IBM SPSS Collaboration and Deployment Services
Version 8 Release 2

Customization Reference

Note

Before using this information and the product it supports, read the information in “Notices” on page 59.

Product Information

This edition applies to version 8, release 2, modification 2 of IBM® SPSS® Collaboration and Deployment Services and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Customization overview

IBM SPSS Collaboration and Deployment Services offers a variety of techniques for customizing the interaction with content stored in the repository, including the following:

- Referencing repository content directly using uniform resource locators (URL) parameters.
- Creating custom web pages based on information obtained from reports and queries stored in the repository using JavaServer Page tags.
- Embedding repository content, such as reports, on portal pages.
- Performing batch processing of repository content using Python scripting. For more information, see the IBM SPSS Collaboration and Deployment Services - Essentials for Python documentation

Prerequisites

For proper processing of custom dialogs, the following requirements must be satisfied:

- A remote execution server for IBM SPSS Statistics must be set up in IBM SPSS Deployment Manager and then designated as the default server for executing custom dialog syntax using the browser-based IBM SPSS Deployment Manager. It is also possible to configure individual custom dialogs to use a specific IBM SPSS Statistics server different from the system default.
- The user must be assigned the *Run Custom Dialogs* action to be able to execute custom dialogs.
- IBM SPSS Statistics save file access is enabled by the IBM SPSS Statistics data file driver service, which must be installed, started, and then designated as the driver for IBM SPSS Statistics data using the browser-based IBM SPSS Deployment Manager. The software is available as a download to IBM Corp. customers.

Important: The IBM SPSS Statistics data file driver service must run on a host with the same operating system type as the repository host. For example, it is impossible to use a repository running on a Linux server in conjunction with the data file driver service running on a Windows server. For information about IBM SPSS Collaboration and Deployment Services system configuration and actions, see the administrator's documentation.

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 focused on the strategic function listed under the recommended migration action. Typically, a feature is not deprecated unless an equivalent alternative is provided.

No features have been deprecated in this release. For reference purposes, the following table indicates features that were deprecated in recent previous versions of the product. Where possible, the table also indicates the recommended migration action.

Table 1. 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

Table 1. Features deprecated in previous versions (continued)	
Deprecation	**Recommended migration action**
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)
`clemrtl_setLogFile()` API function	None

Chapter 2. URL parameters

You can access IBM SPSS Collaboration and Deployment Services Deployment Portal reports and other repository objects using direct URLs (Uniform Resource Locators).

With URLs, you can directly share reporting information in different ways such as embedding reporting into your external websites and applications. This reference document lists various URL parameters and contains some tips for building and using IBM SPSS Collaboration and Deployment Services Deployment Portal URL query strings. For assistance, contact Technical Support.

The URL parameters outlined in this document are unrelated to the URLs available in IBM SPSS Deployment Manager.

URL base path

The base path for all requests is:

```
http://<hostname>:<port>/<contextpath>/peb/view/<content repository path>
```

or

```
http://<hostname>:<port>/<contextpath>/peb/view?id=<object-id>
```

<hostname>
Name or IP address of the repository server

Note: An IPv6 address must be enclosed in square brackets, such as `[3ffe:2a00:100:7031::1]`.

<port>
Port number on which to connect to the repository server

<contextpath>
Optional custom context path for the repository server

<content repository path>
Resource path of the repository object on which to act

<object-id>
Resource ID of the repository object on which to act

Examples

```
http://yourserver:8080/peb/view/sample/employee.str
http://yourserver:8080/peb/view?id=0a58c3461e885d240000010f4cc607188375
```

Query string

The base path for the URL reference can be followed by a query string containing parameters that provide additional processing information. The query string begins with a question mark and contains parameter/value pairs separated by ampersands (&).

Note that if a repository item is referenced by its resource identifier, the question mark initiating the query string is already present for the `id` parameter and should not be repeated for any other parameters.

At a minimum, a URL must contain the content repository path in the base path or the `id` parameter. Other parameters are optional. Unless otherwise stated, parameters and their values are case sensitive. Some parameters, such as `username` and `password`, are used in virtually all URL queries, while the use of other parameters may depend on the type of item being referenced in the query. Note that the system can be configured to use a custom authentication mechanism to eliminate the need to supply security credential parameters in the query string.

