:

- Input values on which the scores are based
- A tabular structure in which the columns correspond to the scoring functions returned with the row entries indicating the value of the scoring functions

Input fields

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

Table 4-13
Fields for *getScore*

Field	Type/Valid Values	Description
scoreRequest	scoreRequest	This element contains the input(s) of a <code>getScore</code> call. Note, due to the possible impact upon performance, this element is as lightweight as possible (minimized XML overhead).

Return information

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

Table 4-14
Return Value

Type	Description
scoreResult	This element contains the output(s) and details of a <code>getScore</code> call. Note, due to the possible impact upon performance, this element is as lightweight as possible (minimized XML overhead).

Java example

To generate a score:

1. Create a `ScoreRequest` object.
2. Use the `setId` method to assign a string corresponding to the identifier of the scoring configuration to use for the score.
3. Create `Input` objects for the input values on which to base the score. Use the `setName` and `setValue` methods for each object to assign values to specific input fields. When complete, use the `setRequestInputTable` method to assign the input values to the request object.
4. If the scoring model uses context data, create a `TableType` object for the context data table. Use the `setName` and `setRowValues` methods to assign values to specific context fields. When complete, use the `setContext` method to assign the context table to the request object.
5. Provide the `getScore` operation with the request object.

The following sample provides the `config-glm` scoring configuration with a request input table consisting of two rows containing values for three input fields to generate a score.

```
ScoreRequest request = new ScoreRequest();
request.setId("config-glm");
```

```
Input[][] inputs = new Input[1][2][3];
inputs[0][0][0].setName("happy");
inputs[0][0][0].setValue("1");
inputs[0][0][1].setName("prestg80");
inputs[0][0][1].setValue("1");
inputs[0][0][2].setName("tax");
inputs[0][0][2].setValue("1");
inputs[0][1][0].setName("happy");
inputs[0][1][0].setValue("3");
inputs[0][1][1].setName("prestg80");
inputs[0][1][1].setValue("2");
inputs[0][1][2].setName("tax");
inputs[0][1][2].setValue("4");
request.setRequestInputTable(inputs);
```

```
ScoreResult result=stub.getScore(request);
```

For the `ScoreResult` object returned by the operation, the `getReturnedRequestInputValue` and `getReturnedDPDOutputValue` methods return arrays of `ReturnedRequestInputValue` and `ReturnedDPDOutputValue` objects, respectively, from which the values for the scoring inputs

can be retrieved. Use the `getColumnNames` and `getRowValues` methods for the result object to access the scores.

```
ReturnedRequestInputValue[][][] inputValue = result.getReturnedRequestInputValue();
System.out.print("Request Inputs:\n");
for (int t=0; t < inputValue.length; t++) {
    System.out.println("Table " + t + "\n");
    for (int r=0; r < inputValue[t].length; r++) {
        for (int c=0; c < inputValue[t][r].length; c++) {
            System.out.print(inputValue[t][r][c].getValue() + "\t");
        }
        System.out.println("\n");
    }
    System.out.println("\n");
}

System.out.print("Outputs:\n");
String[] columns = result.getColumnNames();
RowValues[] rValues = result.getRowValues();
for (int r=0; r < rValues.length; r++) {
    Value[] val = rValues.getValue();
    for (int c=0; c < val.length; c++) {
        System.out.print(columns[c] + " = " + val[c].getValue() + "\n");
    }
    System.out.print("\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:

```
ScoreRequest request = new ScoreRequest();
request.setIid("config-glm");

RequestInputTable rInputTable = new RequestInputTable();
rInputTable.setName("Table1");
RequestInputRow rInputRow = new RequestInputRow();

Input inputs = new Input();
inputs.setName("happy");
inputs.setValue("1");
rInputRow.getInput().add(inputs);
inputs.setName("prestg80");
inputs.setValue("1");
rInputRow.getInput().add(inputs);
inputs.setName("tax");
inputs.setValue("1");
rInputRow.getInput().add(inputs);

rInputTable.getRequestInputRow().add(rInputRow);
request.getRequestInputTable().add(rInputTable);

ScoreResult result=stub.getScore(request);
```

```

List<ReturnedRequestInputTable> rRequestInputTableList = result.getReturnedRequestInputTable();
System.out.print("Inputs:\n");
for (ReturnedRequestInputTable rRequestInputTable : rRequestInputTableList)
{
    List<ReturnedRequestInputRow> rRequestInputRowList = rRequestInputTable.getReturnedRequestInputRow();
    for (ReturnedRequestInputRow rRequestInputRow : rRequestInputRowList)
    {
        List<ReturnedRequestInputValue> rRequestInputValueList = rRequestInputRow.getReturnedRequestInputValue();
        for (ReturnedRequestInputValue rRequestInputValue : rRequestInputValueList)
        {
            System.out.print(rRequestInputValue.getName() + "=" + rRequestInputValue.getValue() + "\n");
        }
    }
}

System.out.print("Outputs:\n");
List<String> colNameList = result.getColumnNames().getName();
for (String colName : colNameList)
{
    System.out.print(colName + "\t");
}
System.out.print("\n");
List<RowValues> rowValuesList = result.getRowValues();
for (RowValues rowValues : rowValuesList)
{
    List<Value> valueList = rowValues.getValue();
    for (Value value : valueList)
    {
        System.out.print(value.getValue() + "\t");
    }
    System.out.print("\n");
}

```

SOAP request example

Client invocation of the `getScore` 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>
  <getScore xmlns="http://xml.spss.com/scoring-v2/remote">
    <scoreRequest id="config-glm" xmlns="http://xml.spss.com/scoring-v2">
      <input name="happy">1</input>
      <input name="prestg80">1</input>
      <input name="tax">1</input>
    </scoreRequest>
  </getScore>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `getScore` 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>
    <d:getScoreResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
      <c:scoreResult id="4eddd000-e737-11dd-8971-94ff11922566">
        <c:columnNames>
          <c:name>Prediction</c:name>
          <c:name>StdDev</c:name>
        </c:columnNames>
        <c:rowValues>
          <c:value>44.5836</c:value>
          <c:value>2.89975</c:value>
        </c:rowValues>
        <c:modelInputValue name="prestg80" type="double">1.0</c:modelInputValue>
        <c:modelInputValue name="tax" type="double">1.0</c:modelInputValue>
        <c:modelInputValue name="happy" type="double">1.0</c:modelInputValue>
      </c:scoreResult>
    </d:getScoreResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

The *getServiceDetails* operation

Returns information about each registered scoring provider available in the system. This information includes the following:

- Provider name
- Internal identifier for the provider
- Version of the provider
- MIME types of files that can be scored using the provider

Return information

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

Table 4-15
Return Value

Type	Description
<code>scoringServiceDetails</code>	This element is a collection of information that provides details about the scoring service.

Java example

After calling the `getServiceDetails` operation, use the `getVersion` method for the resulting `ScoringServiceDetails` object to return the version number for the Scoring Service.

The `getScoreProviderDetails` method returns an array of `ScoreProviderDetails` objects containing information about the scoring providers. Use the `getName`, `getVersion`, and `getSupportedMimeType` methods to access the name, version, and MIME types for each provider.

```
ScoringServiceDetails details = stub.getServiceDetails();
System.out.println("Service version: " + details.getVersion());
ScoreProviderDetails[] providerArray = details.getScoreProviderDetails();
for (int i = 0; i < providerArray.length; i++) {
    System.out.println("Provider: " + providerArray[i].getName() +
        " (version " + providerArray[i].getVersion() + ")");
    String[] mimeTypeTypes = providerArray[i].getSupportedMimeType();
    System.out.println("Used for:");
    for (int j = 0; j < mimeTypeTypes.length; j++) {
        System.out.println(mimeTypeTypes[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:

```
List<ScoreProviderDetails> providerList = details.getScoreProviderDetails();
```


SOAP request example

Client invocation of the `getServiceDetails` 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>
      <getServiceDetails xmlns="http://xml.spss.com/scoring-v2/remote"/>
    </soapenv:Body>
  </soapenv:Envelope>
```

SOAP response example

The server responds to a `getServiceDetails` 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>
    <d:getServiceDetailsResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
      <c:scoringServiceDetails>
        <c:version>4.00.000.133</c:version>
        <c:scoreProviderDetails name="Score Provider - SmartScore">
```

```

        id="0a010a07d522b9690000011ea89ab3f08176">
        <c:version>1.0</c:version>
        <c:supportedMimeTypes>application/x-vnd.spss-pmml</c:supportedMimeTypes>
    </c:scoreProviderDetails>
    <c:scoreProviderDetails name="Modeler Score Provider"
        id="0a010a07d522b9690000011ea89ab3f08309">
        <c:version>1.0</c:version>
        <c:supportedMimeTypes>application/x-vnd.spss-clementine-stream</c:supportedMimeTypes>
        <c:supportedMimeTypes>application/x-vnd.spss-scenario</c:supportedMimeTypes>
    </c:scoreProviderDetails>
</c:scoringServiceDetails>
</d:getServiceDetailsResponse>
</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 4-16
Return Value

Type	Description
string	The service version number.

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/scoring-v2/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>
    <d:getVersionResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
      <d:version>5.0.0.0.000</d:version>
    </d:getVersionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The ping operation

Verifies that the scoring service is available.

SOAP request example

Client invocation of the `ping` 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>
    <ping xmlns="http://xml.spss.com/scoring-v2/remote"/>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response example

The server responds to a `ping` 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>
    <d:pingResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2">
```

```

xmlns:d="http://xml.spss.com/scoring-v2/remote"
xmlns:e="http://xml.spss.com/scoring/exception">
</soapenv:Body>
</soapenv:Envelope>

```

The removeConfiguration operation

Removes a specified scoring configuration from the system, allowing any in-process score requests to complete while preventing any score requests received after the delete request from initiating. The operation generates an entry in the service logging tables indicating the configuration was deleted.

Input fields

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

Table 4-17
Fields for removeConfiguration

Field	Type/Valid Values	Description
id	string	Non-localized ID for the element. This should match the spss_ss:idAttributeGroup type.

Java example

To delete a scoring configuration, supply the removeConfiguration operation with a string corresponding to the identifier for the configuration to be deleted.

```

String configId = "nb-config";
stub.removeConfiguration(configId);

```

SOAP request example

Client invocation of the removeConfiguration 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>
  <removeConfiguration xmlns="http://xml.spss.com/scoring-v2/remote">
    <ns2:id xmlns:ns2="http://xml.spss.com/scoring-v2">nb-config</ns2:id>
  </removeConfiguration>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `removeConfiguration` 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>
    <d:removeConfigurationResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
    </d:removeConfigurationResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

The `setConfigurationDetails` operation

Assigns values to configuration settings for a scoring configuration. Scoring requests for an existing configuration are queued until the new settings are applied.

Input fields

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

Table 4-18

Fields for `setConfigurationDetails`

Field	Type/Valid Values	Description
<code>configurationDetails</code>	<code>configurationDetails</code>	This element contains the information that's exchanged to view or adjust the configuration settings.

Java example

The following sample modifies the default `ConfigurationDetails` object created by the `buildConfigurationDetails` operation. The default configuration contains no identifier so use the `setId` operation to assign one. In addition, the input metadata fields are modified to be returned with any score results. The resulting configuration is then stored by calling the `setConfigurationDetails` operation.

```

ModelReference model = new ModelReference();
modelRef.setId("0a010a07d3dc251c0000011eac84bfd18771");
modelRef.setResourcePath("/PMML/naivebayes.xml");
modelRef.setLabel("LATEST");
ConfigurationDetails details = stub.buildConfigurationDetails(model);

details.setId("nb-config");

ModelInputMetadata inputMetadata = details.getModelInputMetadata();
inputMetadata.setIsReturnEnabled(true);
ModelInputMetadataField[] inputMetadataField = inputMetadata.getModelInputMetadataField();
for (int i = 0; i < inputMetadataField.length; i++) {
    inputMetadataField[i].setIsReturned(true);
}
inputMetadata.setModelInputMetadataField(inputMetadataField);
details.setModelInputMetadata(inputMetadata);

stub.setConfigurationDetails(details);

```

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:

```

ModelReference model = new ModelReference();
modelRef.setId("0a010a07d3dc251c0000011eac84bfd18771");
modelRef.setResourcePath("/PMML/naivebayes.xml");
modelRef.setLabel("LATEST");
ConfigurationDetails details = stub.buildConfigurationDetails(model);

details.setId("nb-config");
ModelInputMetadata inputMetadata = details.getModelInputMetadata();
inputMetadata.setIsReturnEnabled(true);
List inputMetadataFieldList = inputMetadata.getModelInputMetadataField();
for (ModelInputMetadataField inputMetadataField : inputMetadataFieldList)
{
    inputMetadataField.setIsReturned(true);
}
inputMetadata.setModelInputMetadataField(inputMetadataField);
details.setModelInputMetadata(inputMetadata);
stub.setConfigurationDetails(details);

