IBM SPSS Collaboration and Deployment Services
Version 8 Release 0

RESTful Scoring Service Developer's Guide

IBM

Note

Before using this information and the product it supports, read the information in “Notices” on page 45.

Product Information

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

Contents

Chapter 1. Scoring Service

Scoring Service overview

The Scoring Service allows client applications to employ real-time scores derived form 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 IBM SPSS Modeler streams or in PMML from IBM 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 Deployment Manager.

Accessing the Scoring Service from a client

Access to the Scoring Service is through standard HTTP and HTTPS methods. A client initiates a request to a server; the server processes the requests and returns the appropriate responses.

The request and response are built around the concept of a resource and its data.

- "Specifying a request URL." The base URL for accessing the service is the IP address of an enabled server.
- "Specifying request headers" on page 2. Certain requests may require information in the request header.
- "Specifying a request body" on page 3. For operations that transmit data to the server, specify a request body in JSON format.
- "Receiving response headers" on page 4. Requests return headers as part of the response.
- "Receiving a response body" on page 4. For a `GET` operation, the response body contains the requested information in the format of a JSON object or array of objects.

Specifying a request URL

The base URL consists of a server and port specification appended with the service location.

The base URL for accessing the Scoring Service is the following:

```
http://{server}:{port}/scoring/rest
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository

Send the base URL appended with `/configuration` as a `GET` request to obtain information about the scoring configurations available in your IBM SPSS Collaboration and Deployment Services Repository. Most of the remaining URL requests include the identifier for a particular scoring configuration. The following sample URL requests illustrate typical use.

- `GET http://cdssvr:80/scoring/rest/service`

Gets information about the score providers available for your system. See the topic "Service resource" on page 7 for more information.

- `GET http://cdssvr:80/scoring/rest/configuration`

  Gets information about all available scoring configurations. Use this method to determine the identifier for the configuration you want to use for scoring or for performance analysis.See the topic "Configuration resource" on page 8 for more information.
- `GET http://cdssvr:80/scoring/rest/configuration/KMeans/metadata`

  Gets information about the inputs for the scoring configuration named *KMeans*. This information is useful for constructing the body of score requests.See the topic "Metadata resource" on page 10 for more information.
- `POST http://cdssvr:80/scoring/rest/configuration/KMeans/score`

  Retrieves a score using the scoring configuration named *KMeans* The input data for the score is supplied in the request body. See the topic "Score resource" on page 13 for more information.
- `GET http://cdssvr:80/scoring/rest/configuration/KMeans/metric`

  Gets information about all available performance statistics for the scoring configuration named *KMeans*. See the topic "Metric item resource" on page 19 for more information.
- `GET http://cdssvr:80/scoring/rest/configuration/KMeans/metric/CONFIGURATION_UPTIME`

  Gets the current value for the amount of time the scoring configuration named *KMeans* has been available for scoring. See the topic "Metric value resource" on page 22 for more information.

The URL encoding for your application server must be defined to support the characters used in your URL strings. For example, if a scoring configuration name contains multi-byte characters and the application server URL encoding is defined as ISO-8859-1, the server returns an error indicating that the configuration cannot be found. Setting the URL encoding to UTF-8 eliminates this problem. For information about setting the URL encoding for a specific server, see your application server documentation.

Specifying request headers

Certain requests may require headers.

The following table identifies fields commonly used in request headers.

Table 1. Request header values

Name	Value
Accept	Indicates the payload type to return to the client. Valid values include `application/json`, `text/xml`, and `application/xml`.
Authorization	Authentication string encoded using Base64
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information

Accept

The *Accept* field defines the MIME type of the information contained in the body of the response returned to the client application.

Authorization

The Scoring Service includes a security layer to limit access to authorized users. This layer uses HTTP basic access authentication, requiring the specification of a valid user name and password to be included in any service request.

To add authentication information to a request, include the *Authorization* field in the request header. The value for this field consists of the following components:
- The string *Basic* to indicate basic access authentication is being used
- A space
- The Base64 encoding of the concatenation of the user name, a colon, and the password

For example, the Base64 encoding for a user name of *user* and a password of *pass* is the following:

```
Base64 Encode(user:pass) = dXNlcjpwYXNz
```

The *Authorization* value for this user would be specified as the following string:

```
Basic dXNlcjpwYXNz
```

Note that the WebLogic Application Server enforces HTTP Basic Access Authentication at the domain level using its security realms. As a result, the application server authenticates the user name and password before passing it to the Scoring Service. To avoid this problem, either disable the WebLogic HTTP basic access authentication or define the same users in the WebLogic Admin Console.

Globalization

The Scoring Service can return responses containing globalized information. To specify a language for the response information, include either the `Client-Accept-Language` or the `Accept-Language` fields in the request header. For both fields, specify a language value in accordance with RFC 1766.

The `Client-Accept-Language` field is used by other components of IBM SPSS Collaboration and Deployment Services that interact with the Scoring Service. If your request includes both language fields, this field is evaluated before the `Accept-Language` field, which is the standard HTTP field for language specification.

For example, to create a response in German, include the following header field in your request:

```
Client-Accept-Language: de
```

If your request also includes the following header field:

```
Accept-Language: fr
```

The response will still be in German as the `Client-Accept-Language` field takes precedence. In this case, you would either need to change the value of `Client-Accept-Language` to `fr` or omit this field entirely to receive a response in French.

Specifying a request body

For operations that transmit data to the server, such as `POST`, specify a request body in JavaScript Object Notation, or JSON, format.

For information on JSON formatting, see "Receiving a response body" on page 4. For information about the specific JSON elements and objects used by the Scoring Service, see Chapter 3, "JSON reference for the Scoring Service," on page 25.

This sample POST request retrieves scoring output for a specified set of input values:

```
{
   "id":"config_1",
   "requestInputTable":[
      {
         "name":"Table 1",
         "requestInputRow":[
            {
               "input":[
                  {"name":"Age","value":"1"},
                  {"name":"BP","value":"1"},
                  {"name":"Cholesterol","value":"1"},
                  {"name":"Drug","value":"1"},
                  {"name":"K","value":"1"},
                  {"name":"Na","value":"1"},
                  {"name":"Gender","value":"1"}
               ]
            }
         ]
      }
   ],
   "context":[]
}
```

Receiving response headers

Requests return headers as part of the response.

The following table lists response header values.

Table 2. Response header values for GET requests.

Name	Value
Status Code	*200 OK* or an error code
Server	The name of the server processing the request
X-Powered-By	Identifies the technology underlying the application
Content-Type	The MIME type of the content in the response body. For example, for JSON, the value could be *application/json;charset=ISO-8859-1*.
Content-Length	Length of the response body in bytes
Date	Current date and time
Set-Cookie	Specifies an HTTP session cookie

Receiving a response body

The response body contains the requested information in the format specified by the **`Accept`** field in the request header. For JSON, the body is typically an object or an array of objects.

The following sections provide basic formatting rules for JSON. These rules also apply to the request body.

See http://www.json.org for a complete description of JSON. For information about the specific JSON elements and objects used by the Scoring Service, see Chapter 3, "JSON reference for the Scoring Service," on page 25.

Elements

An element consists of a name-value pair having the following structure:

```
"<name>":<value>
```

- *<name>* is a string that identifies the element
- *<value>* corresponds to the value for the specific element. The value may be a boolean, string, number, object, or array.

String values must be enclosed in quotation marks and may contain backslash escape characters. Numeric values are not enclosed in quotation marks. A boolean value is either true or false, and is not enclosed in quotation marks.

For example, the following *label* element corresponds to a string value of *LATEST*:

```
"label":"LATEST"
```

The following *value* element corresponds to a numeric value of *38.0*:

```
"value":38.0
```

The following *isRequired* element corresponds to a boolean value of *true*:

```
"isRequired":true
```

Objects

An object represents an unordered collection of elements. Objects have the following structure:

```
{<element1>, ... ,<elementN>}
```

- *<element#>* corresponds to a particular element in the object. Individual object elements are separated by commas.

For example, the following object consists of the elements *state*, *modelReference*, *configurationStatus*, and *id*:

```
{
   "state":"ACTIVE",
   "modelReference":{
      "resourcePath":"\/june\/multiple_Input_Double_updated.str",
      "label":"LATEST",
      "id":"09775272cf8ee6e5000001328016b69284dc"
   },
   "configurationStatus":{
      "message":"Started",
      "statusCode":"INFORMATION"
   },
   "id":"AC2"
}
```

The values for the *modelReference* and *configurationStatus* elements are also objects consisting of three and two elements.

Arrays

An array represents an ordered collection of values, elements, objects, or other arrays. Arrays have the following structure:

```
[<entry1>, ... ,<entryN>]
```

- *<entry#>* corresponds to a particular entry in the array. Individual array entries are separated by commas.

For example, the value for the following *metadataOutputField* element is an array containing two objects:

```
"metadataOutputField":[
   {
      "categoricalValues":[],
      "isReturned":true,
      "type":"long",
      "name":"Age",
      "description":"Age"
   },
   {
      "categoricalValues":[],
      "isReturned":true,
      "type":"string",
      "name":"$O-Anomaly",
      "description":"$O-Anomaly"
   }
]
```

Each object in the array consists of five elements having values describing individual output fields. For instance, the first output field has a name of *Age* and is of type *long*. The second output field has a name of *$O-Anomaly* and is of type *string*.

Processing a response body

The content of the JSON information depends on the parser that is used to generate the JSON data. As a result, you might encounter subtle differences in the JSON information that is transmitted by the service, particularly regarding null or missing values.

For example, one parser might return an empty array if an element has no values. In the following JSON, the **`categoricalValues`** element has no values.

```
"metadataInputField":[
   {
      "categoricalValues":[],
      "isRequired":<boolean>,
      "type":"<string>",
      "name":"<string>",
      "description":"<string>"
   }
]
```

A different parser might omit the element entirely if it has no values. In the following JSON, the **`categoricalValues`** element is absent from the object.

```
"metadataInputField":[
   {
      "isRequired":<boolean>,
      "type":"<string>",
      "name":"<string>",
      "description":"<string>"
   }
]
```

Alternatively, a parser might include a `null` value if an element has no values. In the following JSON, the **`categoricalValues`** element has a `null` value.

```
"metadataInputField":[
   {
      "categoricalValues":null,
      "isRequired":<boolean>,
      "type":"<string>",
      "name":"<string>",
      "description":"<string>"
   }
]
```

To ensure your application processes the response correctly, review how your parser behaves. To make your application more robust, consider defensive programming approaches to avoid problems with `null` or missing values in the JSON data.

Chapter 2. API reference for the Scoring Service

Service resource

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

Service GET

Gets information about the available score providers.

Defined in

"Service resource"

Request

```
GET http://{server}:{port}/scoring/rest/service
```

The following variables should be replaced with values for your system.
- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository

Table 3. Request header values

Name	Value
Accept	Indicates the payload type to return to the client. Valid values include `application/json`, `text/xml`, and `application/xml`.
Authorization	Authentication string encoded using Base64
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information

The HTTP request body should be empty.

Response

A successful request returns the following information:
- Status code 200
- Response body in the specified format containing the service representation for the request. A JSON response object has the following general structure:

```
{
   "version":"<string>",
   "scoreProviderDetails":[
      {
         "supportedMimeTypes":["<string>"],
         "version":"<string>",
         "name":"<string>",
         "id":"<string>"
      }
   ]
}
```

See the topic "The scoringServiceDetails object" on page 40 for more information.

An unsuccessful request returns a response code other than 200.

Example

This request retrieves information about which scoring providers are installed.

```
GET http://cdssvr:80/scoring/rest/service
```

This is the JSON response object, indicating that there are two scoring providers installed.

```
{
   "version":"5.0.0.0.155",
   "scoreProviderDetails":[
      {
         "supportedMimeTypes":["application\/x-vnd.spss-clementine-stream"],
         "version":"1.0",
         "name":"Modeler Score Provider",
         "id":"091ed7cda16d295c000001327c563c4593de"
      },
      {
         "supportedMimeTypes":["application\/x-vnd.spss-pmml"],
         "version":"1.0",
         "name":"Score Provider - SmartScore",
         "id":"091ed7cd82a9085b000001327c118593a316"
      }
   ]
}
```

The response indicates that the Scoring Service is able to generate scores for PMML files and IBM SPSS Modeler streams.

Configuration resource

Before a model can be used for scoring, supplemental information must be defined. 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.

Typically, you create scoring configurations using IBM SPSS Deployment Manager. You use the Scoring Service to generate scores or report performance metrics for those configurations.

Configuration GET

Gets information about the available scoring configurations.

Defined in

"Configuration resource" on page 8

Request

```
GET http://{server}:{port}/scoring/rest/configuration
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository

Table 4. Request header values

Name	**Value**
Accept	Indicates the payload type to return to the client. Valid values include `application/json`, `text/xml`, and `application/xml`.
Authorization	Authentication string encoded using Base64
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information

The HTTP request body should be empty.

Response

A successful request returns the following information:

- Status code 200
- Response body in the specified format containing the configuration representation for the request. A JSON response array has the following general structure:

```
[
   {
      "state":"<string>",
      "modelReference":{
         "resourcePath":"<string>",
         "label":"<string>",
         "id":"<string>"
      },
      "configurationStatus":{
         "message":"<string>",
         "statusCode":"<string>"
      },
      "id":"<string>"
   }
]
```

See the topic "The configurationReference object" on page 25 for more information.

An unsuccessful request returns a response code other than 200.

Example

This request retrieves information about the scoring configurations available on the server named `cdssvr`.

```
GET http://cdssvr:80/scoring/rest/configuration
```

This is the JSON response object, indicating that there are two scoring configurations.

```
[
   {
      "state":"ACTIVE",
      "modelReference":{
         "resourcePath":"\/zsi\/multiple_Input_Double_updated.str",
         "label":"LATEST",
         "id":"09775272cf8ee6e5000001328016b69284dc"
      },
      "configurationStatus":{"message":"Started","statusCode":"INFORMATION"},
      "id":"AC2"
   },
   {
      "state":"ACTIVE",
      "modelReference":{
         "resourcePath":"\/zsi\/KMeans.xml",
         "label":"LATEST",
         "id":"091ed68980eebc90000001317470503d82fb"
      },
      "id":"SmartScore(RTDPD)"
   }
]
```

The first configuration has an identifier of *AC2*. The `modelReference` element indicates that the configuration uses the latest version of the IBM SPSS Modeler stream *multiple_Input_Double_updated.str* for score requests. The *configurationStatus* element reports that the configuration has been started.

In contrast, the second configuration, which has an identifier of *SmartScore(RTDPD)*, uses the latest version of the PMML file *KMeans.xml* for score requests.

Metadata resource

Metadata describes the input and output structure used by a scoring model.

Metadata for a model includes the following information:
- Name for each table of input fields
- Name and data type for every input field
- Name and data type for every output field

Use the metadata information to create your scoring requests in a structure required by the scoring model.

Metadata GET

Gets information about the input and output fields used by a specific scoring configuration.

Defined in

"Metadata resource"

Request

```
GET http://{server}:{port}/scoring/rest/configuration/{config_id}/metadata
```

The following variables should be replaced with values for your system.
- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository

- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{config_id}* is the identifier for the scoring configuration. You can obtain the identifiers for the available scoring configurations by using a configuration request. See the topic "Configuration GET" on page 9 for more information.