Reserved characters like & and excluded US-ASCII characters like # should be URL encoded before being specifying as a parameter value in the query string. However, characters in the reserved set are not reserved in all contexts. In general, a character is reserved if the semantics of the URI changes if the character is replaced with its escaped US-ASCII encoding. Hence some characters (like ?, =, and :) are not reserved in the parameter values, but characters like & and # are, and hence need to be URL encoded.

For example, the & character should be URL encoded as %26. Thus, the following URL:

```
http://yourserver:8080/peb/view/sample/employee.str?username=testuser&admin
```

should be specified as

```
http://yourserver:8080/peb/view/sample/employee.str?username=testuser%26admin
```

The following sections describe each parameter.

Common parameters

Common parameters are used in virtually all URL references, or are used across multiple types of repository items.

The id parameter

The id parameter specifies the repository identifier for the item on which to act.

Syntax

```
id=<identifier>
```

The value of *<identifier>* corresponds to the repository object identifier.

Example

```
http://yourserver:8080/peb/view?id=0a58c3461e885d240000010f4cc607188375
```

The version parameter

The version parameter specifies the version of the repository object on which to act using the version marker or the version label.

Special characters, such as spaces, must be escaped. Omit this parameter to display the LATEST version.

Syntax

```
version=m.<version marker>
version=l.<label>
```

The value of *<version marker>* corresponds to the version of the repository object. Alternatively, the value of *<label>* designates the version label of the repository object.

Examples

```
http://srvr:80/peb/view/sample/emp.str?version=m.1:2006-12-04%2020:39:17.995

http://yourserver:8080/peb/view/sample/employee.str?version=l.firstVersion
```

The username parameter

The username parameter specifies the user with which to log in to the system.

Syntax

```
username=<user_ID>
```

username=<user_ID>

The value of *<user_ID>* is the user identifier of the person logging in to the repository server.

Example

```
http://yourserver:8080/peb/view/sample/employee.str?username=validUser
&password=pass&provider=Native
```

The `password` parameter

The `password` parameter specifies the password with which to log in to the system.

Syntax

```
password=<password>
```

The value of *<password>* specifies the password of the person logging in to the repository server.

Example

```
http://yourserver:8080/peb/view/sample/employee.str?username=validUser
&password=pass&provider=Native
```

The `provider` parameter

The `provider` parameter specifies the security provider against which to validate credentials.

A value for `provider` must be specified if the `username` and `password` parameters are used.

Syntax

```
provider=<provider>
```

The value of *<provider>* specifies the security provider. Valid values include:

- *Native* for the built-in provider
- *AD_<name>/<domain>* for Active Directory, where *<name>* corresponds to the security provider name within the system and *<domain>* corresponds to the DNS namespace

Special characters, such as spaces, must be escaped.

Example

```
http://yourserver:8080/peb/view/sample/employee.str
?username=validUser&password=pass&provider=Native
```

The `promptstate` parameter

The `promptstate` parameter specifies whether to suppress the runtime prompt dialog for prompted variable values that are not specified in the query string.

Syntax

```
promptstate=<x>
```

A value of *1* for *<x>* suppresses the runtime prompt dialog, using the specified default variable value for any prompted variables that are not specified. A value of *2* displays the runtime prompt dialog for any prompted variables that are not specified. Alternately, you can omit this parameter to allow the prompt dialog to be displayed.

Example

```
http://yourserver:8080/peb/view/sample/employee.viz?&username=validUser
&password=pass&provider=Native&fragment=true&outputtype=html
&var_EmployeeID=1&promptstate=1
```

The waitstate parameter

The waitstate parameter specifies whether to suppress the Wait screen while a report is running.

Syntax

```
waitstate=<x>
```

A value of *1* for *<x>* suppresses the Wait screen. Omit this parameter to display the Wait screen.

Example

```
http://yourserver:8080/peb/view/sample/employee.viz?username=validUser
&password=pass&provider=Native&fragment=true&outputtype=html
&var_EmployeeID=1&promptstate=1&waitstate=1&fragment=true
```

The partId parameter

The partId parameter identifies a specific part of the repository object being referenced.

For HTMLC files, this parameter can reference a specific file within the archive. For IBM SPSS Statistics output files (.spw), the parameter corresponds to the index as shown in the outline for the file. For example, to get the first part, specify partId=0.