```

SOAP request example

Client invocation of the `setConfigurationDetails` 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>
    <setConfigurationDetails xmlns="http://xml.spss.com/scoring-v2/remote">
      <configurationDetails cfgSerial="21c881e0-ddc4-11dd-aca9-dcb0b853b9a7" id="nb-config"
        xmlns="http://xml.spss.com/scoring-v2">
        <modelReference label="LATEST" resourcePath="/PMML/naivebayes.xml"
          id="0a010a07d3dc251c000011eac84bfd18771"/>
        <modelInputMetadata isReturnEnabled="true">
          <modelInputMetadataField name="age" type="double"
            description="Age of Respondent" isReturned="true"/>
          <modelInputMetadataField name="educ" type="double"
            description="Highest Year of School Completed" isReturned="true"/>
          <modelInputMetadataField name="prestg80" type="double"
            description="R's Occupational Prestige Score (1980)" isReturned="true"/>
          <modelInputMetadataField name="speduc" type="double"
            description="Highest Year School Completed, Spouse" isReturned="true"/>
        </modelInputMetadata>
        <logSettings isEnabled="false" logDestination="queue/PASWLog">
          <loggableItemGroup name="Context Data" id="INPUT_CONTEXT"/>
          <loggableItemGroup name="Model Inputs" id="INPUT_MODEL">
            <loggableItem isEnabled="false" name="age" id="age"/>
            <loggableItem isEnabled="false" name="educ" id="educ"/>
            <loggableItem isEnabled="false" name="prestg80" id="prestg80"/>
            <loggableItem isEnabled="false" name="speduc" id="speduc"/>
          </loggableItemGroup>
        </logSettings>
      </configurationDetails>
    </setConfigurationDetails>
  </soapenv:Body>
</soapenv:Envelope>
```

```

</loggableItemGroup>
<loggableItemGroup name="Model Outputs" id="OUTPUT">
  <loggableItem isEnabled="false" name="Prediction" id="Prediction"/>
  <loggableItem isEnabled="false" name="Prob" id="Prob"/>
  <loggableItem isEnabled="false" name="Prob-1.0" id="Prob-1.0"/>
  <loggableItem isEnabled="false" name="Prob-2.0" id="Prob-2.0"/>
  <loggableItem isEnabled="false" name="Confidence" id="Confidence"/>
</loggableItemGroup>
<loggableItemGroup name="Scoring Engine Properties" id="ENGINE_PROPERTY">
  <loggableItem isEnabled="false" name="Model Path" id="MODEL_PATH"/>
  <loggableItem isEnabled="false" name="Scoring Configuration Name"
    id="CONFIGURATION_NAME"/>
  <loggableItem isEnabled="false" name="Model Version Label" id="MODEL_LABEL"/>
  <loggableItem isEnabled="false" name="Model MIME type" id="MODEL_MIME_TYPE"/>
  <loggableItem isEnabled="false" name="Model Version Marker"
    id="MODEL_MARKER"/>
  <loggableItem isEnabled="false" name="Scoring Configuration Serial"
    id="CONFIGURATION_SERIAL"/>
  <loggableItem isEnabled="false" name="Model ID" id="MODEL_ID"/>
</loggableItemGroup>
<loggableItemGroup name="Score Provider Properties" id="PROVIDER_PROPERTY"/>
</logSettings>
<inputAttributeOrder>
  <attributes>age</attributes>
  <attributes>educ</attributes>
  <attributes>prestg80</attributes>
  <attributes>speduc</attributes>
</inputAttributeOrder>
<cacheSize value="1" min="1" max="100"/>
<batchEnabledFlag>false</batchEnabledFlag>
<configurationItem>
  <xml>&lt;?xml version="1.0"
    encoding="UTF-8" ?&gt; &lt;node
    name="Model Specific" objectType="Model
    Specific"&gt;&lt;boolean
    objectType="PredictionOnly" name="A flag for
    prediction only" required="true"
    value="false"/&gt;&lt;choice
    objectType="MissingValuePolicy" name="Missing
    value handling policy for this model"
    required="true"&gt;&lt;item
    value="Substitute: Different model has different substitute
    policy"/&gt;&lt;item value="Score as SYSMIS: Score
    result would be SYSMIS"/&gt;&lt;item value="Throw
    exception: A SCMissingPredictorException is
    thrown"/&gt;&lt;value>Score as SYSMIS: Score result
    would be
    SYSMIS&lt;/value&gt;&lt;/choice&gt;&lt;/node&gt;</xml>
  </configurationItem>
</configurationDetails>
</setConfigurationDetails>
</soapenv:Body>

```



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

SOAP response example

The server responds to a `setConfigurationDetails` 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>
    <d:setConfigurationDetailsResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
    </soapenv:Body>
  </soapenv:Envelope>
```

The updateConfigurationDetails operation

Updates values of configuration settings for an existing scoring configuration. Scoring requests are queued until the updated settings are applied.

Input fields

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

Table 4-19
Fields for `updateConfigurationDetails`

Field	Type/Valid Values	Description
configurationDetails	configurationDetails	This element contains the information that's exchanged to view or adjust the configuration settings.

Return information

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

Table 4-20
Return Value

Type	Description
configurationDetails	This element contains the information that's exchanged to view or adjust the configuration settings.

Java example

To update a scoring configuration:

1. Retrieve a `ConfigurationDetails` object for the configuration to be modified using the `getConfigurationDetails` operation.
2. Update the details object as desired.
3. Supply the `updateConfigurationDetails` operation

The following sample updates the *nb-config* configuration. The input metadata fields are modified to be omitted from any score results. The updated information is then stored by calling the `updateConfigurationDetails` operation.

```
String configId = "nb-config";
ConfigurationDetails details = stub.getConfigurationDetails(configId);

ModelInputMetadata inputMetadata = details.getModelInputMetadata();
inputMetadata.setIsReturnEnabled(false);
ModelInputMetadataField[] inputMetadataField = inputMetadata.getModelInputMetadataField();
for (int i = 0; i < inputMetadataField.length; i++) {
    inputMetadataField[i].setIsReturned(false);
}
inputMetadata.setModelInputMetadataField(inputMetadataField);
details.setModelInputMetadata(inputMetadata);

ConfigurationDetails updatedDetails = stub.updateConfigurationDetails(details);
```

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 configId = "nb-config";
ConfigurationDetails details = stub.getConfigurationDetails(configId);
ModelInputMetadata inputMetadata = details.getModelInputMetadata();
inputMetadata.setIsReturnEnabled(false);
List inputMetadataFieldList = inputMetadata.getModelInputMetadataField();
for (ModelInputMetadataField inputMetadataField : inputMetadataFieldList)
{ inputMetadataField.setIsReturned(false);
}
inputMetadata.setModelInputMetadataField(inputMetadataField);
details.setModelInputMetadata(inputMetadata);
ConfigurationDetails updatedDetails = stub.updateConfigurationDetails(details);
```

SOAP request example

Client invocation of the `updateConfigurationDetails` 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>
    <updateConfigurationDetails xmlns="http://xml.spss.com/scoring-v2/remote">
      <configurationDetails cfgSerial="21c881e0-ddc4-11dd-aca9-dcb0b853b9a7" id="nb-config"
        xmlns="http://xml.spss.com/scoring-v2">
        <modelReference label="LATEST" resourcePath="/PMML/naivebayes.xml"
          id="0a010a07d3dc251c0000011eac84bfd18771"/>
        <modelInputMetadata isReturnEnabled="false">
          <modelInputMetadataField name="age" type="double"
            description="Age of Respondent" isReturned="false"/>
          <modelInputMetadataField name="educ" type="double"
            description="Highest Year of School Completed" isReturned="false"/>
          <modelInputMetadataField name="prestg80" type="double"
            description="R's Occupational Prestige Score (1980)" isReturned="false"/>
          <modelInputMetadataField name="speduc" type="double"
            description="Highest Year School Completed, Spouse" isReturned="false"/>
        </modelInputMetadata>
        <logSettings isEnabled="false" logDestination="queue/PASWLog">
          <loggableItemGroup name="Context Data" id="INPUT_CONTEXT"/>
          <loggableItemGroup name="Model Inputs" id="INPUT_MODEL">
            <loggableItem isEnabled="false" name="age" id="age"/>
            <loggableItem isEnabled="false" name="educ" id="educ"/>
            <loggableItem isEnabled="false" name="prestg80" id="prestg80"/>
            <loggableItem isEnabled="false" name="speduc" id="speduc"/>
          </loggableItemGroup>
          <loggableItemGroup name="Model Outputs" id="OUTPUT">
            <loggableItem isEnabled="false" name="Prediction" id="Prediction"/>
            <loggableItem isEnabled="false" name="Prob" id="Prob"/>
            <loggableItem isEnabled="false" name="Prob-1.0" id="Prob-1.0"/>
            <loggableItem isEnabled="false" name="Prob-2.0" id="Prob-2.0"/>
            <loggableItem isEnabled="false" name="Confidence" id="Confidence"/>
          </loggableItemGroup>
          <loggableItemGroup name="Scoring Engine Properties" id="ENGINE_PROPERTY">

```

```

<loggableItem isEnabled="false" name="Model Path" id="MODEL_PATH"/>
<loggableItem isEnabled="false" name="Scoring Configuration Name"
  id="CONFIGURATION_NAME"/>
<loggableItem isEnabled="false" name="Model Version Label" id="MODEL_LABEL"/>
<loggableItem isEnabled="false" name="Model MIME type" id="MODEL_MIME_TYPE"/>
<loggableItem isEnabled="false" name="Model Version Marker"
  id="MODEL_MARKER"/>
<loggableItem isEnabled="false" name="Scoring Configuration Serial"
  id="CONFIGURATION_SERIAL"/>
<loggableItem isEnabled="false" name="Model ID" id="MODEL_ID"/>
</loggableItemGroup>
<loggableItemGroup name="Score Provider Properties" id="PROVIDER_PROPERTY"/>
</logSettings>
<inputAttributeOrder>
  <attributes>age</attributes>
  <attributes>educ</attributes>
  <attributes>prestg80</attributes>
  <attributes>speduc</attributes>
</inputAttributeOrder>
<cacheSize value="1" min="1" max="100"/>
<batchEnabledFlag>false</batchEnabledFlag>
<configurationItem>
  <xml>&lt;?xml version="1.0" encoding="UTF-8" ?&gt; &lt;node
    name="Model Specific" objectType="Model
    Specific" type="boolean
    objectType="PredictionOnly" name="A flag for
    prediction only" required="true"
    value="false" /&gt; &lt;choice
    objectType="MissingValuePolicy" name="Missing
    value handling policy for this model"
    required="true" /&gt; &lt;item
    value="Substitute: Different model has different substitute
    policy" /&gt; &lt;item value="Score as SYSMIS: Score
    result would be SYSMIS" /&gt; &lt;item value="Throw
    exception: A SCMissingPredictorException is
    thrown" /&gt; &lt;value>Score as SYSMIS: Score result
    would be
    SYSMIS</value> &lt;/choice> &lt;/node></xml>
  </configurationItem>
</configurationDetails>
</updateConfigurationDetails>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP response example

The server responds to a `updateConfigurationDetails` 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>
    <d:updateConfigurationDetailsResponse xmlns:a="http://xml.spss.com/data"
      xmlns:b="http://xml.spss.com/pev/types" xmlns:c="http://xml.spss.com/scoring-v2"
      xmlns:d="http://xml.spss.com/scoring-v2/remote"
      xmlns:e="http://xml.spss.com/scoring/exception">
      <c:configurationDetails cfgSerial="21c881e0-ddc4-11dd-aca9-dcb0b853b9a7" id="nb-config">
        <c:modelReference label="LATEST" resourcePath="/PMML/naivebayes.xml"
          id="0a010a07d3dc251c000011eac84bfd18771"/>
        <c:modelInputMetadata isReturnEnabled="false">
          <c:modelInputMetadataField name="age" type="double"
            description="Age of Respondent" isReturned="false"/>
          <c:modelInputMetadataField name="educ" type="double"
            description="Highest Year of School Completed" isReturned="false"/>
          <c:modelInputMetadataField name="prestg80" type="double"
            description="R's Occupational Prestige Score (1980)" isReturned="false"/>
          <c:modelInputMetadataField name="speduc" type="double"
            description="Highest Year School Completed, Spouse" isReturned="false"/>
        </c:modelInputMetadata>
        <c:logSettings isEnabled="false" logDestination="queue/PASWLog">
          <c:loggableItemGroup name="Context Data" id="INPUT_CONTEXT"/>
          <c:loggableItemGroup name="Model Inputs" id="INPUT_MODEL">
            <c:loggableItem isEnabled="false" name="age" id="age"/>
            <c:loggableItem isEnabled="false" name="educ" id="educ"/>
            <c:loggableItem isEnabled="false" name="prestg80" id="prestg80"/>
            <c:loggableItem isEnabled="false" name="speduc" id="speduc"/>
          </c:loggableItemGroup>
          <c:loggableItemGroup name="Model Outputs" id="OUTPUT">
            <c:loggableItem isEnabled="false" name="Prediction" id="Prediction"/>
            <c:loggableItem isEnabled="false" name="Prob" id="Prob"/>
            <c:loggableItem isEnabled="false" name="Prob-1.0" id="Prob-1.0"/>
            <c:loggableItem isEnabled="false" name="Prob-2.0" id="Prob-2.0"/>
            <c:loggableItem isEnabled="false" name="Confidence" id="Confidence"/>
          </c:loggableItemGroup>
          <c:loggableItemGroup name="Scoring Engine Properties" id="ENGINE_PROPERTY">
            <c:loggableItem isEnabled="false" name="Model Path" id="MODEL_PATH"/>
            <c:loggableItem isEnabled="false" name="Scoring Configuration Name"
              id="CONFIGURATION_NAME"/>
            <c:loggableItem isEnabled="false" name="Model Version Label" id="MODEL_LABEL"/>
            <c:loggableItem isEnabled="false" name="Model MIME type" id="MODEL_MIME_TYPE"/>
            <c:loggableItem isEnabled="false" name="Model Version Marker"
              id="MODEL_MARKER"/>
            <c:loggableItem isEnabled="false" name="Scoring Configuration Serial"
              id="CONFIGURATION_SERIAL"/>
            <c:loggableItem isEnabled="false" name="Model ID" id="MODEL_ID"/>
          </c:loggableItemGroup>
          <c:loggableItemGroup name="Score Provider Properties" id="PROVIDER_PROPERTY"/>
        </c:logSettings>
      </c:configurationDetails>
    </d:updateConfigurationDetailsResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