Table 5. Request header values

Name	**Value**
Accept	Indicates the payload type to return to the client. Valid values include `application/json`, `text/xml`, and `application/xml`.
Authorization	Authentication string encoded using Base64
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information

The HTTP request body should be empty.

Response

A successful request returns the following information:

- Status code 200
- Response body in the specified format containing the metadata representation for the request. A JSON response object has the following general structure:

```
{
   "metadataInputTable":[
      {
         "name":"<string>",
         "id":"<string>",
         "metadataInputField":[
            {
               "categoricalValues":[<string>],
               "isRequired":<boolean>,
               "type":"<string>",
               "name":"<string>",
               "description":"<string>"
            }
         ]
      }
   ],
   "metadataOutputField":[
      {
         "categoricalValues":[<string>],
         "isReturned":<boolean>,
         "type":"<string>",
         "name":"<string>",
         "description":"<string>"
      }
   ],
   "metadataContextTable":[
      {
         "table":"<string>",
         "contextColumn":[
            {
               "categoricalValues":[<string>],
               "type":"<string>",
               "name":"<string>",
               "description":"<string>"
            }
         ]
      }
   ]
}
```

See the topic "The metadataResult object" on page 27 for more information.

An unsuccessful request returns a response code other than 200.

Example

This request retrieves the metadata for the `Kmeans` scoring configuration.

```
GET http://cdssvr:80/scoring/rest/configuration/KMeans/metadata
```

This is the JSON response object.

```
{
   "metadataInputTable":[
      {
         "name":"DRUG1n",
         "id":"id5JAYKBB9IXD",
         "metadataInputField":[
            {
               "categoricalValues":[],"isRequired":true,"type":"long",
               "name":"Age","description":"Age"
            },
            {
               "categoricalValues":[],"isRequired":true,"type":"string",
               "name":"BP","description":"BP"
            },
            {
               "categoricalValues":[],"isRequired":true,"type":"string",
               "name":"Cholesterol","description":"Cholesterol"
            },
            {
               "categoricalValues":[],"isRequired":true,"type":"string",
               "name":"Drug","description":"Drug"
            },
            {
               "categoricalValues":[],'isRequired":true,"type":"double",
               "name":"K","description":"K"
            },
            {
               "categoricalValues":[],"isRequired":true,"type":"double",
               "name":"Na","description":"Na"
            },
            {
               "categoricalValues":[],"isRequired":true,"type":"string",
               "name":"Gender","description":"Gender"
            }
         ]
      }
   ],
   "metadataOutputField":[
      {
         "categoricalValues":[],"isReturned":true,"type":"long",
         "name":"Age","description":"Age"
      },
      {
         "categoricalValues":[],"isReturned":true,"type":"string",
         "name":"Sex","description":"Sex"
      },
      {
         "categoricalValues":[],"isReturned":true,"type":"string",
         "name":"BP","description":"BP"
      },
      {
         "categoricalValues":[],"isReturned":true,"type":"string",
         "name":"Cholesterol","description":"Cholesterol"
      },
      {
         "categoricalValues":[],"isReturned":true,"type":"double",
         "name":"Na","description":"Na"
      },
      {
         "categoricalValues":[],"isReturned":true,"type":"double",
         "name":"K","description":"K"
      },
      {
         "categoricalValues":[],"isReturned":true,"type":"string",
         "name":"Drug","description":"Drug"
      },
      {
         "categoricalValues":[],"isReturned":true,"type":"string",
         "name":"$O-Anomaly","description":"$O-Anomaly"
      },
      {
            "categoricalValues":[],"isReturned":true,"type":"double",
            "name":"$O-AnomalyIndex","description":"$O-AnomalyIndex"
         },
         {
            "categoricalValues":[],"isReturned":true,"type":"long",
            "name":"$O-PeerGroup","description":"$O-PeerGroup"
         },
         {
            "categoricalValues":[],"isReturned":true,"type":"string",
            "name":"$O-Field-1","description":"$O-Field-1"
         },
         {
            "categoricalValues":[],"isReturned":true,"type":"double",
            "name":"$O-FieldImpact-1","description":"$O-FieldImpact-1"
         },
         {
            "categoricalValues":[],"isReturned":true,"type":"string",
            "name":"$O-Field-2","description":"$O-Field-2"
         },
         {
            "categoricalValues":[],"isReturned":true,"type":"double",
            "name":"$O-FieldImpact-2","description":"$O-FieldImpact-2"
         },
         {
            "categoricalValues":[],"isReturned":true,"type":"string",
            "name":"$O-Field-3","description":"$O-Field-3"
         },
         {
            "categoricalValues":[],"isReturned":true,"type":"double",
            "name":"$O-FieldImpact-3","description":"$O-FieldImpact-3"
         }
   ],
   "metadataContextTable":[]
}
```

The `metadataInputTable` element indicates that the configuration expects score requests to include seven input values for a single table. The following list identifies the input fields and their corresponding data types:

- `Age` is of type `long`
- `BP` is of type `string`
- `Cholesterol` is of type `string`
- `Drug` is of type `string`
- `K` is of type `double`
- `Na` is of type `double`
- `Gender` is of type `string`

The `metadataOutputTable` object indicates that the configuration returns score responses that include all input values plus several additional output values containing the scoring results.

The empty array for the `metadataContextTable` object indicates that the configuration does not use any context data.

Score resource

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:

Scoring function	Description
PREDICT	*Returns the predicted value of the target variable.*
STDDEV	*Standard deviation.*

Scoring function	Description
PROBABILITY	*Probability associated with a particular category of a target variable.* Applies only to categorical variables. In the absence of the optional third parameter, *category*, 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.
CONFIDENCE	*A probability measure associated with the predicted value of a categorical target variable.* Applies only to categorical variables.
NODEID	*The terminal node number.* Applies only to tree models.
CUMHAZARD	*Cumulative hazard value.* Applies only to Cox regression models.
NEIGHBOR	*The ID of the kth nearest neighbor.* Applies only to nearest neighbor models. In the absence of the optional third parameter, k, 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.
DISTANCE	*The distance to the kth nearest neighbor.* Applies only to nearest neighbor models. In the absence of the optional third parameter, k, this is the distance to the nearest neighbor. Depending on the model, either Euclidean or City Block distance will be used.

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 (the optional third parameter) for the `PROBABILITY` function.

Table 6. Supported functions by model type.

Model type	**Supported functions**
Tree (categorical target)	`PREDICT`, `PROBABILITY`, `PROBABILITY (category)`, `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 (category)`, `CONFIDENCE`
Conditional Logistic Regression	`PREDICT`
Multinomial Logistic Regression	`PREDICT`, `PROBABILITY`, `PROBABILITY (category)`, `CONFIDENCE`
General Linear Model	`PREDICT`, `STDDEV`
Discriminant	`PREDICT`, `PROBABILITY`, `PROBABILITY (category)`
TwoStep Cluster	`PREDICT`
K-Means Cluster	`PREDICT`
Kohonen	`PREDICT`

Table 6. Supported functions by model type (continued).

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`.

Score POST

Retrieves one or more scores for a specific scoring configuration.