Syntax

```
partId=<reference_id>
```

The value of *<reference_id>* is one of the following values:

- The relative path and name of a file within an HTMLC file
- The index of the wanted output within an .spw file

Example

```
http://yourserver:8080/peb/view/output.htmlc?username=validUser
&password=pass&provider=Native&partId=img/chart.png

http://yourserver:8080/peb/view/output.spw?username=validUser
&password=pass&provider=Native&partId=1
```

The outputtype parameter

The outputtype parameter specifies the file type of the result set.

Syntax

```
outputtype=<file_type>
```

The value of *<file_type>* corresponds to one of the values in the following tables.

Table 2. Visualization output types

Value	Returns
png	Portable Network Graphics format
emf	Enhanced Metafile format
jpeg	JPEG
html	HTML. This is a valid output format for visualization reports only when the output is a table. If HTML is specified as the format for a visualization report that does not produce a table, the output is converted to a PNG image.
pdf	PDF

Table 2. Visualization output types (continued)	
Value	**Returns**
ask	a prompt for the user at runtime to specify an output format

Table 3. Custom dialog output types	
Value	**Returns**
SPW	IBM SPSS Statistics web output viewer
HTML	HTML

Example

```
http://yourserver:8080/peb/view/sample/employee.viz?username=validUser
&password=pass&provider=Native&fragment=true&outputtype=html
```

The `format` parameter

The `format` parameter specifies whether to return the original file stored to the repository, rather than running the file.

Syntax

```
format=raw
```

The value of *raw* returns the original file.

Example

```
http://yourserver:8080/peb/view/sample/employee.rptdesign?username=validUser
&password=pass&provider=Native&format=raw
```

The `fragment` parameter

The `fragment` parameter specifies whether to display the IBM SPSS Collaboration and Deployment Services Deployment Portal user interface elements (i.e., header, footer, Content Repository tree) with the report results.

Syntax

```
fragment=true
```

The value of *true* suppresses the IBM SPSS Collaboration and Deployment Services Deployment Portal interface elements. Omit this parameter to display the interface.

Example

```
http://yourserver:8080/peb/view/sample/employee.viz?username=validUser
&password=pass&provider=Native&fragment=true
```

Parameters for variables

For non-report repository items that use variables, such as jobs, the value for a variable can be specified by including the variable name and value in the URL query string.

For report items, the variable name must be preceded by the `var_` prefix.

Syntax

```
<variable>=<value>
```

The value of *<variable>* corresponds to the name of the variable to satisfy. The value of *<value>* is the entry to use to satisfy the specified report variable

Example

```
http://yourserver:8080/peb/view/sample/myJob?username=validUser
&password=pass&provider=Native&region=1
```

Report parameters

Report parameters are used in references to reports stored within the IBM SPSS Collaboration and Deployment Services Repository.

The reports may be visualization reports.

The dbcredential_datasourcename parameter

The dbcredential_datasourcename parameter specifies the credential with which to log on to the data source.

This is used if the data source user ID differs from the IBM SPSS Collaboration and Deployment Services Deployment Portal user ID.

Syntax

```
dbcredential_<datasourcename>=<credential id>
```

The value of *<datasourcename>* is the name of the given data source. The value of *<credential id>* is the identifier of the credential object to be used for connecting to the data source.

Example

```
http://yourserver:8080/peb/view/sample/employee.viz
?dbcredential_yourDS=0a58c346cd5b72010000010f3df6d5e28130
```

The dbuser_datasourcename parameter

The dbuser_datasourcename parameter specifies the user identifier with which to log on to the data source.

This is used if the user identifier for the data source differs from the user identifier for the IBM SPSS Collaboration and Deployment Services Deployment Portal.

Syntax

```
dbuser_<datasourcename>=<user_ID>
```

The value of *<datasourcename>* is the name of the given data source. The value of *<user_ID>* is the user identifier of the person connecting to the data source.

Example

```
http://yourserver:8080/peb/view/sample/employee.viz?dbuser_yourDS=sa
```

The dbpwd_datasourcename parameter

The dbpwd_datasourcename parameter specifies the password with which to log on to the data source.

This is used if the data source user identifier differs from the IBM SPSS Collaboration and Deployment Services Deployment Portal user identifier.

Syntax

```
dbpwd_<datasourcename>=<password>
```

The value of *<datasourcename>* is the name of the given data source. The value of *<password>* is the password of the person connecting to the data source.

Example

```
http://yourserver:8080/peb/view/sample/employee.viz
?dbuser_yourDB=sa&dbpwd_yourDB=sa
```

Note: If the `dbcredential_datasourcename` parameter has been specified, that parameter will be considered for connecting to the data source before the `dbuser_datasourcename` and `dbpwd_datasourcename` parameters.