```

    <c:attributes>age</c:attributes>
    <c:attributes>educ</c:attributes>
    <c:attributes>prestg80</c:attributes>
    <c:attributes>speduc</c:attributes>
  </c:inputAttributeOrder>
  <c:cacheSize value="1" min="1" max="100"/>
  <c:batchEnabledFlag>false</c:batchEnabledFlag>
  <c:configurationItem>
    <c:xml>&lt;?xml version="1.0"
      encoding="UTF-8" ?&gt; &lt;node
      name="Model Specific" objectType="Model
      Specific" &gt;&lt;boolean
      objectType="PredictionOnly" name="A flag for
      prediction only" required="true"
      value="false" /&gt;&lt;choice
      objectType="MissingValuePolicy" name="Missing
      value handling policy for this model"
      required="true" &gt;&lt;item
      value="Substitute: Different model has different substitute
      policy" /&gt;&lt;item value="Score as SYSMIS: Score
      result would be SYSMIS" /&gt;&lt;item value="Throw
      exception: A SCMissingPredictorException is
      thrown" /&gt;&lt;value &gt;Score as SYSMIS: Score result
      would be
      SYSMIS&lt;/value &gt;&lt;/choice &gt;&lt;/node &gt;</c:xml>
  </c:configurationItem>
</c:configurationDetails>
</d:updateConfigurationDetailsResponse>
</soapenv:Body>
</soapenv:Envelope>

```

Scoring Service logging

IBM® SPSS® Collaboration and Deployment Services provides the facilities for logging the operations of the Scoring Service which include database objects for storing the information as well as programmable J2EE objects that allow customization of the Scoring Service logging.

Database objects

The following database objects are used to store the Scoring Service log information:

- Request log table
- Database views
- XML schema

Scoring Service logging is supported on the following DBMSs that can be used for the IBM® SPSS® Collaboration and Deployment Services Repository:

- DB2 (Windows and UNIX)
- MS SQL Server
- Oracle

DB2 on iSeries cannot be used for Scoring Service logging.

Request log table

By default, the scoring service request information is stored in the SPSSSCORE_LOG table. Each row in the table corresponds to a scoring service request.

Scoring log table (SPSSSCORE_LOG)

SERIAL. The unique identifier of the scoring service request.

STAMP. The date and time of the scoring service request.

INFO. Additional information about the scoring request in XML format. The information is generated according to the XML schema registered with the database. For more information, see the topic [XML schema](#) on p. 61. The same information is available in relational format from the scoring log view.

Clean-up and maintenance

Over time, as scoring service requests are logged, the SPSSSCORE_LOG can become quite large and it may be necessary to delete records from this table. For example, the administrator may to purge old records prior to January 1, 2009 by running the following SQL statement:

```
DELETE FROM spssscore_log WHERE STAMP < '2009-01-01'
```

Database views

The following scoring views are created in the database by default when the repository is installed. They present the information stored as XML in the INFO column of SPSSSCORE_LOG table in relational format. Use database client application tools to obtain additional information about the properties of the views or run SQL queries.

Scoring request (SPSSSCORE_V_LOG_HEADER)

This view contains a row for every scoring request row in the SPSSSCORE_LOG table.

SERIAL. The unique identifier of the scoring request.

ADDRESS. The IP address for the machine initiating the scoring request. Note that in certain cases it may be the address of the server rather than the client, for example, the address of the cluster load balancer or proxy server.

HOSTNAME. The name of the machine initiating the scoring request. If the servlet container running the scoring service on this machine does not allow Domain Name System reverse lookups, the value corresponds to the IP address of the machine. If no host name can be determined, a null value is used. In cases when hostname lookup takes too long, it may be possible to improve scoring service performance by configuring the system not to look up the hostname using the corresponding configuration option in browser-based IBM® SPSS® Collaboration and Deployment Services Deployment Manager.

PRINCIPAL. The user name associated with the scoring request. If this value is not included in the request, no information is logged.

STAMP. This column contains the timestamp of when the scoring service logged the request.

MODEL_OBJECT_ID. The repository identifier of the object that was configured with the scoring service. For example, if a IBM® SPSS® Modeler stream was configured for scoring, this is the repository identifier of the stream.

MODEL_VERSION_MARKER. The identifier of the specific version of the repository object that was configured for scoring.

CONFIGURATION_NAME The name of the scoring service configuration entry. The name is assigned when a model is configured for scoring.

Scoring request input (SPSSSCORE_V_LOG_INPUT)

The view contains the information about the model inputs that were used to produce the score. There may be multiple rows in SPSSSCORE_V_LOG_INPUT for each row in SPSSSCORE_LOG table and SPSSSCORE_V_LOG_HEADER view. Each row in the SPSSSCORE_V_LOG_HEADER represents a single input value.

SERIAL. The unique identifier of the scoring request row.

INPUT_TABLE. If the input source is the IBM® SPSS® Collaboration and Deployment Services Enterprise View, this is the Enterprise View table name.

INPUT_NAME. The name of an input field. If the input source is the Enterprise View, this is the Enterprise View column name.

INPUT_VALUE. Input value.

INPUT_TYPE. Input data type. The following data types are allowed:

- date
- daytime
- decimal
- double
- float
- integer
- long
- string
- timestamp

Scoring request context data (SPSSSCORE_V_LOG_CONTEXT_INPUT)

This view contains the information about the data that was passed to the scoring service and used as a Context Data Source for the Enterprise View Data Provider Definition - Real Time. There may be multiple rows in SPSSSCORE_V_LOG_CONTEXT_INPUT view for each row in SPSSSCORE_V_LOG_HEADER view.

SERIAL. The unique identifier of the scoring request row.

CONTEXT_TABLE. The name of the table used in the Context Data Source.

CONTEXT_ROW. The row number of the context data row starting at 1.

CONTEXT_NAME. The name of an input field corresponding to the name of the column in the Context Data Source.

CONTEXT_VALUE. Input value.

Scoring request DPD output (SPSSSCORE_V_LOG_DPD_OUTPUT)

This view contains the information about the data that was passed to the scoring service from a Data Provider Definition - Real Time. If a Data Provider Definition - Real Time is not being used, these entries are absent.

SERIAL. The unique identifier of the scoring request row.

DO_TABLE. The name of the table used in the Data Provider Definition - Real Time.

DO_ROW. The row number of the Data Provider Definition - Real Time data row starting at 1.

DO_NAME. The name of an input field corresponding to the name of the column in the Data Provider Definition - Real Time.

DO_VALUE. Input value.

Scoring request input (SPSSSCORE_V_LOG_REQUEST_INPUT)

This view contains the information about the data used as input for the scoring service request.

SERIAL. The unique identifier of the scoring request row.

RI_TABLE. The name of the table used in the request.

RI_ROW. The row number of the request data row starting at 1.

RI_NAME. The name of an input field corresponding to the name of the column in the request.

RI_VALUE. Input value.

Scoring request properties (SPSSSCORE_V_LOG_REQUEST_PROP)

This view contains the information about the properties associated with an input table.

SERIAL. The unique identifier of the scoring request row.

RI_TABLE. The name of the table used in the request.

RI_PROP_NAME. The name of the property.

RI_PROP_VALUE. The value for the property.

Scoring request output (SPSSSCORE_V_LOG_OUTPUT)

The SPSSSCORE_V_LOG_OUTPUT view is used to log the outputs of the scoring service. There may be multiple rows in SPSSSCORE_V_LOG_OUTPUT view for each row in SPSSSCORE_V_LOG_HEADER view. The scoring service has the ability to provide multiple outputs. Each output can consist of multiple values. For example, the scoring service may provide

two recommendations (two outputs). Each of these recommendation will be assigned a unique row number starting at 1. For each recommendation, there may be multiple output values.

SERIAL. The unique identifier of the scoring request row.

OUTPUT_ROW. The row number of context data row starting at 1.

OUTPUT_NAME. The output field name (attribute name) corresponding to the name of the column in the Context Data Source.

OUTPUT_VALUE. Output value.

Scoring request metrics (SPSSSCORE_V_LOG_METRIC)

The SPSSSCORE_V_LOG_METRIC view is used to log the output metrics of the scoring service, for example, the time to process the scoring request. There may be multiple rows in SPSSSCORE_V_LOG_METRIC view for each row in SPSSSCORE_V_LOG_HEADER view.

SERIAL. The unique identifier of the scoring request row.

METRIC_NAME. The name of an metric field.

METRIC_VALUE. Metric value.

Scoring request properties (SPSSSCORE_V_LOG_PROPERTY)

The SPSSSCORE_V_LOG_PROPERTY view is used to log the properties used in processing the request. There may be multiple rows in the SPSSSCORE_V_LOG_PROPERTY view for each row in the SPSSSCORE_V_LOG_HEADER view. The properties that can be logged depend on the selected score provider.

SERIAL. The unique identifier of the scoring request row.

METRIC_NAME. The name of a property.

OUTPUT_VALUE. Property value.

XML schema

The following XML schema is registered with the database and used for the INFO column of the SPSSSCORE_LOG table. This schema is required for MS SQL Server and Oracle. It is not required on DB2.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="http://xml.spss.com/scoring/logging"
  version="2.0"
```

```

jaxb:version="2.0"
xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
xmlns:spss_ss_logging="http://xml.spss.com/scoring/logging"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- ***** -->
<!-- SIMPLE TYPES -->
<!-- ***** -->
<xs:simpleType name="pevDataType">
  <xs:annotation>
    <xs:documentation>The type of this column. This maps to the same types defined by
      the DD EventServer. We will map these types to the SQL types using the same
      mapping that the DD Event Server uses.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="boolean"/>
    <!-- <xs:enumeration value="character"/></xs:enumeration> not needed, as string
      should be sufficient for mapping to SQL -->
    <xs:enumeration value="date"/>
    <xs:enumeration value="daytime"/>
    <xs:enumeration value="decimal"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="float"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="long"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="timestamp"/>
  </xs:restriction>
</xs:simpleType>

<xs:attributeGroup name="nillableValueAttributeGroup">
  <xs:attribute name="value" type="xs:string" use="optional">
    <xs:annotation>
      <xs:documentation>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'.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:attributeGroup>

<!-- ***** -->
<!-- COMPLEX TYPES -->
<!-- ***** -->
<xs:complexType name="modelInputValue">
  <xs:annotation>
    <xs:documentation>This element is optionally returned as part of the scoreResult
      element. If the configuration is programmed to return the model input fields
      (see spss_ss:modelInputMetadataField), then this element contains the value that
      was used to produce the score. The value might be null.</xs:documentation>
  </xs:annotation>

```

```

</xs:annotation>
<xs:attribute name="name" type="xs:string" use="required">
  <xs:annotation>
    <xs:documentation>The name of the input item.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="type" type="spss_ss_logging:pevDataType" use="required">
  <xs:annotation>
    <xs:documentation>The data type of the input item.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attributeGroup ref="spss_ss_logging:nilableValueAttributeGroup"/>
</xs:complexType>

<xs:complexType name="inputTable">
  <xs:annotation>
    <xs:documentation>One table of input values, may contain zero or more
      rows.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="columns" type="spss_ss_logging:inputColumn" minOccurs="1"
      maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>An ordered list of column names</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="rowValues" type="spss_ss_logging:rowValues" minOccurs="0"
      maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>A row of values, value order must match defined column
          order.</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="sourceTable" type="xs:string" use="required">
    <xs:annotation>
      <xs:documentation>This attribute holds the name of the source table as defined
        in the model.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="inputColumn">
  <xs:annotation>
    <xs:documentation>Describes a column in the designated input table. If the
      configuration is programmed to return the model input fields (see
      spss_ss:modelInputMetadataField), then this element contains the value that
      was used to produce the score. The value might be null.</xs:documentation>
  </xs:annotation>
  <xs:attribute name="name" type="xs:string" use="required">
    <xs:annotation>
      <xs:documentation>The name of the input item.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>