Defined in

"Score resource" on page 13

Request

```
POST http://{server}:{port}/scoring/rest/configuration/{config_id}/score
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{config_id}* is the identifier for the scoring configuration. You can obtain the identifiers for the available scoring configurations by using a configuration request. See the topic "Configuration GET" on page 9 for more information.

Table 7. Request header values.

Name	**Value**
Content-Type	Indicates the payload type of the score request that is sent from the client to the server. Valid values include `application/json`, `text/xml`, and `application/xml`.
Accept	Indicates the payload type to return to the client. Valid values include `application/json`, `text/xml`, and `application/xml`.
Authorization	Authentication string encoded using Base64
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information

The HTTP request body should contain the input data for the score request. A JSON request object has the following general structure:

```
{
   "id":"<string>",
   "requestInputTable":[
      {
         "name":"<string>",
         "requestInputRow":[
            {
               "input":[
                  {
                     "name":"<string>",
                     "value":"<string>"
                  }
                ]
            }
         ]
      }
   ],
   "context":[
      {
         "name":"<string>",
         "columnName":["<string>"],
         "rowValues":[
            {
               "value":[
                  {
                     "value":"<string>"
```

```
                    }
                 ]
              }
           ]
        }
     ]
  }
```

See the topic "The scoreRequest object" on page 32 for more information.

Response

A successful request returns the following information:

- Status code 200
- Response body in the specified format containing the score information for the request. A JSON response object has the following general structure:

```
{
   "id":"<string>",
   "columnNames":{
        "name":["<string>"]
   },
   "rowValues":[
      {
         "reserved":"<string>"
         "value":[
            {
               "value":"<string>"
            }
         ]
      }
   ],
   "returnedRequestInputTable":[
      {
         "id":"<string>",
         "name":"<string>",
         "returnedRequestInputRow":[
            {
               "returnedRequestInputValue":[
                  {
                     "name":"<string>",
                     "type":"<string>",
                     "value":"<string>"
                  }
               ]
            }
         ]
      }
   ],
   "returnedDPDOutputTable":[
      {
         "id":"<string>",
         "name":"<string>",
         "returnedDPDOutputRow":[
            {
               returnedDPDOutputValue[
                  {
                     "name":"<string>",
                     "type":"<string>",
                     "value":"<string>"
                  }
               ]
            }
         ]
      }
   ]
}
```

See the topic "The scoreResult object" on page 37 for more information.

An unsuccessful request returns a response code other than 200.

Example

This request retrieves a score for the *config_1* scoring configuration.

```
GET http://cdssvr:80/scoring/rest/configuration/config_1/metric
```

This is the JSON request body, which includes a single table consisting of one row. This row specifies the values for the seven model inputs for the configuration.

```
{
   "id":"config_1",
   "requestInputTable":[
      {
         "name":"Table 1",
         "requestInputRow":[
            {
               "input":[
                  {"name":"Age","value":"1"},
                  {"name":"BP","value":"1"},
                  {"name":"Cholesterol","value":"1"},
                  {"name":"Drug","value":"1"},
                  {"name":"K","value":"1"},
                  {"name":"Na","value":"1"},
                  {"name":"Gender","value":"1"}
                ]
            }
         ]
      }
   ],
   "context":[]
}
```

This is the JSON response object. The `columnNames` element indicates that the configuration returned scoring function values for *Prediction* and *Confidence*. The `rowValues` element contains a single object that reports the actual scoring values of *2* and *1.34153*.

```
{
   "id":"954d9db3-049f-464c-b218-eadd628cf349",
   "columnNames":[
      "name":["Prediction","Confidence"]
   ],
   "rowValues":[
      {
         "reserved":null,
         "value":[{"value":"2"},{"value":"1.3415297614991626"}]
      }
   ],
   "returnedRequestInputTable":[
      {
         "name":"Table 1",
         "id":"Table 1",
         "returnedRequestInputRow":[
            {
               "returnedRequestInputValue":[
                  {"name":"Age","type":"double","value":"1"},
                  {"name":"BP","type":"string","value":"1"},
                  {"name":"Cholesterol","type":"string","value":"1"},
                  {"name":"Drug","type":"string","value":"1"},
                  {"name":"K","type":"double","value":"1"},
                  {"name":"Na","type":"double","value":"1"},
                  {"name":"Gender","type":"string","value":"1"}
               ]
            }
         ]
      }
   ],
   "returnedDPDOutputTable":[]
}
```

The `returnedRequestInputTable` element contains the input values that resulted in these scores.

Metric item resource

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 log data on to the JMS queue.
- **Minimum Log Queue Time.** Smallest amount of time, measured in milliseconds, to place the XML log data on to the JMS queue.
- **Maximum Log Queue Time.** Largest amount of time, measured in milliseconds, to place the XML log data on to 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.

Metric item GET

Retrieves a list of available scoring metric items for a specific scoring configuration.

Defined in

"Metric item resource" on page 19

Request

```
GET http://{server}:{port}/scoring/rest/configuration/{config_id}/metric
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{config_id}* is the identifier for the scoring configuration. You can obtain the identifiers for the available scoring configurations by using a configuration request. See the topic "Configuration GET" on page 9 for more information.

Table 8. Request header values

Name	**Value**
Accept	Indicates the payload type to return to the client. Valid values include `application/json`, `text/xml`, and `application/xml`.
Authorization	Authentication string encoded using Base64
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information

The HTTP request body should be empty.

Response

A successful request returns the following information:

- Status code 200
- Response body in the specified format containing the metric item information for the request. A JSON response array has the following general structure:

```
[
   {
      "unit":"<string>",
      "scale":<number>,
      "id":"<string>",
      "name":"<string>"
   }
]
```

See the topic "The metricItem object" on page 31 for more information.

An unsuccessful request returns a response code other than 200.

Example

This request retrieves information about all available metrics for the `Kmeans` scoring configuration.

```
GET http://cdssvr:80/scoring/rest/configuration/KMeans/metric
```

This is the JSON response array. Each object in the array corresponds to a separate metric.

```
[
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_COMPUTATION_WAIT_TIME_MINIMUM","name":"Minimum Computation Wait Time"
   },
   {
      "unit":"hits","scale":0,
      "id":"CONFIGURATION_CACHE_HITS","name":"Cache Hits"
   },
   {
      "unit":"scores","scale":0,
      "id":"SERVICE_TOTAL_SCORES","name":"Service Scores"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_DATA_INIT_TIME_AVERAGE","name":"Average Data Initialization Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_LOG_QUEUE_TIME_MINIMUM","name":"Minimum Log Queue Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_DATA_INIT_TIME_MAXIMUM","name":"Maximum Data Initialization Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_RESPONSE_TIME_MINIMUM","name":"Minimum Latency"
   },
   {
      "unit":"seconds","scale":0,
      "id":"SERVICE_UPTIME","name":"Service Uptime"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_COMPUTATION_WAIT_TIME_AVERAGE","name":"Average Computation Wait Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_COMPUTATION_WAIT_TIME_MAXIMUM","name":"Maximum Computation Wait Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_LOG_QUEUE_TIME_AVERAGE","name":"Average Log Queue Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_COMPUTATION_TIME_MINIMUM","name":"Minimum Computation Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_LOG_SERIALIZE_TIME_MINIMUM","name":"Minimum Log Serialization Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_RESPONSE_TIME_AVERAGE","name":"Average Latency"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_LOG_QUEUE_TIME_MAXIMUM","name":"Maximum Log Queue Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_RESPONSE_TIME_MAXIMUM","name":"Maximum Latency"
   },
   {
      "unit":"scores","scale":0,
      "id":"CONFIGURATION_TOTAL_SCORES","name":"Configuration Scores"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_DATA_ACCESS_TIME_MINIMUM","name":"Minimum Data Access Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_COMPUTATION_TIME_AVERAGE","name":"Average Computation Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_LOG_SERIALIZE_TIME_AVERAGE","name":"Average Log Serialization Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_COMPUTATION_TIME_MAXIMUM","name":"Maximum Computation Time"
   },
   {
      "unit":"misses","scale":0,
      "id":"CONFIGURATION_CACHE_MISSES","name":"Cache Misses"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_LOG_SERIALIZE_TIME_MAXIMUM","name":"Maximum Log Serialization Time"
   },
   {
      "unit":"seconds","scale":0,
      "id":"CONFIGURATION_UPTIME","name":"Configuration Uptime"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_DATA_ACCESS_TIME_AVERAGE","name":"Average Data Access Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_DATA_ACCESS_TIME_MAXIMUM","name":"Maximum Data Access Time"
   },
   {
      "unit":"milliseconds","scale":3,
      "id":"CONFIGURATION_DATA_INIT_TIME_MINIMUM","name":"Minimum Data Initialization Time"
   }
]
```

Use the “Metric value GET” method to retrieve the value for a specific metric item.