The `width` parameter

The `width` parameter specifies width of the resulting image or graph. This parameter is used specifically with visualization reports.

For reports containing height and width specifications, both `height` and `width` parameters must be provided. If either parameter is missing, the graph would be rendered with its default height and width.

Syntax

```
width=<x>
```

The value of *<x>* specifies the integer value for the width in pixels.

Example

```
http://yourserver:8080/peb/view/sample/employee.viz?username=validUser
&password=pass&provider=Native&fragment=true&outputtype=html
&var_EmployeeID=1&promptstate=1&waitstate=1&width=500&height=1000
```

The `height` parameter

The `height` parameter specifies height of the resulting image or graph. This parameter is used specifically with visualization reports.

For reports containing height and width specifications, both `height` and `width` parameters must be provided. If either parameter is missing, the graph would be rendered with its default height and width.

Syntax

```
height=<x>
```

The value of *<x>* specifies the integer value for height in pixels.

Example

```
http://yourserver:8080/peb/view/sample/employee.viz?username=validUser
&password=pass&provider=Native&fragment=true&outputtype=html
&var_EmployeeID=1&PROMPTSTATE=1&waitstate=1&width=500&height=1000
```

The `var_variable` parameter

The `var_variable` parameter specifies the value to use to satisfy the specified report variable.

Syntax

```
var_<variable>=<value>
```

The value of *<variable>* is the name of the variable passed to the report.

Example

```
http://yourserver:8080/peb/view/sample/employee.rptdesign?username=validUser
&password=pass&provider=Native&fragment=true&outputtype=html&var_EmployeeID=1
```

Notes

- For reports, specifying a variable value on the URL will suppress the runtime prompt for that variable.
- To specify a single variable value (=), use the syntax `var_Lastname=Curtis`
- To specify multiple variable values (IN), use the syntax `var_Lastname=Curtis&var_Lastname=McLind`
- To specify a range of variable values (BETWEEN), use the syntax `var_Dateship=3-1-2007&var_Dateship=3-31-2007`
- To specify values for multiple variables, use the syntax `var_Lastname=Curtis&var_Dateship=3-1-2007&var_Dateship=3-31-2007`

Scoring parameters

Scoring parameters are used when referencing scoring configurations to generate scores.

The `dataset` parameter

The `dataset` parameter specifies the location of a SQL definition that will be used for batch scoring.

The value of this parameter will be a relative path within the IBM SPSS Collaboration and Deployment Services Repository.

Syntax

```
dataset=<location>
dataset.<tableID>=<location>
```

The value of *<location>* is the repository path.

If the data set includes multiple tables, append the parameter with a period followed by the table identifier corresponding to the wanted table. You can obtain the value of *<tableID>* by examining the scoring model in IBM SPSS Modeler.

Example

```
http://yourserver:8080/peb/view/myPMML.xml?username=validUser
&password=pass&scoring_configuration=testConfig
&dataset=/datasets/dataset.sql
```

The `dataset_label` parameter

The `dataset_label` parameter allows the user to specify the appropriate version of the data set. The specified data set version must be compatible with the data provider defined in the scoring configuration.

If not specified, the *LATEST* version is used.

Syntax

```
dataset_label=<myLabel>
dataset_label.<tableID>=<myLabel>
```

The value of *<myLabel>* is the label for the data set version.

If the data set includes multiple tables, append the parameter with a period followed by the table identifier corresponding to the wanted table. You can obtain the value of *<tableID>* by examining the scoring model in IBM SPSS Modeler.

Example

```
http://yourserver:8080/peb/view/myPMML.xml?username=validUser
&password=pass&scoring_configuration=testConfig
&dataset=/datasets/dataset.sql&dataset_label=PRODUCTION
```

The `dataset_table` parameter

The `dataset_table` parameter allows the user to specify a table within a data set.

Syntax

```
dataset_table=<myTable>
dataset_table.<tableID>=<myTable>
```

The value of *<myTable>* is the name of the data set table.

If the data set includes multiple tables, append the parameter with a period followed by the table identifier corresponding to the wanted table. You can obtain the value of *<tableID>* by examining the scoring model in IBM SPSS Modeler.