```

```

    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="type" type="spss_ss_logging:pevDataType" use="required">
    <xs:annotation>
      <xs:documentation>The data type of the input item.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="inputTableWithProperties" >
  <xs:annotation>
    <xs:documentation>Input tables can have loggable properties</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="spss_ss_logging:inputTable">
      <xs:sequence>
        <xs:element name="RequestInputProperties"
          type="spss_ss_logging:requestInputProperties" minOccurs="0" maxOccurs="1">
          <xs:annotation>
            <xs:documentation>Properties that are associated with an input
              table</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="requestInputProperties">
  <xs:annotation>
    <xs:documentation>Properties that are associated with an input table</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="property" type="spss_ss_logging:nameValueType" minOccurs="1"
      maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>Properties that are associated with an input
          table</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="columnNames">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

```

```

<xs:complexType name="rowValues">
  <xs:annotation>
    <xs:documentation>One row of values, note that a value may be null.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="value" type="spss_ss_logging:nilableValue" minOccurs="1"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="output">
  <xs:sequence>
    <xs:element name="columnNames" type="spss_ss_logging:columnNames">
      <xs:annotation>
        <xs:documentation>An ordered list of column names</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="rowValues" type="spss_ss_logging:rowValues" minOccurs="1"
      maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>A row of score data, following the order in the
          columnNames element</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="nameValueType">
  <xs:annotation>
    <xs:documentation>A name value pair.</xs:documentation>
  </xs:annotation>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="context">
  <xs:annotation>
    <xs:documentation>This element contains all the context data inputs to the score
      request.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="columnNames" type="spss_ss_logging:columnNames">
      <xs:annotation>
        <xs:documentation>An ordered list of column names</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="rowValues" type="spss_ss_logging:rowValues" minOccurs="1"
      maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>A row of context data, following the order in the
          columnNames element</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

```

    </xs:element>
  </xs:sequence>
  <xs:attribute name="table" type="xs:string" use="required">
    <xs:annotation>
      <xs:documentation>This attribute describes which context table the input data
        belongs to.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="nillableValue">
  <xs:annotation>
    <xs:documentation>Nillable elements and simpleTypes are not well supported by most
      of the popular frameworks, especially Castor. Instead of a nillable string element,
      use an optional string attribute to represent null values.</xs:documentation>
    </xs:annotation>
  <xs:attributeGroup ref="spss_ss_logging:nillableValueAttributeGroup"/>
</xs:complexType>

<!-- ***** -->
<!-- ELEMENTS -->
<!-- ***** -->
<xs:element name="Info">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Output" type="spss_ss_logging:output" minOccurs="0" maxOccurs="1">
        <xs:annotation>
          <xs:documentation>PA has the ability to generate multiple outputs
            (multiple offers). There will be one OutputRow for each output
            (for each offer). </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Input" type="spss_ss_logging:modelInputValue" minOccurs="0"
        maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>This might be a PEV AV input column or data from some
            other source. THIS ELEMENT IS NOW DEPRECATED. The DpdOutputs and
            RequestInputs elements will be used for all future log entries, and this
            element is kept for backwards compatibility.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="ContextInput" type="spss_ss_logging:context" minOccurs="0"
        maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>Context data that is fed into the DPD (data engine)
            and not necessarily into the model. </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="DpdOutputs" type="spss_ss_logging:inputTable" minOccurs="0"
        maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>Zero to N input tables. The data contained in each

```



```

        table represents the values that have been provided by the data service.
        If a RT-DPD is not used for the model, these entries are not
        present.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="RequestInputs" type="spss_ss_logging:inputTableWithProperties"
minOccurs="0" maxOccurs="unbounded">
    <xs:annotation>
        <xs:documentation>Zero to N score request input tables. The data
        contained in each table represents the inputs provided with the score
        request.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="Metric" type="spss_ss_logging:nameValueType" minOccurs="0"
maxOccurs="unbounded">
    <xs:annotation>
        <xs:documentation>A metric which is defined by either the HSS engine
        or the provider.
        Value is a double represented as a string to account for the
        correct precision and scale.
        An example might be the time to produce the output.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="Property" type="spss_ss_logging:nameValueType" minOccurs="0"
maxOccurs="unbounded">
    <xs:annotation>
        <xs:documentation>A property value. The name is the name of the
        property.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="ModelObjectId" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="ModelVersionMarker" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="ConfigurationName" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="ModelInputTable" type="xs:string" minOccurs="0" maxOccurs="1">
    <xs:annotation>
        <xs:documentation>THIS ELEMENT IS NOW DEPRECATED.</xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

SQL for creating database objects

The following are SQL statements for creating the default database objects for scoring logging.

MS SQL Server (scoring_logging_sql_server.sql)

```

CREATE XML SCHEMA COLLECTION SPSSSCORE_LOG AS '<?xml version="1.0" encoding="UTF-8"?>
<xs:schema

```

```

attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://xml.spss.com/scoring/logging"
version="2.0"
jaxb:version="2.0"
xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
xmlns:spss_ss_logging="http://xml.spss.com/scoring/logging"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- ***** ->
<!--SIMPLE TYPES ->
<!-- ***** ->
<xs:simpleType name="pevDataType">
  <xs:annotation>
    <xs:documentation>The type of this column. This maps to the same types defined by
      the DD EventServer. We will map these types to the SQL types using the same
      mapping that the DD Event Server uses.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="boolean"/>
    <!-- <xs:enumeration value="character"></xs:enumeration> not needed, as string
      should be sufficient for mapping to SQL ->
    <xs:enumeration value="date"/>
    <xs:enumeration value="daytime"/>
    <xs:enumeration value="decimal"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="float"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="long"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="timestamp"/>
  </xs:restriction>
</xs:simpleType>

<xs:attributeGroup name="nillableValueAttributeGroup">
  <xs:attribute name="value" type="xs:string" use="optional">
    <xs:annotation>
      <xs:documentation>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'.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:attributeGroup>

<!-- ***** ->
<!--COMPLEX TYPES ->
<!-- ***** ->
<xs:complexType name="modelInputValue">
  <xs:annotation>

```

```

    <xs:documentation>This element is optionally returned as part of the scoreResult
    element. If the configuration is programmed to return the model input fields
    (see spss_ss:modelInputMetadataField), then this element contains the value that
    was used to produce the score. The value might be null.</xs:documentation>
  </xs:annotation>
  <xs:attribute name="name" type="xs:string" use="required">
    <xs:annotation>
      <xs:documentation>The name of the input item.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="type" type="spss_ss_logging:pevDataType" use="required">
    <xs:annotation>
      <xs:documentation>The data type of the input item.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attributeGroup ref="spss_ss_logging:nilableValueAttributeGroup"/>
</xs:complexType>

<xs:complexType name="inputTable">
  <xs:annotation>
    <xs:documentation>One table of input values, may contain zero or more
    rows.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="columns" type="spss_ss_logging:inputColumn" minOccurs="1"
    maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>An ordered list of column names</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="rowValues" type="spss_ss_logging:rowValues" minOccurs="0"
    maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>A row of values, value order must match defined column
        order.</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="sourceTable" type="xs:string" use="required">
    <xs:annotation>
      <xs:documentation>This attribute holds the name of the source table as defined
      in the model.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="inputColumn">
  <xs:annotation>
    <xs:documentation>Describes a column in the designated input table. If the
    configuration is programmed to return the model input fields (see
    spss_ss:modelInputMetadataField), then this element contains the value that
    was used to produce the score. The value might be null.</xs:documentation>
  </xs:annotation>

```

```

</xs:annotation>
<xs:attribute name="name" type="xs:string" use="required">
  <xs:annotation>
    <xs:documentation>The name of the input item.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="type" type="spss_ss_logging:pevDataType" use="required">
  <xs:annotation>
    <xs:documentation>The data type of the input item.</xs:documentation>
  </xs:annotation>
</xs:attribute>
</xs:complexType>

<xs:complexType name="inputTableWithProperties" >
  <xs:annotation>
    <xs:documentation>Input tables can have loggable properties</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="spss_ss_logging:inputTable">
      <xs:sequence>
        <xs:element name="RequestInputProperties"
          type="spss_ss_logging:requestInputProperties" minOccurs="0" maxOccurs="1">
          <xs:annotation>
            <xs:documentation>Properties that are associated with an input
              table</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="requestInputProperties">
  <xs:annotation>
    <xs:documentation>Properties that are associated with an input table</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="property" type="spss_ss_logging:nameValueType" minOccurs="1"
      maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>Properties that are associated with an input
          table</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="columnNames">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:sequence>

```

```
        <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="rowValues">
    <xs:annotation>
        <xs:documentation>One row of values, note that a value may be null.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="value" type="spss_ss_logging:nilableValue" minOccurs="1"
            maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="output">
    <xs:sequence>
        <xs:element name="columnNames" type="spss_ss_logging:columnNames">
            <xs:annotation>
                <xs:documentation>An ordered list of column names</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="rowValues" type="spss_ss_logging:rowValues" minOccurs="1"
            maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>A row of score data, following the order in the
                    columnNames element</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="nameValueType">
    <xs:annotation>
        <xs:documentation>A name value pair.</xs:documentation>
    </xs:annotation>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="context">
    <xs:annotation>
        <xs:documentation>This element contains all the context data inputs to the score
            request.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="columnNames" type="spss_ss_logging:columnNames">
            <xs:annotation>
                <xs:documentation>An ordered list of column names</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="rowValues" type="spss_ss_logging:rowValues" minOccurs="1"
            maxOccurs="unbounded">
    </xs:sequence>
</xs:complexType>
```

```

        <xs:annotation>
          <xs:documentation>A row of context data, following the order in the
            columnNames element</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="table" type="xs:string" use="required">
      <xs:annotation>
        <xs:documentation>This attribute describes which context table the input data
          belongs to.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>

  <xs:complexType name="nillableValue">
    <xs:annotation>
      <xs:documentation>Nillable elements and simpleTypes are not well supported by most
        of the popular frameworks, especially Castor. Instead of a nillable string element,
        use an optional string attribute to represent null values.</xs:documentation>
    </xs:annotation>
    <xs:attributeGroup ref="spss_ss_logging:nillableValueAttributeGroup"/>
  </xs:complexType>

<!-- ***** ->
<!--ELEMENTS ->
<!-- ***** ->
  <xs:element name="Info">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Output" type="spss_ss_logging:output" minOccurs="0" maxOccurs="1">
          <xs:annotation>
            <xs:documentation>PA has the ability to generate multiple outputs
              (multiple offers). There will be one OutputRow for each output
              (for each offer). </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="Input" type="spss_ss_logging:modelInputValue" minOccurs="0"
          maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>This might be a PEV AV input column or data from some
              other source. THIS ELEMENT IS NOW DEPRECATED. The DpdOutputs and
              RequestInputs elements will be used for all future log entries, and this
              element is kept for backwards compatibility.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="ContextInput" type="spss_ss_logging:context" minOccurs="0"
          maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>Context data that is fed into the DPD (data engine)
              and not necessarily into the model. </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```

<xs:element name="DpdOutputs" type="spss_ss_logging:inputTable" minOccurs="0"
maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>Zero to N input tables. The data contained in each
    table represents the values that have been provided by the data service.
    If a RT-DPD is not used for the model, these entries are not
    present.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="RequestInputs" type="spss_ss_logging:inputTableWithProperties"
minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>Zero to N score request input tables. The data
    contained in each table represents the inputs provided with the score
    request.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="Metric" type="spss_ss_logging:nameValueType" minOccurs="0"
maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>A metric which is defined by either the HSS engine
    or the provider.
    Value is a double represented as a string to account for the
    correct precision and scale.
    An example might be the time to produce the output.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="Property" type="spss_ss_logging:nameValueType" minOccurs="0"
maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>A property value. The name is the name of the
    property.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="ModelObjectId" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="ModelVersionMarker" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="ConfigurationName" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="ModelInputTable" type="xs:string" minOccurs="0" maxOccurs="1">
  <xs:annotation>
    <xs:documentation>THIS ELEMENT IS NOW DEPRECATED.</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema';
GO

```

```

CREATE TABLE SPSSSCORE_LOG (
SERIAL  NVARCHAR(40) NOT NULL,
STAMP   datetime NOT NULL,

```

```

INFO XML (
CONTENT SPSSSCORE_LOG) NULL,
CONSTRAINT PK_SPSSSCORE_LOG PRIMARY KEY (SERIAL )
);
GO

```

```

CREATE VIEW SPSSSCORE_V_LOG_HEADER
AS with xmlnamespaces(default 'http://xml.spss.com/scoring/logging')
SELECT
SERIAL,
STAMP,
INFO.value('data(/Info/ModelObjectId)[1]', 'VARCHAR(40)') AS MODEL_OBJECT_ID,
INFO.value('data(/Info/ModelVersionMarker)[1]', 'VARCHAR(40)') AS MODEL_VERSION_MARKER,
INFO.value('data(/Info/ConfigurationName)[1]', 'VARCHAR(256)') AS CONFIGURATION_NAME,
INFO.value('data(/Info/Property[@name="PRINCIPAL"])[1]/@value', 'VARCHAR(256)') AS PRINCIPAL,
INFO.value('data(/Info/Property[@name="ADDRESS"])[1]/@value', 'VARCHAR(256)') AS ADDRESS,
INFO.value('data(/Info/Property[@name="HOSTNAME"])[1]/@value', 'VARCHAR(256)') AS HOSTNAME
FROM dbo.spssscore_log
;
GO