Metric value resource

A metric value corresponds to actual value for a specific metric item. The identifier for any metric item can be used to retrieve the value for that metric.

Metric value GET

Retrieves the value for a specific scoring configuration metric item.

Defined in

“Metric value resource”

Request

```
GET http://{server}:{port}/scoring/rest/configuration/{config_id}/metric/{name}
```

The following variables should be replaced with values for your system.

- *{server}* corresponds to the server name or IP address for your IBM SPSS Collaboration and Deployment Services Repository
- *{port}* is the port number on which to access the IBM SPSS Collaboration and Deployment Services Repository
- *{config_id}* is the identifier for the scoring configuration. You can obtain the identifiers for the available scoring configurations by using a configuration request. See the topic "Configuration GET" on page 9 for more information.
- *{name}* identifies the metric item whose value is being retrieved. You can obtain a list of available metric items by using a metric item request. See the topic "Metric item GET" on page 20 for more information.

Table 9. Request header values

Name	Value
Accept	Indicates the payload type to return to the client. Valid values include `application/json`, `text/xml`, and `application/xml`.
Authorization	Authentication string encoded using Base64
Client-Accept-Language	Tag defining the language for the response information. This field takes precedence over the Accept-Language field.
Accept-Language	Tag defining the language for the response information

The HTTP request body should be empty.

Response

A successful request returns the following information:

- Status code 200
- Response body in the specified format containing the metric value for the request. A JSON response object has the following general structure:

```
{
   "value":<number>
}
```

See the topic "The metricValue object" on page 32 for more information.

An unsuccessful request returns a response code other than 200.

Example

This request retrieves the value of the `CONFIGURATION_UPTIME` metric for the KMeans scoring configuration.

```
GET http://cdssvr:80/scoring/rest/configuration/KMeans/metric/CONFIGURATION_UPTIME
```

This is the JSON response object, indicating that the scoring configuration has been available for scoring for 38 seconds.

```
{
   "value":38.0
}
```

Chapter 3. JSON reference for the Scoring Service

Describes all JSON objects used in the Scoring Service REST API.

The configurationReference object

Describes a scoring configuration available in the IBM SPSS Collaboration and Deployment Services Repository. This object contains a reference to the model on which the configuration is based, as well as the identifier of the scoring configuration itself.

The "Configuration GET" on page 9 method returns an array of `configurationReference` objects in the response body.

Table 10. Child elements for configurationReference.

Element name	**Value type**	**Description**
id	string	Non-localized identifier for the configuration. Use the id value in URL requests to identify a specific configuration.
state	string	Indicates the current status of the configuration. A state value of *ACTIVE* indicates the configuration is consuming resources and can process incoming scoring requests. In contrast, a state value of *SUSPENDED* indicates the configuration is not consuming resources and cannot process any scoring requests.
modelReference	object	Identifies the model used by a scoring configuration.
configurationStatus	object	Reports the status of a scoring configuration.

Example

The following array includes two `configurationReference` objects. The first has an identifier of *AC2* and uses latest version of the file *multiple_Input_Double_updated.str* for scoring requests. The second has an identifier of *withRTDPD* and uses the version of the file *MultipleSourceNodes.str* that is labeled *PRODUCTION*. The `configurationStatus` object for the *withRTDPD* configuration indicates that the configuration is unable to process any scoring requests due to a problem with the specified data provider.

```
[
   {
      "state":"ACTIVE",
      "modelReference":{
         "resourcePath":"\/june\/multiple_Input_Double_updated.str",
         "label":"LATEST",
         "id":"09775272cf8ee6e5000001328016b69284dc"
      },
      "configurationStatus":{
         "message":"Started",
         "statusCode":"INFORMATION"
      },
      "id":"AC2"
   },
   {
      "state":"ACTIVE",
      "modelReference":{
         "resourcePath":"\/july\/MultipleSourceNodes.str",
         "label":"PRODUCTION",
         "id":"091ed68980eebc90000001317470503d84a5"
      },
      "id":"withRTDPD"
   }
]
```

The modelReference element

Provides all of the information necessary to locate a specific version of a model file in the IBM SPSS Collaboration and Deployment Services Repository.

Table 11. Child elements for modelReference.

Element name	Value type	Description
id	string	Non-localized identifier for the scoring model
label	string	The label corresponding to the version of the model file
resourcePath	string	Full path to the file in the content repository

Example

The following `modelReference` element corresponds to the latest version of the file *multiple_Input_Double_updated.str* in the *june* folder. The path includes the backslash (\) escape character before each slash (/) to optimize client processing.

```
"modelReference":{
   "resourcePath":"\/june\/multiple_Input_Double_updated.str",
   "label":"LATEST",
   "id":"09775272cf8ee6e5000001328016b69284dc"
}
```

The configurationStatus element

Provides information about the current status of a scoring configuration.

Table 12. Child elements for configurationStatus.

Element name	Value type	Description
message	string	Describes the status of the configuration.
statusCode	string	Indicates the type of message. Valid values include the following strings: • `INFORMATION`. The message is informational, describing the current status of the configuration in the system. • `WARNING`. A problem exists that may affect the configuration results. • `ERROR`. A problem exists that prevents the configuration from processing requests.

Example

The following `configurationStatus` element indicates that the configuration has started and is available for processing scoring requests.

```
"configurationStatus":{
   "message":"Started",
   "statusCode":"INFORMATION"
}
```

The metadataResult object

Describes the input and output fields used by a scoring configuration.

The "Metadata GET" on page 10 method returns the `metadataResult` object in the response body.

Table 13. Child elements for metadataResult.

Element name	Value type	Description
metadataInputTable	object array	The input tables for a scoring configuration
metadataContextTable	object array	The context tables for a scoring configuration
metadataOutputField	object array	The output fields for a scoring configuration

Example

The following `metadataResult` object contains one input table, *DRUG*, consisting of three input fields. A scoring request returns four output fields. The empty `metadataContextTable` object indicates that the configuration does not use any context data.

```
{
   "metadataInputTable":[
      {
         "name":"DRUG",
         "id":"id5JAYKBB9IXD",
         "metadataInputField":[
            {
               "categoricalValues":[],"isRequired":true,"type":"long",
               "name":"Age","description":"Age"
            },
            {
               "categoricalValues":[],"isRequired":true,"type":"string",
               "name":"BP","description":"BP"
            },
            {
               "categoricalValues":[],"isRequired":true,"type":"string",
               "name":"Cholesterol","description":"Cholesterol"
            }
         ]
      }
   ],
   "metadataOutputField":[
      {
         "categoricalValues":[],"isReturned":true,"type":"string",
         "name":"BP","description":"BP"
      },
      {
         "categoricalValues":[],"isReturned":true,"type":"string",
         "name":"Cholesterol","description":"Cholesterol"
      },
      {
         "categoricalValues":[],"isReturned":true,"type":"string",
         "name":"$O-Anomaly","description":"$O-Anomaly"
      },
      {
         "categoricalValues":[],"isReturned":true,"type":"double",
         "name":"$O-AnomalyIndex","description":"$O-AnomalyIndex"
      }
   ],
   "metadataContextTable":[]
}
```

The metadataInputTable element

Describes an input table for a scoring configuration.

Table 14. Child elements for metadataInputTable.