Example

```
http://yourserver:8080/peb/view/myPMML.xml?username=validUser
&password=pass&scoring_configuration=testConfig&fragment=true&
dataset=/data/mySet&
dataset_label=PRODUCTION&
dataset_rowlimit=2&
dataset_table=myTable&
promptstate=1
```

The `dataset_rowlimit` parameter

The user may limit the amount of data processed from the data set for batch scoring. This will help prevent long running processes.

The `dataset_rowlimit` specifies the number of rows of data that will be extracted from the data set.

Syntax

```
dataset_rowlimit=<x>
dataset_rowlimit.<tableID>=<x>
```

The value of *<x>* denotes the number of data set rows to be extracted.

If the data set includes multiple tables, append the parameter with a period followed by the table identifier corresponding to the wanted table. You can obtain the value of *<tableID>* by examining the scoring model in IBM SPSS Modeler.

Example

```
http://yourserver:8080/peb/view/myPMML.xml?username=validUser
&password=pass&scoring_configuration=testConfig
&dataset=/datasets/dataset.sql&dataset_rowlimit=1000
```

The `scoring_configuration` parameter

The `scoring_configuration` parameter specifies the scoring configuration used by the scoring engine to score the specified model.

Syntax

```
scoring_configuration=<configName>
```

The value of *<configName>* is the name of scoring configuration to use for scoring. The specified configuration must be able to process a scoring request. A reference to a suspended configuration will be unable to produce scores.

Example

```
http://yourserver:8080/peb/view/myPMML.xml?username=validUser
&password=pass&scoring_configuration=testConfig
&dataset=/datasets/dataset.sql
```

The batch_type parameter

The batch_type parameter specifies which scoring input prompts should be displayed.

If the parameter specifies *dataset*, the scoring interface will generate the input prompts for the data set and label. If the batch_type is not specified and parameter inputs are not defined, the interface based on scoring parameters is used.

Syntax

```
batch_type=<inputPrompt>
```

The value of *<inputPrompt>* indicates the source for the input prompts. Currently, the only supported source is *dataset*. Omit this parameter to prompt the user for input values based on parameters.

Example

```
http://yourserver:8080/peb/view/myPMML.xml?username=validUser
&password=pass&scoring_configuration=testConfig&batch_type=dataset
```

Custom dialog parameters

Custom dialog parameters are used when referencing custom dialog (.spd) files.

This functionality requires IBM SPSS Statistics adapters in the IBM SPSS Collaboration and Deployment Services environment. For more information, see the IBM SPSS Statistics installation documentation.

The dataset.uri parameter

The URI of the data set to be use by the custom dialog. For .sav files in the IBM SPSS Collaboration and Deployment Services Repository, the URI can be specified as a repository path or the resource ID. When the URI references a file on the file system, the path to the file must be valid from the IBM SPSS Statistics data file driver server that is used to retrieve the variable metadata. It must also be a valid path on the IBM SPSS Statistics Server that will execute the syntax. If a repository data set object is used, the version of the object can be appended to the URI either as a version maker or a label.

Syntax

```
dataset.uri=<myURI>
```

The value of *<myURI>* denotes the URI for the data set.

Example

```
http://yourserver:8080/peb/view/myDialog.spd
?dataset.uri=spsscr:///Datasets/SpecificURI.sav

http://yourserver:8080/peb/view/myDialog.spd
?dataset.uri=spsscr:///?id=0a30063bc975ede40000011cafb8deda8327.

http://yourserver:8080/peb/view/myDialog.spd
?dataset.uri=file:///C:/Program%20Files/SPSSInc/Samples/accidents.sav
```

The dataset.table parameter

For IBM SPSS Collaboration and Deployment Services data sources, the table to be used by the custom dialog. If no name is specified, the user will be prompted to select from the list of tables.

Syntax

```
dataset.table=<myTable>
```

The value of *<myTable>* identifies the table to use.

Example

```
http://yourserver:8080/peb/view/myDialog.spd
?dataset.uri=spsscr:///Datasets/dataset.table=myTableName
```

The `dataset.prompt` parameter

Indicates that the user will be forced to select a dataset for the custom dialogs. Otherwise, the dataset selected for the first dialog opened by the user that contains matching search criteria during a session will be used for any subsequent custom dialogs that are not configured to use a specific dataset.

Syntax

```
dataset.prompt=<indicator>
```

The value of *<indicator>* is either *true* or *false*.