```

```

CREATE VIEW SPSSSCORE_V_LOG_INPUT
AS with xmlnamespaces(default 'http://xml.spss.com/scoring/logging')
SELECT
SERIAL,
INFO.value('data(/Info/ModelInputTable)[1]', 'VARCHAR(256)') AS INPUT_TABLE,
R.nref.value('@name[1]', 'VARCHAR(256)') AS INPUT_NAME,
R.nref.value('./@value', 'VARCHAR(256)') AS INPUT_VALUE,
R.nref.value('@type[1]', 'VARCHAR(20)') AS INPUT_TYPE
FROM dbo.spssscore_log
CROSS APPLY INFO.nodes('/Info/Input') AS R(nref)
;
GO

```

```

CREATE VIEW SPSSSCORE_V_LOG_OUTPUT
AS with xmlnamespaces(default 'http://xml.spss.com/scoring/logging')
SELECT
SERIAL,
R.nref.query('for $s in ../. return count($s/../rowValues[. << $s])+1').value('.', 'int') as OUTPUT_ROW,
R.nref.query('for $s in . return $s/.././columnNames/name[count($s/../.*[. << $s] +1)').value('.', 'nvarchar(256)')
as OUTPUT_NAME,
R.nref.value('./@value', 'nvarchar(256)') as OUTPUT_VALUE
FROM dbo.spssscore_log
CROSS APPLY INFO.nodes('/Info/Output/rowValues/value') AS R(nref)
;
GO

```

```

CREATE VIEW SPSSSCORE_V_LOG_METRIC
AS with xmlnamespaces(default 'http://xml.spss.com/scoring/logging')

```



```

SELECT
SERIAL,
R.nref.value('@name[1]', 'VARCHAR(256)') AS METRIC_NAME,
R.nref.value('@value', 'VARCHAR(256)') AS METRIC_VALUE
FROM dbo.spssscore_log
CROSS APPLY INFO.nodes('/Info/Metric') AS R(nref)
;
GO

```

```

CREATE VIEW SPSSSCORE_V_LOG_PROPERTY
AS with xmlnspaces(default 'http://xml.spss.com/scoring/logging')
SELECT
SERIAL,
R.nref.value('@name[1]', 'VARCHAR(256)') AS PROPERTY_NAME,
R.nref.value('@value', 'VARCHAR(256)') AS PROPERTY_VALUE
FROM dbo.spssscore_log
CROSS APPLY INFO.nodes('/Info/Property') AS R(nref)
;
GO

```

```

CREATE VIEW SPSSSCORE_V_LOG_CONTEXT_INPUT
AS with xmlnspaces(default 'http://xml.spss.com/scoring/logging')
SELECT
SERIAL,
R.nref.value('string((../)[1]/@table)', 'nvarchar(256)') as CONTEXT_TABLE,
R.nref.query('for $s in ../. return count($s/./rowValues[. << $s])+1').value('.', 'int') as CONTEXT_ROW,
R.nref.query('for $s in . return $s/./columnNames/name[count($s/./[*]. << $s) +1]').value('.', 'nvarchar(256)')
as CONTEXT_NAME,
R.nref.value('./@value', 'nvarchar(256)') as CONTEXT_VALUE
FROM dbo.spssscore_log
CROSS APPLY INFO.nodes('/Info/ContextInput/rowValues/value') AS R(nref)
;
GO

```

```

CREATE VIEW SPSSSCORE_V_LOG_DPD_OUTPUT
AS with xmlnspaces(default 'http://xml.spss.com/scoring/logging')
SELECT
SERIAL,
R.nref.value('string((../)[1]/@sourceTable)', 'nvarchar(256)') as DO_TABLE,
R.nref.query('for $s in ../. return count($s/./rowValues[. &lt;&lt; $s])+1').value('.', 'int') AS DO_ROW,
R.nref.query('data(for $s in . return $s/./columns[count($s/./[*]. &lt;&lt; $s) +1]/@name) ').value('.', 'nvarchar(256)')
as DO_NAME,
R.nref.value('./@value', 'nvarchar(256)') AS DO_VALUE
FROM dbo.spssscore_log
CROSS APPLY INFO.nodes('/Info/DpdOutputs/rowValues/value') AS R(nref)
;
GO

```

```

CREATE VIEW SPSSSCORE_V_LOG_REQUEST_INPUT
AS with xmlnamespaces(default 'http://xml.spss.com/scoring/logging')
SELECT
SERIAL,
R.nref.value('string(..../[1]/@sourceTable)', 'nvarchar(256)') AS RI_TABLE,
R.nref.query('for $s in ../. return count($s/./rowValues[. &lt;&lt; $s])+1' ).value('.', 'int') AS RI_ROW,
R.nref.query('data(for $s in . return $s/././columns[count($s/./[*] [ &lt;&lt; $s] +1)/@name)' ).value('.', 'nvarchar(256)')
AS RI_NAME,
R.nref.value('./@value', 'nvarchar(256)') AS RI_VALUE
FROM dbo.spssscore_log
CROSS APPLY INFO.nodes('/Info/RequestInputs/rowValues/value') AS R(nref)
;
GO

```

```

CREATE VIEW SPSSSCORE_V_LOG_REQUEST_PROP
AS with xmlnamespaces(default 'http://xml.spss.com/scoring/logging')
SELECT
SERIAL,
R.nref.value('string(..../[1]/@sourceTable)', 'nvarchar(256)') AS RI_TABLE,
R.nref.value('./@name', 'nvarchar(256)') AS RI_PROP_NAME,
R.nref.value('./@value', 'nvarchar(256)') AS RI_PROP_VALUE
FROM dbo.spssscore_log
CROSS APPLY INFO.nodes('/Info/RequestInputs/RequestInputProperties/property') AS R(nref)
;
GO

```

Oracle (scoring_logging_ora.sql)

```

declare x VARCHAR(15000) := '<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="http://xml.spss.com/scoring/logging"
  version="2.0"
  jaxb:version="2.0"
  xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
  xmlns:spss_ss_logging="http://xml.spss.com/scoring/logging"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- ***** -->
<!-- SIMPLE TYPES -->
<!-- ***** -->
<xs:simpleType name="pevDataType">
  <xs:annotation>
    <xs:documentation>The type of this column. This maps to the same types defined by
      the DD EventServer. We will map these types to the SQL types using the same
      mapping that the DD Event Server uses.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="boolean"/>
    <!-- <xs:enumeration value="character"></xs:enumeration> not needed, as string

```

should be sufficient for mapping to SQL →

```

    <xs:enumeration value="date"/>
    <xs:enumeration value="daytime"/>
    <xs:enumeration value="decimal"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="float"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="long"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="timestamp"/>
  </xs:restriction>
</xs:simpleType>

<xs:attributeGroup name="nillableValueAttributeGroup">
  <xs:attribute name="value" type="xs:string" use="optional">
    <xs:annotation>
      <xs:documentation>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'.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:attributeGroup>

<!-- ***** -->
<!-- COMPLEX TYPES -->
<!-- ***** -->
<xs:complexType name="modelInputValue">
  <xs:annotation>
    <xs:documentation>This element is optionally returned as part of the scoreResult
      element. If the configuration is programmed to return the model input fields
      (see spss_ss:modelInputMetadataField), then this element contains the value that
      was used to produce the score. The value might be null.</xs:documentation>
  </xs:annotation>
  <xs:attribute name="name" type="xs:string" use="required">
    <xs:annotation>
      <xs:documentation>The name of the input item.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="type" type="spss_ss_logging:pevDataType" use="required">
    <xs:annotation>
      <xs:documentation>The data type of the input item.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attributeGroup ref="spss_ss_logging:nillableValueAttributeGroup"/>
</xs:complexType>

<xs:complexType name="inputTable">
  <xs:annotation>
    <xs:documentation>One table of input values, may contain zero or more

```

```

    rows.</xs:documentation>
</xs:annotation>
<xs:sequence>
  <xs:element name="columns" type="spss_ss_logging:inputColumn" minOccurs="1"
    maxOccurs="unbounded">
    <xs:annotation>
      <xs:documentation>An ordered list of column names</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="rowValues" type="spss_ss_logging:rowValues" minOccurs="0"
    maxOccurs="unbounded">
    <xs:annotation>
      <xs:documentation>A row of values, value order must match defined column
        order.</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:sequence>
<xs:attribute name="sourceTable" type="xs:string" use="required">
  <xs:annotation>
    <xs:documentation>This attribute holds the name of the source table as defined
      in the model.</xs:documentation>
  </xs:annotation>
</xs:attribute>
</xs:complexType>

<xs:complexType name="inputColumn">
  <xs:annotation>
    <xs:documentation>Describes a column in the designated input table. If the
      configuration is programmed to return the model input fields (see
      spss_ss:modelInputMetadataField), then this element contains the value that
      was used to produce the score. The value might be null.</xs:documentation>
  </xs:annotation>
  <xs:attribute name="name" type="xs:string" use="required">
    <xs:annotation>
      <xs:documentation>The name of the input item.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="type" type="spss_ss_logging:pevDataType" use="required">
    <xs:annotation>
      <xs:documentation>The data type of the input item.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="inputTableWithProperties" >
  <xs:annotation>
    <xs:documentation>Input tables can have loggable properties</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="spss_ss_logging:inputTable">
      <xs:sequence>
        <xs:element name="RequestInputProperties"

```

```

    type="spss_ss_logging:requestInputProperties" minOccurs="0" maxOccurs="1">
      <xs:annotation>
        <xs:documentation>Properties that are associated with an input
          table</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="requestInputProperties">
  <xs:annotation>
    <xs:documentation>Properties that are associated with an input table</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="property" type="spss_ss_logging:nameValueType" minOccurs="1"
      maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>Properties that are associated with an input
          table</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="columnNames">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="rowValues">
  <xs:annotation>
    <xs:documentation>One row of values, note that a value may be null.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="value" type="spss_ss_logging:nilableValue" minOccurs="1"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="output">
  <xs:sequence>
    <xs:element name="columnNames" type="spss_ss_logging:columnNames">
      <xs:annotation>
        <xs:documentation>An ordered list of column names</xs:documentation>
      </xs:annotation>
    </xs:element>

```

```

    <xs:element name="rowValues" type="spss_ss_logging:rowValues" minOccurs="1"
    maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>A row of score data, following the order in the
        columnNames element</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="nameValueType">
  <xs:annotation>
    <xs:documentation>A name value pair.</xs:documentation>
  </xs:annotation>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="context">
  <xs:annotation>
    <xs:documentation>This element contains all the context data inputs to the score
    request.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="columnNames" type="spss_ss_logging:columnNames">
      <xs:annotation>
        <xs:documentation>An ordered list of column names</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="rowValues" type="spss_ss_logging:rowValues" minOccurs="1"
    maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>A row of context data, following the order in the
        columnNames element</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="table" type="xs:string" use="required">
    <xs:annotation>
      <xs:documentation>This attribute describes which context table the input data
      belongs to.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="nillableValue">
  <xs:annotation>
    <xs:documentation>Nillable elements and simpleTypes are not well supported by most
    of the popular frameworks, especially Castor. Instead of a nillable string element,
    use an optional string attribute to represent null values.</xs:documentation>
  </xs:annotation>
  <xs:attributeGroup ref="spss_ss_logging:nillableValueAttributeGroup"/>

```

```

</xs:complexType>
<!-- ***** -->
<!-- ELEMENTS -->
<!-- ***** -->
<xs:element name="Info">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Output" type="spss_ss_logging:output" minOccurs="0" maxOccurs="1">
        <xs:annotation>
          <xs:documentation>PA has the ability to generate multiple outputs
            (multiple offers). There will be one OutputRow for each output
            (for each offer). </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Input" type="spss_ss_logging:modelInputValue" minOccurs="0"
        maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>This might be a PEV AV input column or data from some
            other source. THIS ELEMENT IS NOW DEPRECATED. The DpdOutputs and
            RequestInputs elements will be used for all future log entries, and this
            element is kept for backwards compatibility.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="ContextInput" type="spss_ss_logging:context" minOccurs="0"
        maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>Context data that is fed into the DPD (data engine)
            and not necessarily into the model. </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="DpdOutputs" type="spss_ss_logging:inputTable" minOccurs="0"
        maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>Zero to N input tables. The data contained in each
            table represents the values that have been provided by the data service.
            If a RT-DPD is not used for the model, these entries are not
            present.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="RequestInputs" type="spss_ss_logging:inputTableWithProperties"
        minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>Zero to N score request input tables. The data
            contained in each table represents the inputs provided with the score
            request.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Metric" type="spss_ss_logging:nameValueType" minOccurs="0"
        maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>A metric which is defined by either the HSS engine

```

or the provider.

Value is a double represented as a string to account for the correct precision and scale.