Element name	Value type	Description
id	string	Non-localized identifier for the input table
name	string	Name, possibly localized, for the input table
metadataInputField	object array	The input fields used for scoring

Example

The following `metadataInputTable` element describes an input table named *DRUG*. The table consists of the input fields *Age*, *BP*, and *Cholesterol*.

```
"metadataInputTable":[
   {
      "name":"DRUG",
      "id":"id5JAYKBB9IXD",
      "metadataInputField":[
         {
            "categoricalValues":[],"isRequired":true,"type":"long",
            "name":"Age","description":"Age"
         },
         {
            "categoricalValues":[],"isRequired":true,"type":"string",
            "name":"BP","description":"BP"
         },
         {
            "categoricalValues":[],"isRequired":true,"type":"string",
            "name":"Cholesterol","description":"Cholesterol"
         }
      ]
   }
]
```

The metadataInputField element

Describes the scoring input fields for a model. Each object in the array corresponds to an input field.

The objects in this array come directly from the model and all fields are required.

Table 15. Child elements for metadataInputField.

Element name	Value type	Description
description	string	Optional description of the field
isRequired	boolean	Indicates whether or not a value for the field is required for scoring requests
name	string	Name of the field

Table 15. Child elements for metadataInputField (continued).

Element name	Value type	Description
type	string	Data type of the field. • boolean • date • daytime • decimal • double • float • integer • long • string • timestamp
categoricalValues	string array	Set of expected values, if any, for a categorical field

Example

The following `metadataInputField` element consists of an array of objects corresponding to the three fields in the table. The first input field, *Age*, has a type of *long*. The other two fields, *BP* and *Cholesterol*, are both of the *string* type. Values for all three fields are required in scoring requests.

```
"metadataInputField":[
   {
      "categoricalValues":[],'isRequired":true,"type":"long",
      "name":"Age","description":"Age"
   },
   {
      "categoricalValues":[],'isRequired":true,"type":"string",
      "name":"BP","description":"BP"
   },
   {
      "categoricalValues":[],"isRequired":true,"type":"string",
      "name":"Cholesterol","description":"Cholesterol"
   }
]
```

The metadataContextTable element

Describes a context table for a scoring configuration.

Table 16. Child elements for metadataContextTable.

Element name	Value type	Description
table	string	Name for the context table
contextColumn	object array	The context data columns for the context table

The contextColumn element

Describes the context data columns for a context table.

Table 17. Child elements for contextColumn.

Element name	Value type	Description
description	string	Optional description of the field
name	string	Name of the field

Table 17. Child elements for contextColumn (continued).

Element name	Value type	Description
type	string	Data type of the field. • boolean • date • daytime • decimal • double • float • integer • long • string • timestamp
categoricalValues	string array	Set of expected values, if any, for a categorical field

The metadataOutputField element

Describes the output fields for a scoring configuration. Each object in this array corresponds to an output field. The model definition is directly responsible for the objects in this array.

Table 18. Child elements for metadataOutputField.

Element name	Value type	Description
description	string	Optional description of the field
isReturned	boolean	Indicates if the output is returned in a score request. A true value is used if the output is returned in a score request, false otherwise. If this value is not present, the output will be returned.
name	string	Name of the field
type	string	Data type of the field. • boolean • date • daytime • decimal • double • float • integer • long • string • timestamp
categoricalValues	string array	Set of expected values, if any, for a categorical field

Example

The following `metadataOutputField` element consists of an array of objects corresponding to the fields returned as output from a scoring request. The first field *Age*, has a type of *long*. The other field, *$O-Anomaly*, is of the *string* type.

```
"metadataOutputField":[
   {
      "categoricalValues":[],"isReturned":true,"type":"long",
      "name":"Age","description":"Age"
   },
   {
      "categoricalValues":[],"isReturned":true,"type":"string",
      "name":"$O-Anomaly","description":"$O-Anomaly"
   }
]
```

The metricItem object

Identifies all performance measurements available for a scoring configuration. Each object in this array corresponds to a performance metric.

You can retrieve the value for a specific metric by supplying the metric identifier in a request URL. See the topic "Metric value GET" on page 22 for more information.

The "Metric item GET" on page 20 method returns the `metricItem` object in the response body.

Table 19. Child elements for metricItem.

Element name	Value type	Description
id	string	Non-localized identifier for the item
name	string	Name, possibly localized, for the item
scale	number	Identifies the number of digits to the right of the decimal point for the item value
unit	string	Unit of measurement for the item, such as milliseconds

Example

The following `metricItem` object corresponds to the *Service Scores* metric for a configuration. The *unit* property indicates the metric is measured in *scores*. The *scale* property reports that the value of this metric has 0 decimal places. You can retrieve the current value of this metric by including the *id* value, *SERVICE_TOTAL_SCORES*, in a request URL.

```
{
   "unit":"scores",
   "scale":0,
   "id":"SERVICE_TOTAL_SCORES",
   "name":"Service Scores"
}
```

The metricValue object

Reports the current value for a performance metric.

See the topic "The metricItem object" on page 31 for more information.

The "Metric value GET" on page 22 method returns the `metricValue` object in the response body.

Table 20. Child elements for metricValue.

Element name	Value type	Description
value	number	The value for a metric item

Example

The value element of the following `metricValue` object indicates the current value of the requested metric item is 38.

```
{
   "value":38.0
}
```

The scoreRequest object

Represents a request for a score as an object containing the input values for a scoring model.

You can obtain the structure for the input tables by using the "Metadata GET" on page 10 method.

The "Score POST" on page 16 method uses the `scoreResult` object in the request body.

Table 21. Child elements for scoreRequest.

Element name	Value type	Description
id	string	Identifier for the scoring configuration being used to generate scores
requestInputTable	object array	Input tables for the scoring model
context	object array	Context tables containing indirect inputs for the scoring model

Example

The following object corresponds to a scoring request for the *config_1* scoring configuration. The request includes one table named *Table 1* that contains a single input row.

```
{
   "id":"config_1",
   "requestInputTable":[
      {
         "name":"Table 1",
         "requestInputRow":[
            {
               "input":[
                  {"name":"Age","value":"1"},
                  {"name":"BP","value":"1"},
                  {"name":"Cholesterol","value":"1"},
                  {"name":"Drug","value":"1"},
                  {"name":"K","value":"1"},
                  {"name":"Na","value":"1"},
                  {"name":"Gender","value":"1"}
               ]
            }
         ]
      }
   ],
   "context":[]
}
```

The requestInputTable element

Describes the input tables used in a score request. The value for this element is an object array in which each object represents an input table.

Table 22. Child elements for requestInputTable.

Element name	Value type	Description
name	string	Name, possibly localized, for the input table
requestInputRow	element array	One or more rows of input data

Example

The value for the following `requestInputTable` element corresponds to an array containing one object. That object represents a table named *Table 1* that contains a single input row consisting of the values for seven input fields.

```
"requestInputTable":[
   {
      "name":"Table 1",
      "requestInputRow":[
         {
            "input":[
               {"name":"Age","value":"1"},
               {"name":"BP","value":"1"},
               {"name":"Cholesterol","value":"1"},
               {"name":"Drug","value":"1"},
               {"name":"K","value":"1"},
               {"name":"Na","value":"1"},
               {"name":"Gender","value":"1"}
            ]
         }
      ]
   }
]
```

If the model required data from two input tables, the array would contain multiple objects. For example, the following element represents two tables.

```
"requestInputTable":[
   {
      "name":"Table 1",
      "requestInputRow":[
         {
            "input":[
               {"name":"Age","value":"1"},
               {"name":"BP","value":"1"},
               {"name":"Cholesterol","value":"1"},
            ]
         }
      ]
   },
   {
      "name":"Table 2",
      "requestInputRow":[
         {
            "input":[
               {"name":"Drug","value":"1"},
               {"name":"K","value":"1"},
               {"name":"Na","value":"1"},
               {"name":"Gender","value":"1"}
            ]
         }
      ]
   }
]
```

The requestInputRow element

Represents rows of input values used to produce score results. The value for this element is an element array in which each element represents an input row.