Example

```
http://yourserver:8080/peb/view/myDialog.spd?dataset.prompt=true
```

The `dataset.search.criteria` parameter

Search criteria to be used for generating a list of data sets at run time. The entire search string must be entered on a single line. Multiple conditions may be combined using parenthesis, logical and, and logical or.

Search criteria

`$$repository/title_field_name=<Object name>`

`$$search/mimetype=<Object MIME type>`

`$$repository/version_created_by_field=<Created by user stamp>`

`$$repository/version_created_date_field=<Version created date>`

`$$repository/description_field_name=<Object description>`

`$$repository/object_last_modified_by=<Created by user stamp>`

Syntax

```
dataset.search.criteria=<myCriteria>
```

The value of *<myCriteria>* corresponds to the search expression.

Example

```
# locates all SAV files
http://yourserver:8080/peb/view/myDialog.spd
?dataset.search.criteria=
'$$search/mimetype%3Dapplication/x-vnd.spss-spss-data%20or%20
$$search/mimetype%3Dapplication/x-vnd.spss-statistics-data'

# locates all files that match the keyword SPECIAL_DATASET
http://yourserver:8080/peb/view/myDialog.spd
?dataset.search.criteria='$$repository/keyword_field_name%3D%3DSPECIAL_DATASET'
```

The `variable.display` parameter

The `variable.display` parameter indicates whether or not to show variable names or labels.

Syntax

```
variable.display=<type>
```

The value of *<type>* is either *names* to show variable names or *labels* to show variable labels.

Example

```
    http://yourserver:8080/peb/view/myDialog.spd
?dataset.uri=spsscr:///Datasets/SpecificURI.sav&variable.display=labels
```

The `variable.sort` parameter

The `variable.sort` parameter specifies the sort criterion used for ordering variables.

Syntax

```
variable.sort=<myCriteria>
```

The value of *<myCriteria>* is one of the following:

- *none* to do no additional sorting beyond the original order in the data
- *alphanumeric* for an alphanumeric sort of field names or labels, whichever is displayed
- *measurement* to sort by the field measurement levels

Example

```
http://yourserver:8080/peb/view/myDialog.spd
?dataset.uri=spsscr:///Datasets/SpecificURI.sav&variable.sort=alphanumeric
```

The `stylesheet.url` parameter

If you are using a CSS style sheet stored in the repository, the repository URL of the style sheet.

Syntax

```
stylesheet.url=<myURL>
```

The value of *<myURL>* is the URL for the style sheet.

Example

```
http://yourserver:8080/peb/view/myDialog.spd
?stylesheet.url=/peb/view/EditBox_pes.css&fragment=true
```

The `stylesheet.name` parameter

If you are using a CSS style sheet embedded in the custom dialog file, the name of the style sheet. The style sheet file can be added to the custom dialog file using compressed archive software, such as WinZip.

Syntax

```
stylesheet.name=<myStyles>
```

The value of *<myStyles>* specifies the name of the style sheet.

Example

```
http://yourserver:8080/peb/view/myDialog.spd
?stylesheet.name=EditBox.css
```

The `javascript.url` parameter

If you are using a JavaScript stored in the repository, the repository URL of the script file.

Syntax

```
javascript.url=<myURL>
```

The value of *<myURL>* is the URL for the JavaScript file.

Example

```
http://yourserver:8080/peb/view/myDialog.spd
?javascript.url=/peb/view/EditBox_pes.js&fragment=true
```

The `javascript.name` parameter

If you are using a JavaScript sheet embedded in the custom dialog file, the name of the script file.

Syntax

```
javascript.name=<myFile>
```

The value of *<myFile>* is the name of the JavaScript file.

Example

```
http://yourserver:8080/peb/view/myDialog.spd?javascript.name=EditBox.js
```

The `validate.method` parameter

A validation method from the specified JavaScript file to call before a page is submitted. The form that is being submitted should be the only parameter for the method. Upon evaluating the form input, the method should return a Boolean value. The method should return true if everything is valid and false if the submit should be cancelled.

Syntax

```
validate.method=<myMethod>
```

The value of *<myMethod>* is the name of the method in the JavaScript file to use for validation.

Example

```
http://yourserver:8080/peb/view/myDialog.spd?javascript.name=EditBox.js
&validate.method=myValidate
```

The `output.format` parameter

The format of the output to create. Default format is IBM SPSS Statistics Web Output viewer format (*.spw*. In some cases, it may be appropriate to create HTML instead. The output format is case sensitive.