An example might be the time to produce the output.</xs:documentation>

```
</xs:annotation>
</xs:element>
<xs:element name="Property" type="spss_ss_logging:nameValueType" minOccurs="0"
maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>A property value. The name is the name of the
property.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="ModelObjectId" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="ModelVersionMarker" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="ConfigurationName" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="ModelInputTable" type="xs:string" minOccurs="0" maxOccurs="1">
  <xs:annotation>
    <xs:documentation>THIS ELEMENT IS NOW DEPRECATED.</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>;
```

```
begin
```

```
dbms_xmlschema.registerSchema('http://xml.spss.com/scoring/logging', xmltype.createxml(x) );
```

```
CREATE TABLE SPSSSCORE_LOG (
SERIAL VARCHAR(40) NOT NULL, STAMP DATE NOT NULL, INFO XMLTYPE NULL,
CONSTRAINT PK_SPSSSCORE_LOG PRIMARY KEY (SERIAL )
)XMLTYPE INFO STORE AS OBJECT RELATIONALXMLSCHEMA "http://xml.spss.com/scoring/logging" ELEMENT "Info";
```

```
CREATE VIEW SPSSSCORE_V_LOG_HEADER
```

```
AS
```

```
SELECT SERIAL, STAMP,
```

```
CAST(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
```

```
/s:Info/s:ModelObjectId/text()' passing INFO returning content) AS VARCHAR(40)) as MODEL_OBJECT_ID,
```

```
CAST(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
```

```
/s:Info/s:ModelVersionMarker/text()' passing INFO returning content) AS VARCHAR(40)) as MODEL_VERSION_MARKER,
```

```
CAST(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
```

```
/s:Info/s:ConfigurationName/text()' passing INFO returning content) AS VARCHAR(256)) as CONFIGURATION_NAME,
```

```
CAST(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
```

```
/s:Info/s:Property[@name="PRINCIPAL"]/@value' passing INFO returning content) AS VARCHAR(256)) as PRINCIPAL,
```

```
CAST(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
```

```
/s:Info/s:Property[@name="ADDRESS"]/@value' passing INFO returning content) AS VARCHAR(256)) as ADDRESS,
```

```
CAST(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
```

```
/s:Info/s:Property[@name="HOSTNAME"]/@value' passing INFO returning content) AS VARCHAR(256)) as HOSTNAME
```

```
FROM spssscore_log
```

```
;
```



```

CREATE VIEW SPSSSCORE_V_LOG_INPUT
AS
select SERIAL,
CAST(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
/s:Info/s:ModelInputTable/text() passing INFO returning content) AS VARCHAR(256)) as INPUT_TABLE,
xtab.INPUT_NAME,
xtab.INPUT_VALUE,
xtab.INPUT_TYPE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
/s:Info/s:Input' PASSING Info COLUMNS INPUT_NAME VARCHAR(256) PATH './@name',
INPUT_VALUE VARCHAR(256) PATH './@value', INPUT_TYPE VARCHAR(20) PATH './@type') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_OUTPUT
AS
select SERIAL,
xtab.OUTPUT_ROW,
xtab.OUTPUT_NAME,
xtab.OUTPUT_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
for $i in /s:Info/s:Output/s:rowValues/s:value return
<row>
<num>{count($i/././s:rowValues[. << $i/./]) + 1} </num>
<col>{$i/././s:columnNames/s:name[count($i/./[*][. << $i]) + 1]}</col>
{if (($i/./)[1]/@value) then <val>{ string(($i/./)[1]/@value)} </val> else <dummy></dummy>}
</row>'
PASSING Info COLUMNS OUTPUT_NAME VARCHAR(256) PATH '/row/col', OUTPUT_VALUE VARCHAR(256)
PATH '/row/val',OUTPUT_ROW INT PATH '/row/num') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_METRIC
AS
select SERIAL,
xtab.METRIC_NAME,
xtab.METRIC_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
/s:Info/s:Metric' PASSING Info COLUMNS METRIC_NAME VARCHAR(256) PATH './@name',
METRIC_VALUE VARCHAR(256) PATH './@value') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_PROPERTY
AS
select SERIAL,
xtab.PROPERTY_NAME,

```

```

xtab.PROPERTY_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
/s:Info/s:Property' PASSING Info COLUMNS PROPERTY_NAME VARCHAR(256) PATH './@name',
PROPERTY_VALUE VARCHAR(256) PATH './@value') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_CONTEXT_INPUT
AS
select SERIAL,
xtab.CONTEXT_TABLE,
xtab.CONTEXT_ROW,
xtab.CONTEXT_NAME,
xtab.CONTEXT_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
for $i in /s:Info/s:ContextInput/s:rowValues/s:value return
<row>
<table>{string(($i/./..)[1]/@table)} </table>
<num>{count($i/./..:s:rowValues[. << $i/./..]) +1} </num>
<col>{string($i/./..:s:columnNames/s:name[count($i/./..[*]. << $i)] +1)}</col>
{if (($i/./..)[1]/@value) then <val>{ string(($i/./..)[1]/@value)} </val> else <dummy></dummy>}
</row>'
PASSING Info COLUMNS CONTEXT_TABLE VARCHAR(256) PATH '/row/table', CONTEXT_NAME VARCHAR(256)
PATH '/row/col', CONTEXT_VALUE VARCHAR(256) PATH '/row/val',CONTEXT_ROW INT PATH '/row/num') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_DPD_OUTPUT
AS
SELECT SERIAL,
xtab.DO_TABLE,
xtab.DO_ROW,
xtab.DO_NAME,
xtab.DO_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
for $i in /s:Info/s:DpdOutputs/s:rowValues/s:value return
<row>
<table>{string(($i/./..)[1]/@sourceTable)} </table>
<num>{count($i/./..:s:rowValues[. << $i/./..]) +1} </num>
<col>{string($i/./..:s:columns[count($i/./..[*]. << $i)] +1)/@name}</col>
{if (($i/./..)[1]/@value) then <val>{ string(($i/./..)[1]/@value)} </val> else <dummy></dummy>}
</row>'
PASSING Info COLUMNS DO_TABLE VARCHAR(256) PATH '/row/table', DO_NAME VARCHAR(256)
PATH '/row/col', DO_VALUE VARCHAR(256) PATH '/row/val', DO_ROW INT PATH '/row/num') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_REQUEST_INPUT
AS

```

```

SELECT SERIAL,
xtab.RI_TABLE,
xtab.RI_ROW,
xtab.RI_NAME,
xtab.RI_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
for $i in /s:Info/s:RequestInputs/s:rowValues/s:value return
<row>
<table>{string(($i/./..)[1]/@sourceTable)} </table>
<num>{count($i/./..:s:rowValues[. << $i/./..]) +1} </num>
<col>{string($i/./..:s:columns[count($i/./..[*[. << $i]) +1]/@name)}</col>
{if (($i/./..)[1]/@value) then <val>{ string(($i/./..)[1]/@value)} </val> else <dummy></dummy>}
</row>'
PASSING Info COLUMNS RI_TABLE VARCHAR(256) PATH '/row/table', RI_NAME VARCHAR(256)
PATH '/row/col', RI_VALUE VARCHAR(256) PATH '/row/val', RI_ROW INT PATH '/row/num') xtab
;

```

```

CREATE VIEW SPSSSCORE_V_LOG_REQUEST_PROP
AS
SELECT SERIAL,
xtab.RI_TABLE,
xtab.RI_PROP_NAME,
xtab.RI_PROP_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
for $i in /s:Info/s:RequestInputs/s:RequestInputProperties/s:property return
<row>
<table>{string(($i/./..)[1]/@sourceTable)} </table>
{if (($i/./..)[1]/@name) then <nam>{ string(($i/./..)[1]/@name)} </nam> else <dummy></dummy>}
{if (($i/./..)[1]/@value) then <val>{ string(($i/./..)[1]/@value)} </val> else <dummy></dummy>}
</row>'
PASSING Info COLUMNS RI_TABLE VARCHAR(256) PATH '/row/table', RI_PROP_NAME VARCHAR(256)
PATH '/row/nam', RI_PROP_VALUE VARCHAR(256) PATH '/row/val', RI_ROW INT PATH '/row/val') xtab
;

```

DB2 (scoring_logging_db2.sql)

```

CREATE TABLE SPSSSCORE_LOG (
SERIAL VARCHAR(40) NOT NULL, STAMP TIMESTAMP NOT NULL, INFO XML,
CONSTRAINT PK_SPSSSCORE_LOG PRIMARY KEY (SERIAL)
);

CREATE VIEW SPSSSCORE_V_LOG_HEADER
AS
SELECT SERIAL, STAMP,
xmlcast(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$/s:Info/s:ModelObjectId/text()')
passing INFO as "x" ) as varchar(40) ) as MODEL_OBJECT_ID,
xmlcast(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$/s:Info/s:ModelVersionMarker/text()')

```

```

passing INFO as "x" ) as varchar(40) ) as MODEL_VERSION_MARKER,
xmlcast(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$X/s:Info/s:ConfigurationName/text()')
passing INFO as "x" ) as varchar(256) ) as CONFIGURATION_NAME,
xmlcast(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$X/s:Info/s:Property[@name="PRINCIPAL"]/@value'
passing INFO as "x" ) as varchar(256) ) as PRINCIPAL,
xmlcast(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$X/s:Info/s:Property[@name="ADDRESS"]/@value'
passing INFO as "x" ) as varchar(256) ) as ADDRESS,
xmlcast(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$X/s:Info/s:Property[@name="HOSTNAME"]/@value'
passing INFO as "x" ) as varchar(256) ) as HOSTNAME
FROM spssscore_log
;

CREATE VIEW SPSSSCORE_V_LOG_INPUT
AS
select SERIAL,
xmlcast(xmlquery('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$X/s:Info/s:ModelInputTable/text()') passing INFO as "x" ) as VARCHAR(256) ) as INPUT_TABLE,
xtab.INPUT_NAME,
xtab.INPUT_VALUE,
xtab.INPUT_TYPE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$X/s:Info/s:Input' PASSING Info as "x" COLUMNS INPUT_NAME VARCHAR(256)
PATH './@name', INPUT_VALUE VARCHAR(256) PATH './@value', INPUT_TYPE VARCHAR(20) PATH './@type' ) xtab
;

CREATE VIEW SPSSSCORE_V_LOG_OUTPUT
AS
select SERIAL,
xtab.OUTPUT_ROW,
xtab.OUTPUT_NAME,
xtab.OUTPUT_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
for $i in $X/s:Info/s:Output/s:rowValues/s:value return
document {
<row>
<num>{count($i/././s:rowValues[. << $i/././]) +1} </num>
<col>{$i/././s:columnNames/s:name[count($i/./.*[. << $i]) +1]}</col>
{if (($i/./)[1]/@value) then <val>{ string(($i/./)[1]/@value)} </val> else <dummy></dummy>}
</row> }'
PASSING Info as "x" COLUMNS OUTPUT_NAME VARCHAR(256) PATH '/row/col', OUTPUT_VALUE VARCHAR(256)
PATH '/row/val',OUTPUT_ROW INT PATH '/row/num') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_METRIC
AS
select SERIAL,

```

```

xtab.METRIC_NAME,
xtab.METRIC_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$x/s:Info/s:Metric' PASSING Info as "x" COLUMNS METRIC_NAME VARCHAR(256) PATH './@name',
METRIC_VALUE VARCHAR(256) PATH './@value') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_PROPERTY
AS
select SERIAL,
xtab.PROPERTY_NAME,
xtab.PROPERTY_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$x/s:Info/s:Property' PASSING Info as "x" COLUMNS PROPERTY_NAME VARCHAR(256) PATH './@name',
PROPERTY_VALUE VARCHAR(256) PATH './@value') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_CONTEXT_INPUT
AS
select SERIAL,
xtab.CONTEXT_TABLE,
xtab.CONTEXT_ROW,
xtab.CONTEXT_NAME,
xtab.CONTEXT_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
for $i in $x/s:Info/s:ContextInput/s:rowValues/s:value return
document {
<row>
<table>{string(($i/./..)[1]/@table)} </table>
<num>{count($i/./..:s:rowValues[. << $i/./..]) + 1} </num>
<col>{$i/./..:s:columnNames/s:name[count($i/./.*[. << $i]) + 1]}</col>
{if (($i/./..)[1]/@value) then <val>{ string(($i/./..)[1]/@value)} </val> else <dummy></dummy>}
</row>}
PASSING Info as "x" COLUMNS CONTEXT_TABLE VARCHAR(256) PATH '/row/table', CONTEXT_NAME VARCHAR(256)
PATH '/row/col', CONTEXT_VALUE VARCHAR(256) PATH '/row/val',CONTEXT_ROW INT PATH '/row/num') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_DPD_OUTPUT
AS
SELECT SERIAL,
xtab.DO_TABLE,
xtab.DO_ROW,
xtab.DO_NAME,
xtab.DO_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
for $i in $x/s:Info/s:DpdOutputs/s:rowValues/s:value return
document {
<row>