Table 23. Child elements for requestInputRow.

Element name	Value type	Description
input	object array	Values for input fields

Example

The value for the following `requestInputRow` element corresponds to an array containing one element. The element represents a single input row consisting of the values for seven input fields.

```
"requestInputRow":[
   {
      "input":[
         {"name":"Age","value":"1"},
         {"name":"BP","value":"1"},
         {"name":"Cholesterol","value":"1"},
         {"name":"Drug","value":"1"},
         {"name":"K","value":"1"},
         {"name":"Na","value":"1"},
         {"name":"Gender","value":"1"}
      ]
   }
]
```

To include multiple input rows, add `input` elements to the `requestInputRow` element. For example, the following element contains two rows of input data.

```
"requestInputRow":[
   {
      "input":[
         {"name":"Age","value":"1"},
         {"name":"BP","value":"1"},
         {"name":"Cholesterol","value":"1"},
         {"name":"Drug","value":"1"},
         {"name":"K","value":"1"},
         {"name":"Na","value":"1"},
         {"name":"Gender","value":"1"}
      ]
   },
   {
      "input":[
         {"name":"Age","value":"3"},
         {"name":"BP","value":"2"},
         {"name":"Cholesterol","value":"4"},
         {"name":"Drug","value":"2"},
         {"name":"K","value":"3"},
         {"name":"Na","value":"1"},
         {"name":"Gender","value":"2"}
      ]
   }
]
```

The input element:

Defines the scoring input data values. The value for this element is an object array in which each object represents a value for an input field. The format of the values must match the data type, as specified in the scoring configuration metadata.

Table 24. Child elements for input.

Element name	Value type	Description
name	string	Name of the input item

Table 24. Child elements for input (continued).

Element name	Value type	Description
value	string	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`(not case sensitive) or `1` or `false`(not case sensitive) or `0` • date=`yyyy-MM-dd` • daytime=`HH:mm:ss` • timestamp=`yyyy-MM-dd'T'HH:mm:ss`

Example

The value for the following `input` element corresponds to an array containing the values for seven input fields.

```
"input":[
   {"name":"Age","value":"1"},
   {"name":"BP","value":"1"},
   {"name":"Cholesterol","value":"1"},
   {"name":"Drug","value":"1"},
   {"name":"K","value":"1"},
   {"name":"Na","value":"1"},
   {"name":"Gender","value":"1"}
]
```

The context element

Describes the indirect inputs to the scoring model. The value for this element is an object array in which each object represents an input table.

Table 25. Child elements for context.

Element name	Value type	Description
name	string	Name, possibly localized, for the context table
columnName	string array	An ordered list of column names
rowValues	object array	A row of table data, following the order in the columnName list

Example

The value for the following `context` element is an array containing an object representing a context table named *Context 1*. The table contains two columns named *Education* and *Area* having values of *2* and *3*.

```
"context":[
   {
      "name":"Context 1",
      "columnName":["Education","Area"],
      "rowValues":[
         {
            "value":[
               {"value":"2"},
               {"value":"3"}
            ]
         }
      ]
   }
]
```

The rowValues element

Represents a row of context field values. The value for this element is an object array in which each object represents a set of values.

The order of the values corresponds to the order of the field names in the columnName element.

Table 26. Child elements for rowValues.

Element name	Value type	Description
value	object array	Values for the indirect model inputs

Example

The value for the following `rowValues` element corresponds to an array containing a single object representing two values.

```
"rowValues":[
   {
      "value":[
         {"value":"2"},
         {"value":"3"}
      ]
   }
]
```

The value element:

Provides values for the context fields. The value for this element is an object array in which each object represents a value.

Table 27. Child elements for value.

Element name	Value type	Description
value	string	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`(not case sensitive) or `1` or `false`(not case sensitive) or `0` • date=`yyyy-MM-dd` • daytime=`HH:mm:ss` • timestamp=`yyyy-MM-dd'T'HH:mm:ss`

Example

The array for the following `value` element contains two values.

```
"value":[
   {"value":"2"},
   {"value":"3"}
]
```

The scoreResult object

Represents the results of a scoring request as an object containing the output specified by the scoring configuration.

The "Score POST" on page 16 method returns the `scoreResult` object in the response body.

Table 28. Child elements for scoreResult.

Element name	Value type	Description
id	string	Internal identifier for the score result
columnNames	element array	Names of the scoring functions included in the result
rowValues	object array	Values for the scoring functions
returnedRequestInputTable	object array	Tables containing the input values on which the scores are based. The presence of this array is determined by the scoring configuration settings.

Example

The following object corresponds to a scoring response for a model having seven input fields. The response contains a value of *2* for the *Prediction* scoring function and a value of *1.34153* for the *Confidence* scoring function.

```
{
   "id":"954d9db3-049f-464c-b218-eadd628cf349",
   "columnNames":[
      "name":["Prediction","Confidence"]
   ],
   "rowValues":[
      {
         "reserved":null,
         "value":[{"value":"2"},{"value":"1.3415297614991626"}]
      }
   ],
   "returnedRequestInputTable":[
      {
         "name":"Table 1",
         "id":"Table 1",
         "returnedRequestInputRow":[
            {
               "returnedRequestInputValue":[
                  {"name":"Age","type":"double","value":"1"},
                  {"name":"BP","type":"string","value":"1"},
                  {"name":"Cholesterol","type":"string","value":"1"},
                  {"name":"Drug","type":"string","value":"1"},
                  {"name":"K","type":"double","value":"1"},
                  {"name":"Na","type":"double","value":"1"},
                  {"name":"Gender","type":"string","value":"1"}
               ]
            }
         ]
      }
   ],
}
```

The columnNames element

Identifies the scoring functions included in a score result. The value for this element is an element array in which each element represents a set of scoring functions.

Table 29. Child elements for columnNames.

Element name	Value type	Description
name	string array	Scoring function names.

Example

The value for the following `columnNames` element identifies two scoring functions, *Prediction* and *Confidence*.

```
"columnNames":[
   "name":["Prediction","Confidence"]
]
```

The rowValues element

Represents a row of scoring result values. The value for this element is an object array in which each object represents a set of scoring values.

Table 30. Child elements for rowValues.

Element name	Value type	Description
reserved	string	Internal string that is currently unused.
value	object array	Scoring function values

Example

The value for the following `rowValues` element reports scoring function values of *2* and *1.34153*.

```
"rowValues":[
   {
      "reserved":null,
      "value":[{"value":"2"},{"value":"1.3415297614991626"}]
   }
]
```

The value element

Provides the score result values. The value for this element is an object array in which each object represents a scoring value.

Table 31. Child elements for value.

Element name	Value type	Description
value	string	A scoring value, in string representation.

Example

The value for the following `value` element is an array containing two objects. The first object reports a value of *4* and the second object reports a value of *2.434156*.

```
"value":[
   {"value":"4"},
   {"value":"2.434156"}
]
```

The returnedRequestInputTable element

Describes the input tables used in a score request. The value for this element is an object array in which each object represents an input table.

Table 32. Child elements for returnedRequestInputTable.

Element name	Value type	Description
id	string	Non-localized identifier for the input table

Table 32. Child elements for returnedRequestInputTable (continued).