This parameter specifies the same information as the `outputtype` parameter, but is honored only for custom dialogs.

Syntax

```
output.format=<myFormat>
```

The value of *<myFormat>* is the format for the output. Valid values include the following:

- *SPW* for IBM SPSS Statistics web output viewer
- *HTML* for HTML output

Example

```
http://yourserver:8080/peb/view/myDialog.spd?output.format=SPW
```

The `output.filename` parameter

The name of the output file. If not specified, the output file will be generated with the same name as the custom dialog file name but without the *.spw* extension.

Syntax

```
output.filename=<myFile>
```

The value of *<myFile>* is the name for the output file.

Example

```
http://yourserver:8080/peb/view/myDialog.spd
?output.filename=MyOutputName.spw
```

The `showOutline` parameter

Indicates whether the outline should be displayed. Default is `true`.

Syntax

```
showOutline=<indicator>
```

The value of *<indicator>* is either *true* or *false*.

Example

```
http://yourserver:8080/peb/view/myDialog.spd?showOutline=true
```

The `allowPivoting` parameter

Indicates whether table manipulation should be allowed. When the option is disabled, the user will not be allowed to pivot, flip, or change layers, save views or open data in a new window. Default is `true`.

Syntax

```
allowPivoting=<indicator>
```

The value of *<indicator>* is either *true* or *false*.

Example

```
http://yourserver:8080/peb/view/myDialog.spd?allowPivoting=true
```

The `allowPrinterFriendly` parameter

Indicates whether the printer friendly display can be opened for a particular table. Default is `true`.

Syntax

```
allowPrinterFriendly=<indicator>
```

The value of *<indicator>* is either *true* or *false*.

Example

```
http://yourserver:8080/peb/view/myDialog.spd?allowPrinterFriendly=true
```

The `allowDownload` parameter

Indicates whether the data can be downloaded to a local data file. Default is `true`.

Syntax

```
allowDownload=<indicator>
```

The value of *<indicator>* is either *true* or *false*.

Example

```
http://yourserver:8080/peb/view/myDialog.spd?allowDownload=true
```

The showLogs parameter

Indicates whether log entries should be shown in the output. Default is `true`.

Syntax

```
showLogs=<indicator>
```

The value of *<indicator>* is either *true* or *false*.

Example

```
http://yourserver:8080/peb/view/myDialog.spd?showLogs=true
```

The `statistics.server` parameter

IBM SPSS Statistics server used to execute the syntax of the custom dialog. The value may be a URI or a name that references a server defined in IBM SPSS Collaboration and Deployment Services. If you have multiple servers, this value can specify the URI or name of a server cluster.

Syntax

```
statistics.server=<serverIdentifier>
```

The value of *<serverIdentifier>* identifies the server to use for execution.

Example

```
http://yourserver:8080/peb/view/myDialog.spd?
statistics.server=spsscr:///?id=0a30063bc975ede40000011cafb8deda8327

http://yourserver:8080/peb/view/myDialog.spd
?statistics.server=localStatisticsServer

http://yourserver:8080/peb/view/myDialog.spd
?statistics.server=copServerCluster
```

The `statistics.server.credential` parameter

The credential that should be used to connect to the IBM SPSS Statistics server when executing syntax. The value may be a URI or a name that references a credential for IBM SPSS Collaboration and Deployment Services.

Syntax

```
statistics.server.credential=<myCredential>
```

The value of *<myCredential>* identifies the credential under which execution occurs.

Example

```
http://yourserver:8080/peb/view/myDialog.spd?statistics.server=localStatisticsServer
&statistics.server.credential=spsscr:///?id=0a30063bc975ede40000011cafb8deda8327.

http://yourserver:8080/peb/view/myDialog.spd?statistics.server=localStatisticsServer
&statistics.server.credential=administrator
```

HTML techniques

Use an HTML editor

Many HTML editors can simplify the creation of URL query strings and insert the proper delimiters between parameters.