```

```

<table>{string(($i/./..)[1]/@sourceTable)}</table>
<num>{count($i/././s:rowValues[. << $i/./..] +1)}</num>
<col>{string($i/././s:columns[count($i/./.*[. << $i]) +1]/@name)}</col>
{if (($i/./)[1]/@value) then <val>{ string(($i/./)[1]/@value)} </val> else <dummy></dummy>}
</row>}
PASSING Info as "x" COLUMNS DO_TABLE VARCHAR(256) PATH '/row/table', DO_NAME VARCHAR(256)
PATH '/row/col', DO_VALUE VARCHAR(256) PATH '/row/val', DO_ROW INT PATH '/row/num') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_REQUEST_INPUT
AS
SELECT SERIAL,
xtab.RI_TABLE,
xtab.RI_ROW,
xtab.RI_NAME,
xtab.RI_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
for $i in $x/s:Info/s:RequestInputs/s:rowValues/s:value return
document {
<row>
<table>{string(($i/./..)[1]/@sourceTable)}</table>
<num>{count($i/././s:rowValues[. << $i/./..] +1)}</num>
<col>{string($i/././s:columns[count($i/./.*[. << $i]) +1]/@name)}</col>
{if (($i/./)[1]/@value) then <val>{ string(($i/./)[1]/@value)} </val> else <dummy></dummy>}
</row>}
PASSING Info as "x" COLUMNS RI_TABLE VARCHAR(256) PATH '/row/table', RI_NAME VARCHAR(256)
PATH '/row/col', RI_VALUE VARCHAR(256) PATH '/row/val', RI_ROW INT PATH '/row/num') xtab
;

CREATE VIEW SPSSSCORE_V_LOG_REQUEST_PROP
AS
SELECT SERIAL,
xtab.RI_TABLE,
xtab.RI_PROP_NAME,
xtab.RI_PROP_VALUE
FROM spssscore_log,
XMLTable('xquery version "1.0"; declare namespace s="http://xml.spss.com/scoring/logging";
$x/s:Info/s:RequestInputs/s:RequestInputProperties/s:property'
PASSING Info as "x" COLUMNS RI_TABLE VARCHAR(256) PATH './././@sourceTable',
RI_PROP_NAME VARCHAR(256) PATH '././@name', RI_PROP_VALUE VARCHAR(256) PATH '././@value') xtab
;

```

Customizing logging

The Scoring Service uses the Java Messaging Service (JMS) to log information. Messages generated by the service are placed into *PASWLog* queue (JNDI name *queue/PASWLog*) and subsequently retrieved and written to the database by the *ScoreLogMDB* Message Driven Bean (MDB). The MDB is preinstalled with IBM® SPSS® Collaboration and Deployment Services and changes to the default configuration are not allowed. To customize logging, you must deploy

a custom MDB. While *ScoreLogMDB* may be patched by applying IBM SPSS Collaboration and Deployment Services patches, any custom MDB deployments must be patched manually.

The IBM SPSS Collaboration and Deployment Services architecture enables the following customizations of Scoring Service logging:

- JMS settings
- Logging all scoring request to a different table
- Logging requests for an individual scoring configuration to a different table

Custom MDB

Custom MDBs must be deployed in accordance with J2EE EJB specification and their configuration must include the following settings:

- **<ejb-name>** The name of the EJB. Always substitute the name with a new one when creating your MDB. Do not use *ScoreLogMDB*.
- **<ejb-class>** The class name. The property must always be set to *com.spss.logging.ejb.LoggingMDB*. The class can be found in *logging.jar*.
- **<message-selector>** By default, when the Scoring Service queues messages onto the PASWLog, it sets the *LoggingDestination* property to *ScoringService*. The *ScoreLogMDB* only requests messages with this property set.
- **<env-entry-name>** The target database table name to which the MDB writes the records as it retrieves them from the JMS queue.
- **<transaction-type>** Transaction type. *ScoreLogMDB* uses bean-managed transactions on *jdbc/spss/PlatformDS* data source. The data source does not support XA transactions. If you need to use container-managed transactions, you must configure a different data source that is capable of XA transactions.

The following example is the deployment descriptor for *ScoreLogMDB* (*ejb-jar.xml*).

```
<message-driven>
  <ejb-name>ScoreLogMDB</ejb-name>
  <ejb-class>com.spss.logging.ejb.LoggingMDB</ejb-class>
  <message-selector>LoggingDestination = 'ScoringService'</message-selector>
  <transaction-type>Bean</transaction-type>
  <message-driven-destination>
    <destination-type>javax.jms.Queue</destination-type>
  </message-driven-destination>
  <env-entry>
    <description>target table name</description>
    <env-entry-name>target_table</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>SPSSSCORE_LOG</env-entry-value>
  </env-entry>
</message-driven>
```

Data source

The scoring MDB class (*com.spss.logging.ejb.LoggingMDB*) determines what J2EE data source to log to by doing a JNDI lookup on *jdbc/spss/PlatformDS*.

```
(DataSource)initialContext.lookup("jdbc/spss/PlatformDS")
```

The JNDI name *jdbc/spss/PlatformDS* points to the repository database. As a result, *ScoreLogMDB* logs to the SPSSSCORE_LOG table in the repository. When deploying a custom MDB, use the application server configuration facilities to map this JNDI name to a different data source, if desired.

Custom JMS settings

In certain cases it may be necessary to customize the JMS settings of Scoring Service logging, for instance, to improve performance. JMS settings must be modified through the configuration facilities of the application server. For example, when using WebSphere it is possible to customize the maximum number of instances (threads) for the MDB. In general, allowing more instances will result in greater message throughput at the expense of higher system resources use.

Logging all requests to a different table

By default, the scoring requests are logged to SPSSSCORE_LOG in the repository. For more information, see the topic [Request log table](#) on p. 57. To log to a different table:

1. Customize the SQL scripts to create a different table. You must re-use the existing schema. The script must also be modified to use the new target table for creating the database views. For more information, see the topic [SQL for creating database objects](#) on p. 67.
2. Create your own MDB. In the deployment descriptor, specify a different value for the *target_table* element.
3. Deploy your new MDB. If the new target table is in a different database, map the JNDI name *jdbc/spss/PlatformDS* for your deployment.

Logging specific scoring configurations to a different table

To log specific Scoring Service configurations to a table other than the default:

- ▶ Follow the steps in [Logging all requests to a different table](#).
- ▶ When creating the MDB, specify a unique message selector. For example:

```
LoggingDestination = 'MyCustomLogging'
```
- ▶ When configuring the Scoring Service for a particular model, specify *MyCustomLogging* as the Log Destination property. If the model configuration is performed in IBM® SPSS® Collaboration and Deployment Services Deployment Manager, the setting appears on the Advanced Settings page of the configuration wizard.

Scoring Service access using the Java Message Service

The standard mechanism for accessing the Scoring Service is through SOAP messages. The typical approach involves using HTTP as the transport layer for the messages. However, it is possible to access the service using the Java Message Service (JMS) as the transport layer.

Basically, your client application needs to access the Scoring Service using JMS from outside of the application server. The requirements for gaining access to a JMS queue vary across application servers. If you are not familiar with configuring your environment for such access, you should consult the application server documentation. This discussion assists you with configuring your environment, but the information may not correspond to your particular application server version. The application server documentation should serve as the ultimate reference.

Using JMS to access the Scoring Service

The following list is a general overview of the required steps you should take to access the scoring service through JMS. Each of these steps is explained in detail, and sample Java code is provided afterward.

1. Create your SOAP message.
2. Initialize JMS.
3. Open a temporary reply-to queue.
4. Create a consumer for the temporary reply-to queue.
5. Create a JMS message using the SOAP message.
6. Set the reply-to queue on the JMS message. Set the reply-to queue for the outgoing message to the temporary queue created earlier. This will ensure that the scoring service knows where to send the response message.
7. Send the message to a JMS queue.
8. Receive the reply message.

After sending and receiving JMS messages, be sure to close out the various JMS objects. Care should be taken to ensure that all communication is complete before closing these objects.

Creating the SOAP message

Create a SOAP message to send via JMS. For example, a SOAP message for the `getVersion` operation follows:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:rem="http://xml.spss.com/scoring-v2/remote">
  <soapenv:Header/>
  <soapenv:Body>
    <rem:getVersion/>
  </soapenv:Body>
</soapenv:Envelope>
```

Note that the SOAP envelopes you send via JMS do not require a SOAP security header. It is assumed that an application sever administrator will take the necessary measures to secure the JMS queues. Consult your application server documentation for further information.

Initializing JMS

Create a `Hashtable` with the environment properties that are required to access your application server. For information about values for specific application servers, see [Environment configuration](#).

Once the `InitialContext` object has been populated with the `Hashtable`, use it to discover the `Queue` and `QueueConnectionFactory` that are defined in the application server. With the `QueueConnectionFactory`, obtain a `QueueConnection` and start communicating with the application server.

Opening a queue

Use the `QueueConnection` to create a `QueueSession`, which can be used to create a temporary queue. This temporary queue will be used to receive replies from the scoring service.

You can define a dedicated reply-to queue instead of a temporary queue. In that case, you would likely be using a `Message Driven Bean` or a `MessageListener` to receive the reply messages. Due to the asynchronous nature of this alternate approach, the messages that are sent and received must be matched to each other using the `JMSCorrelationID`. When the message is received, be sure to extract this value for later processing. You should also consider matching the `JMSCorrelationID` between outgoing and incoming messages even when using a temporary reply-to queue and receiving messages synchronously.

Creating a consumer

Create a message consumer that will listen to the temporary queue for reply messages by using a `MessageConsumer` object. Alternatively, you could use a `Message Driven Bean` that implements the `MessageListener` interface. The `MessageConsumer` object receives messages synchronously, with calls to the `receive` method blocking further processing until a reply message arrives. It is possible to receive the messages asynchronously using a `MessageListener` or `Message Driven Bean`. See the Java documentation for details.

Creating a JMS message

Create a JMS message that contains the SOAP message by creating a `QueueSession` for sending the message. Use this session object to create a `QueueSender` that points to the destination queue and to create a `BytesMessage` object.

Sending the message

Place the bytes of the outgoing SOAP message into the JMS message and send the message. Be sure to use UTF-8 encoding when converting the string. Once the message has been sent, you can extract the `JMSCorrelationID` from the message. The `JMSCorrelationID` header field is used for linking a sent message with a reply message. Do not try to obtain this value before sending the message because the value may not have been set prior to sending.

Receiving the reply message

To receive the JMS reply, call the `receive` method for the `MessageConsumer`. Assuming you get a reply message, extract the body of the message by casting the `javax.jms.Message` object to a `javax.jms.BytesMessage`. The JMS message is a sequence of bytes that represents the SOAP reply message. These bytes can be reconstituted into a Java String object using UTF-8 encoding.

Java example for JMS

The following Java example assumes the method `getSOAPMessage` returns a string that defines a SOAP message. Once the SOAP message is created, the remaining steps should be followed to send the message and receive a response. To focus on the JMS steps, the example omits error handling. Actual implementations should include code that checks for errors.

```
// Get a SOAP envelope using whatever means you desire.
String soap = getSOAPMessage();
System.out.println("Message to send...\n " + soap);

// Initialize JMS.
// Create a hash table of settings required to access JMS.
// Note that these settings are application server specific.
Hashtable<String, String> hashTable = new Hashtable<String, String>();
hashTable.put(javax.naming.InitialContext.URL_PKG_PREFIXES, "...");
hashTable.put(javax.naming.InitialContext.PROVIDER_URL, "...");
hashTable.put(javax.naming.InitialContext.INITIAL_CONTEXT_FACTORY, "...");

// Create an InitialContext object so you can discover
// various objects such as connection factories and queues.
javax.naming.InitialContext context = new javax.naming.InitialContext(hashTable);

// Attempt to get the connection factory and queue.
javax.jms.Queue scoringQueue = (javax.jms.Queue)context.lookup("queue/PASWScoring");
javax.jms.QueueConnectionFactory factory = (javax.jms.QueueConnectionFactory)context.lookup("ConnectionFactory");

// Create a connection to the queue and start it.
```

```
javax.jms.QueueConnection queueConnection = factory.createQueueConnection();
queueConnection.start();

// Open a temporary reply queue.
javax.jms.QueueSession listenerSession =
    queueConnection.createQueueSession(false, javax.jms.Session.AUTO_ACKNOWLEDGE);
javax.jms.TemporaryQueue temporaryQueue = listenerSession.createTemporaryQueue();

// Listen to the temporary reply queue for messages returned by the scoring service
javax.jms.MessageConsumer consumer = listenerSession.createConsumer(temporaryQueue);

// Create a JMS message using the SOAP message.
// With this session, you can send as many messages as you like.
javax.jms.QueueSession senderSession =
    queueConnection.createQueueSession(false, javax.jms.Session.AUTO_ACKNOWLEDGE);
javax.jms.QueueSender sender = senderSession.createSender(scoringQueue);
javax.jms.BytesMessage message = senderSession.createBytesMessage();

// Set the reply-to queue on the JMS message.
message.setJMSReplyTo(temporaryQueue);

// Send the message to a JMS queue.
// Populate the message with the soap envelope and send it
message.writeBytes(soap.getBytes("UTF-8"));
sender.send(message);

// Receive the reply message.
// NOTE: This method blocks until a message is received.
Message replyJMSMessage = consumer.receive();

// The message format should be a bytes message.
if (replyJMSMessage != null && replyJMSMessage instanceof javax.jms.BytesMessage)
{
    javax.jms.BytesMessage bytesMessage = (javax.jms.BytesMessage) replyJMSMessage;
    byte[] bytes = new byte[(int) bytesMessage.getBodyLength()];
    bytesMessage.readBytes(bytes);
    System.out.println("Reply Message");
    // the reply message
    String replyMessage = new String(bytes, "UTF-8");
    System.out.println(" " + replyMessage);
    // the JMS correlation ID can be used to match a sent message with a response message
    String jmsCorrelationID = replyJMSMessage.getJMSCorrelationID();
    System.out.println(" reply message ID = " + jmsCorrelationID);
}