Element name	Value type	Description
name	string	Name, possibly localized, for the input table
returnedRequestInputRow	object array	One or more rows of input values used for scoring

Example

The value for the following `returnedRequestInputTable` element is an array containing one object representing a table named *Table 1*.

```
"returnedRequestInputTable":[
   {
      "name":"Table 1",
      "id":"Table 1",
      "returnedRequestInputRow":[
         {
            "returnedRequestInputValue":[
               {"name":"Age","type":"double","value":"1"},
               {"name":"BP","type":"string","value":"1"},
               {"name":"Cholesterol","type":"string","value":"1"},
               {"name":"Drug","type":"string","value":"1"},
               {"name":"K","type":"double","value":"1"},
               {"name":"Na","type":"double","value":"1"},
               {"name":"Gender","type":"string","value":"1"}
            ]
         }
      ]
   }
]
```

The returnedRequestInputRow element

Represents rows of input values used to produce score results. The value for this element is an object array in which each object represents an input row.

Table 33. Child elements for returnedRequestInputRow.

Element name	Value type	Description
returnedRequestInputValue	object array	Values used to produce the score results. A value might be null.

Example

The value for the following `returnedRequestInputRow` element is an array containing one object representing a single row of input values.

```
"returnedRequestInputRow":[
   {
      "returnedRequestInputValue":[
         {"name":"Age","type":"double","value":"1"},
         {"name":"BP","type":"string","value":"1"},
         {"name":"Cholesterol","type":"string","value":"1"},
         {"name":"Drug","type":"string","value":"1"},
         {"name":"K","type":"double","value":"1"},
         {"name":"Na","type":"double","value":"1"},
         {"name":"Gender","type":"string","value":"1"}
      ]
   }
]
```

The returnedRequestInputValue element:

Represents the input values used to produce score results. The value for this element is an object array in which each object represents an input field.

Table 34. Child elements for returnedRequestInputValue.

Element name	Value type	Description
name	string	Name of the input item
type	string	Data type of the field. Valid entries include the following values: • boolean • date • daytime • decimal • double • float • integer • long • string • timestamp
value	string	A value, in string representation.

Example

The value for the following `returnedRequestInputValue` element is an array containing several objects. Each object represents the value for a single input field. The inputs consist of four string fields and three double fields.

```
"returnedRequestInputValue":[
   {"name":"Age","type":"double","value":"1"},
   {"name":"BP","type":"string","value":"2"},
   {"name":"Cholesterol","type":"string","value":"3"},
   {"name":"Drug","type":"string","value":"4"},
   {"name":"K","type":"double","value":"3"},
   {"name":"Na","type":"double","value":"2"},
   {"name":"Gender","type":"string","value":"1"}
]
```

The scoringServiceDetails object

Provides details about the capabilities of the Scoring Service. In addition to reporting the service version, this object includes information about each individual score provider currently installed.

The "Service GET" on page 7 method returns the `scoringServiceDetails` object in the response body.

Table 35. Child elements for scoringServiceDetails.

Element name	Value type	Description
version	string	The service version number
scoreProviderDetails	object array	Information about the score providers available in the system

Example

The following object indicates that version *5.0.0.0.155* of the Scoring Service is in use. The `Modeler Score Provider` score provider is available for scoring requests using IBM SPSS Modeler streams. In addition, the `Score Provider - SmartScore` score provider is available for scoring requests using PMML files.

```
{
   "version":"5.0.0.0.155",
   "scoreProviderDetails":[
      {
         "supportedMimeTypes":["application\/x-vnd.spss-clementine-stream"],
         "version":"1.0",
         "name":"Modeler Score Provider",
         "id":"091ed7cda16d295c000001327c563c4593de"
      },
      {
         "supportedMimeTypes":["application\/x-vnd.spss-pmml"],
         "version":"1.0",
         "name":"Score Provider - SmartScore",
         "id":"091ed7cd82a9085b000001327c118593a316"
      }
   ]
}
```

The scoreProviderDetails element

Provides details about the score providers in the system. The value of this element consists of an object array in which each object corresponds to a score provider.

The following table identifies the elements comprising each object in the array.

Table 36. Child elements for scoreProviderDetails.

Element name	Value type	Description
id	string	Non-localized identifier for the score provider
name	string	Name, possibly localized, for the score provider
version	string	The score provider version number
supportedMimeValue types	string array	One or more MIME types indicating the type of files that can be used with the score provider

Example

The following `scoreProviderDetails` element consists of an array containing one object. The object corresponds to version 1.0 of the `Modeler Score Provider` score provider that is used for IBM SPSS Modeler stream files.

```
"scoreProviderDetails":[
   {
      "supportedMimeTypes":[
         "application\/x-vnd.spss-clementine-stream"
      ],
      "version":"1.0",
      "name":"Modeler Score Provider",
      "id":"091ed7cda16d295c000001327c563c4593de"
   }
]
```

Appendix. Deprecated features

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

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

The following tables indicate what is deprecated. Where possible, the table also indicates the recommended migration action.

Table 37. Features deprecated in previous versions

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

For updated information about deprecated features, see the IBM Knowledge Center.

Notices

Privacy policy considerations

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

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

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

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

Trademarks

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

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

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

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

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

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

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

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

Glossary

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

The following cross-references are used in this glossary:

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

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

A

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

ACL See access control list.

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

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

AD See Active Directory.

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

API See application programming interface.

appender
A component that receives logging requests from a logger and writes log statements to a specified file or console. See also logger.

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

B

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

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

BLOB See binary large object.

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

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

C

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

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

character string might exceed the limits of the VARCHAR data type.

CLOB See character large object.

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

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

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

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

CWM See common warehouse metamodel.

D

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

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

DN See distinguished name.

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

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

DOM See Document Object Model.

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

DTD See document type definition.

E

EAR See enterprise archive.

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

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

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

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

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

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

F

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

G

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

I

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

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

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

J

JAAS See Java Authentication and Authorization Service.

JAR See Java archive.

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

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

Java Generic Security Services (JGSS)
A specification that provides Java programs access to the services that include the signing and sealing of messages and a generic authentication mechanism.

Java Naming and Directory Interface (JNDI)
An extension to the Java platform that provides a standard interface for heterogeneous naming and directory services.

JGSS See Java Generic Security Services.

JNDI See Java Naming and Directory Interface.

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

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

K

KDC See key distribution center.

Kerberos
A network authentication protocol that is based on symmetric key cryptography. Kerberos assigns a unique key, called a

ticket, to each user who logs on to the network. The ticket is embedded in messages that are sent over the network. The receiver of a message uses the ticket to authenticate the sender.

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

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

L

LDAP See Lightweight Directory Access Protocol.

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

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

logger A component that prepares log statements to be written to console or log file. See also appender.

M

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

metamodel
A model that defines the language for expressing a model.

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

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

MIME See Multipurpose Internet Mail Extensions.

MOF See meta-object facility.

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

N

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

O

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

ODS See Output Delivery System.

OMG See Object Management Group.

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

P

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

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

R

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

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

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

resource
A content repository object.

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

role A set of permissions or access rights. See also action.

S

SAX See Simple API for XML.

schedule
A content repository object that triggers job execution.

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

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

service provider interface (SPI)
An API that supports replaceable components and can be implemented or extended by a third party.

SGML
See Standard Generalized Markup Language.

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

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

single sign-on (SSO)
An authentication process in which a user can access more than one system or application by entering a single user ID and password.

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

SPI See service provider interface.

SSO See single sign-on.

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

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

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

T

TGT See ticket-granting ticket.

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

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

U

Universally Unique Identifier (UUID)
The 128-bit numeric identifier that is used to ensure that two components do not have the same identifier.

UUID See Universally Unique Identifier.

V

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

W

W3C See World Wide Web Consortium.

WAR See web archive.

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

Web Services Description Language (WSDL)
An XML-based specification for describing networked services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

World Wide Web Consortium (W3C)
An international industry consortium set up to develop common protocols to promote evolution and interoperability of the World Wide Web.

WSDL
See Web Services Description Language.

X

XMI See XML Metadata Interchange.

XML See Extensible Markup Language.

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

XSL See Extensible Stylesheet Language.

Index

Printed in USA