// After the message is sent, get the message ID.
// You would keep the message ID around somewhere so you can match it to a reply later.
String messageId = message.getJMSMessageID();
System.out.println(" message ID = " + messageId + "\n");
sender.close();

// Cleanup
```

```
listenerSession.close();
queueConnection.stop();
senderSession.close();
queueConnection.close();
```

Environment configuration

Each application server has its own specific requirements for accessing JMS outside of the application server environment. However, in general, configuring your environment to use JMS as the transport protocol for web service calls involves the following steps:

- Use the correct JMS and InitialContext values for your application server in the client code
- Include the correct *.jar* files on the class path. The application server documentation is the authoritative reference for setting your environment to access JMS. Refer to that documentation for the exact jars that are required to be in the class path.

When defining these values, be sure to use the correct host name and port number corresponding to your application server.

IBM® SPSS® Collaboration and Deployment Services defines a queue under the JNDI name *queue/PASWScoring*, and a queue connection factory under the JNDI name *ConnectionFactory*. Use these values in your code when performing JNDI lookups.

Configuring a Websphere environment

To communicate with a WebSphere application server using JMS, your client should use the following values:

```
javax.naming.InitialContext.PROVIDER_URL = iiop://cds.server.example.com:<BOOTSTRAP_ADDRESS>
javax.naming.InitialContext.INITIAL_CONTEXT_FACTORY = com.ibm.websphere.naming.WsnInitialContextFactory
```

Replace *<BOOTSTRAP_ADDRESS>* with the WebSphere *BOOTSTRAP_ADDRESS* port number listed in the WebSphere Integrated Solutions Console (ISC).

The WebSphere documentation is the authoritative reference for setting your environment to access JMS. Refer to that documentation for the exact jars that are required to be in the class path.

The JNDI name *ConnectionFactory* is used to refer to the queue connection factory. In order to enable external access to this factory, you must define a provider endpoint for this factory.

Determining endpoint provider triplet for a service integration bus

From the WebSphere Integrated Solutions Console (ISC), locate the service integration bus port number, *SIB_ENDPOINT_ADDRESS*. The endpoint triplet reference for this service integration bus is in the following format:

```
<HOST>:<SIB-ENDPOINT-ADDRESS-PORT>:BootstrapBasicMessaging
```

The value of `<HOST>` is the host name or IP address for the WebSphere server and `<SIB-ENDPOINT-ADDRESS-PORT>` is the Service Integration Bus port number.

Configuring a JBoss environment

To communicate with a JBoss application server using JMS, your client should use the following values:

```
javax.naming.InitialContext.URL_PKG_PREFIXES = org.jboss.naming:org.jnp.interfaces
javax.naming.InitialContext.PROVIDER_URL = jnp://cds.server.example.com:<JNP_PORT>
javax.naming.InitialContext.INITIAL_CONTEXT_FACTORY = org.jnp.interfaces.NamingContextFactory
```

Replace `<JNP_PORT>` with the JNP port listed in the JBoss Admin Console. On the *Configuration* page, under the *Standard Bindings* section, the JNP port number is listed in the row containing the following service and binding names:

- Service Name: `jboss:service=Naming`
- Binding Name: `Port`

The JBoss documentation is the authoritative reference for setting your environment to access JMS. Refer to that documentation for the exact jars that are required to be in the class path.

Configuring a WebLogic environment

To communicate with a WebLogic application server using JMS, your client should use the following values:

```
javax.naming.InitialContext.PROVIDER_URL = t3://cds.server.example.com:<PORT>
javax.naming.InitialContext.INITIAL_CONTEXT_FACTORY = weblogic.jndi.WLInitialContextFactory
```

Replace `<PORT>` with the TCP port number listed in the WebLogic Administration Console for *Listen Port*.

The WebLogic documentation is the authoritative reference for setting your environment to access JMS. Refer to that documentation for the exact jars that are required to be in the class path.

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. 97.
2. Configure the web service endpoint. For more information, see the topic [Configuring the web service endpoint](#) on p. 99.
3. Programmatically configure the necessary endpoint behaviors. For more information, see the topic [Configuring endpoint behaviors](#) on p. 100.
4. Exercise the web service. For more information, see the topic [Exercising the service](#) on p. 100.

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.



- active
 - running state, 19
- anomaly detection models
 - scoring, 11
- app.config files
 - WCF clients, 99

- Bayesian networks
 - scoring, 11
- bindings
 - in WSDL files, 5
- body elements
 - in SOAP messages, 3
- buildConfigurationDetails operation, 15

- cache hits
 - for scoring configurations, 14
- cache misses
 - for scoring configurations, 14
- changeConfigurationRunningState operation, 19
- class path
 - for JBoss, 96
 - for JMS access, 95
 - for WebLogic, 96
 - for WebSphere, 95
- cluster models
 - scoring, 11
- computation time
 - for scoring configurations, 13
- computation wait time
 - for scoring configurations, 13
- confidences
 - in scoring, 11
- ConfigurationDetails objects, 17, 22, 52
 - setId method, 48
- ConfigurationItem objects, 23
- ConfigurationReference objects, 26
- ConfigurationRunningState objects, 20
- configurations
 - for scoring models, 10
- ConfigurationStatus objects, 27
- Content Repository service
 - WCF clients, 98
- Content Repository URI service
 - WCF clients, 98
- Cox regression models
 - scoring, 11
- creating
 - scoring configurations, 15, 47
- cumulative hazard values
 - in scoring, 11

- data access time
 - for scoring configurations, 13

- data initialization time
 - for scoring configurations, 13
- deleting
 - scoring configurations, 46
- discriminant models
 - scoring, 11

- general linear models
 - scoring, 11
- generalized linear models
 - scoring, 11
- getAttributes method
 - for InputAttributeOrder objects, 23
- getColumnNames method
 - for ScoreResult objects, 38
- getConfigurationDetails operation, 21
- getConfigurationItem method
 - for ConfigurationDetails objects, 23
- getConfigurations operation, 26
- getConfigurationStatus method
 - for ConfigurationReference objects, 27
- getDescription method
 - for MetadataInputField objects, 29
 - for MetadataOutputField objects, 29
 - for ModelInputMetadataField objects, 22
- getId method
 - for ConfigurationReference objects, 26
 - for LoggableItem objects, 22
 - for metrics, 31
- getInputAttributeOrder method
 - for ConfigurationDetails objects, 23
- getIsEnabled method
 - for LoggableItem objects, 22
- getIsRequired method
 - for MetadataInputField objects, 29
- getIsReturned method
 - for ModelInputMetadataField objects, 22
- getLabel method
 - for ModelReference objects, 22, 26
- getLogDestination method
 - for LogSettings objects, 22
- getLoggableItem method
 - for LoggableItemGroup objects, 22
- getLoggableItemGroup method
 - for LogSettings objects, 22
- getLogSettings method
 - for ConfigurationDetails objects, 22
- getMessage method
 - for ConfigurationStatus objects, 27
- getMetadata operation, 28
- getMetadataInputField method
 - for MetadataResult objects, 29
- getMetadataOutputField method
 - for MetadataResult objects, 29
- getMetricItems operation, 31
- getMetricValue operation, 34

- getModelInputMetadata method
 - for ConfigurationDetails objects, 22
 - getModelInputMetadataField method
 - for ModelInputMetadata objects, 22
 - getModelReference method
 - for ConfigurationDetails objects, 22
 - for ConfigurationReference objects, 26
 - getName method
 - for LoggableItem objects, 22
 - for MetadataInputField objects, 29
 - for MetadataOutputField objects, 29
 - for metrics, 31
 - for ModelInputMetadataField objects, 22
 - for ScoreProviderDetails objects, 42
 - getResourcePath method
 - for ModelReference objects, 22, 26
 - getReturnedDPDOutputValue method
 - for ScoreResult objects, 38
 - getReturnedRequestInputValue method
 - for ScoreResult objects, 38
 - getRowValues method
 - for ScoreResult objects, 38
 - getScale method
 - for metrics, 31
 - getScore operation, 37
 - getScoreProviderDetails method
 - for ScoringServiceDetails objects, 42
 - getServiceDetails operation, 42
 - getStatusCode method
 - for ConfigurationStatus objects, 27
 - getSupportedMimeTypes method
 - for ScoreProviderDetails objects, 42
 - getType method
 - for MetadataInputField objects, 29
 - for MetadataOutputField objects, 29
 - for ModelInputMetadataField objects, 22
 - getUnit method
 - for metrics, 31
 - getValue method
 - for metric values, 35
 - getVersion method
 - for ScoreProviderDetails objects, 42
 - for ScoringServiceDetails objects, 42
 - getVersion operation, 44
 - getXML method
 - for ConfigurationItem objects, 23
- header elements
- in SOAP messages, 3
- Holder classes
- in JAX-WS, 6
- HTTP, 2
- HTTPS, 2
- identifiers
- for scoring configurations, 48
- input fields, 29
- for scoring, 28
- Input objects, 38
- InputAttributeOrder objects, 23
- Java Message Service, 91
- Java example, 93
- Java proxies, 6
- JAX-WS, 6, 16, 23, 27, 30, 32, 39, 42, 48, 52
- JBoss
- class path, 96
 - configuration, 96
- JMS, 91
- configuration, 95–96
 - Java example, 93
- Kohonen models
- scoring, 11
- labels
- for models, 15, 22
- latency
- for scoring configurations, 13
- legal notices, 102
- linear regression models
- scoring, 11
- List collections
- in JAX-WS, 6
- log queue
- for scoring configurations, 14
- log serialization
- for scoring configurations, 13
- logistic regression models
- scoring, 11
- LogSettings objects, 22
- MessageBodyMemberAttribute
- for WCF clients, 100
- messages
- in WSDL files, 5
- MetadataInputField objects, 29
- MetadataOutputField objects, 29
- MetadataResult objects, 29
- metrics
- for scoring performance, 12, 31, 34
- MIME types, 42
- for scoring providers, 42
- ModelInputMetadata objects, 22
- ModelReference objects, 15, 22, 26
- naive Bayes models
- scoring, 11
- nearest neighbors
- in scoring, 11

- .NET framework, 97
- .NET proxies, 7
- neural networks
 - scoring, 11
- node identifiers
 - in scoring, 11

- ordinal multinomial regression models
 - scoring, 11
- output fields, 29

- PevServices service
 - WCF clients, 98
- ping operation, 45
- port types
 - in WSDL files, 5
- predicted values, 11
- probabilities
 - in scoring, 11
- Process Management service
 - WCF clients, 98
- protocols
 - in web services, 2
- proxies, 6
 - Java, 6
 - .NET, 7

- removeConfiguration operation, 46
- resource paths
 - for models, 15, 20, 22
- retrieving
 - scoring configurations, 21, 26
- ruleset models
 - scoring, 11
- running state
 - active, 19
 - for scoring configurations, 19
 - suspended, 19

- scale
 - for performance metrics, 14
- score providers, 10
 - identifiers, 42
 - MIME types, 42
- ScoreProviderDetails objects, 42
- ScoreRequest objects, 38
- ScoreResult objects, 38
- scores, 11, 37
- scoring
 - configurations, 10
- scoring configurations, 10
 - creating, 15, 47
 - deleting, 46
 - generating scores, 37
 - identifiers, 26
 - performance metrics, 12, 31, 34
 - retrieving, 21, 26
 - running state, 19
 - status, 19, 26
 - updating, 51
- scoring configuratons
 - input fields, 28
- scoring functions, 11
- scoring service
 - stubs, 8
- Scoring service
 - service endpoint, 8
 - WCF clients, 98
- ScoringServiceDetails objects, 42
- service endpoints
 - Scoring service, 8
- services
 - in WSDL files, 6
- setConfigurationDetails operation, 47
- setId method
 - for ConfigurationDetails objects, 48
 - for ConfigurationRunningState objects, 20
 - for ModelReference objects, 15
 - for ScoreRequest objects, 38
- setInput method
 - for ScoreRequest objects, 38
- setLabel method
 - for ModelReference objects, 15
- setName method
 - for Input objects, 38
- setResourcePath method
 - for ModelReference objects, 15
- setState method
 - for ConfigurationRunningState objects, 20
- setValue method
 - for Input objects, 38
- single sign-on
 - WCF clients, 97
- SOAP, 2–3
- standard deviations
 - in scoring, 11
- status
 - for scoring configurations, 27
- stubs
 - scoring service, 8
- support vector machine models
 - scoring, 11
- suspended
 - running state, 19

- trademarks, 103
- tree models
 - scoring, 11
- types
 - in WSDL files, 4

- updateConfigurationDetails operation, 51

Index

- updating
 - scoring configurations, 51
- uptime
 - for scoring configurations, 14

- Visual Studio, 97

- WCF clients, 97, 100
 - endpoint behaviors, 100
 - endpoint configuration, 99
 - limitations, 97
 - service reference, 97–98
 - single sign-on, 97
- web services
 - introduction to web services, 1
 - protocol stack, 2
 - system architecture, 1
 - what are web services?, 1
- web.config files
 - WCF clients, 99
- WebLogic
 - class path, 96
 - configuration, 96
- WebSphere
 - class path, 95
 - configuration, 95
- Windows Communication Foundation, 97
- 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, 100