Use HTML forms to submit requests

IBM SPSS Collaboration and Deployment Services Deployment Portal requests can be submitted from HTML forms included on a web page. For example, a form can be used to allow a user to:

- Select from a list of available reports
- Select an output file type
- Specify prompted variables before submitting the report request
- Supply an ID and password before running a report

The following example references a custom dialog file in the action for a form.

```
<form name='AnalyzeOptions' method='POST' target='Iframe_1'
    action='/peb/view/SamplesStatistics/SPD/Simple.spd
?fragment=true&promptstate=1&waitstate=1'>
    <input type='hidden' name='username' value='userA'/>
    <input type='hidden' name='password' value='passwordA/>
    <input type='hidden' name='provider' value='Native'/>
    <input type='hidden' name='dataset.uri'
value='spsscr:///SamplesStatistics/SAV/multipleResponseData.sav'/>
    <input type='hidden' name='allowPivoting' value='false'/>

    <input name='PromptParameter1' type='checkbox' value='true'/>
    Check the box to select parameter 1

    <br>
    <input type='submit' value='Run Report' />
</form>
```

Use the IBM SPSS Collaboration and Deployment Services Repository to store custom web pages containing relative paths

The repository can be used as a central location for storing all files for a custom website. Relative or absolute paths can be used within the custom website to link to items such as `.css` style sheets, images, IBM SPSS Collaboration and Deployment Services Deployment Portal reporting objects, and JavaScript.

For example, you might store a folder called `MyWebPage` in the repository containing a custom web page called `MyWebPage.htm` and resources such as images, style sheets, and JavaScript files. `MyWebPage.htm` can contain **relative** references to the resources such as the following:

```
<img src="MyLogo.gif?fragment=true">

<script language="javascript" src="MyJS.js?fragment=true">

</script>

<LINK REL="StyleSheet" HREF="MyStyles.css?fragment=true" TYPE="text/css"
MEDIA="screen" />
```

Note that for such relative references to work properly, the web page needs to be accessed using the parameter `fragment=true` in the URL. For example:

```
http://yourserver:port/peb/view/MyWebPage/MyWebPage.htm?
username=validUser&password=pass&provider=Native&fragment=true
```

If you want to store the resources for your website in a different repository location from where your web page is stored, they can be referenced from your web page (for example, `MyWebPage.htm`) using **absolute** paths as follows:

```
<img src="/peb/view/MyWebPage/images/MyLogo.gif?fragment=true">
```

```
<script language="javascript" src="/peb/view/MyWebPage/js/MyJS.js?
fragment=true">

</script>

<LINK REL="StyleSheet" HREF="/peb/view/MyWebPage/CSS/MyStyles.css?
fragment=true" TYPE="text/css" MEDIA="screen" />
```

Or, they can be referenced by using the full host name and port in the path:

```
<img src="http://yourserver:8080/peb/view/MyWebPage/images/MyLogo.gif?
fragment=true">

<script language="javascript" src="http://yourserver:8080/peb/view/
MyWebPage/js/MyJS.js?fragment=true">

</script>

<LINK REL="StyleSheet" HREF="http://yourserver:8080/peb/view/ MyWebPage/CSS/
MyStyles.css?fragment=true" TYPE="text/css" MEDIA="screen"/>
```

Note:

If you create an HTML page that references published reports (`.spw`) embedded in an HTML IFRAME, you must adjust Internet Explorer's privacy settings for third party cookies to avoid seeing the login screen.

This scenario only occurs when you call the `.spw` report(s) from an external web page, using IFRAMEs to embed and display multiple reports on one web page. When accessing the HTML page, the reports will run and will display within the IFRAMEs. When embedding the IBM SPSS Collaboration and Deployment Services Deployment Portal URL parameter into an IFRAME, the cookies will be considered as third party cookies. In Internet Explorer's privacy settings (Tools/Internet Options/Privacy), third party cookies are blocked by default. This results in the login screen being displayed when accessing the web page (even though the URL contains the correct user name and password). To modify this behavior, you can update Internet Explorer's privacy settings. Add the IBM SPSS Collaboration and Deployment Services domain name or IP address to the Managed Sites with "Allow." This ensures all cookies from the address are accepted and no login screen will appear.

Following is an example of the HTML used reference an `.spw` report published to the repository:

```
<iframe frameborder=1 src="http://yourserver:8080/peb/view
/jba/accidents.spw?partId=5&fragment=true&username=admin&password=
yourpwd&provider=Native" style="WIDTH: 800px;HEIGHT: 280px" name="I1">
</iframe>
```

Note that this issue only occurs with Internet Explorer, not other supported browsers.

Chapter 3. IBM SPSS Collaboration and Deployment Services Tag Library

A JavaServer Pages (JSP) tag library is provided with IBM SPSS Collaboration and Deployment Services for administrators and advanced users who want to create relationships between repository items and create custom Web pages (`.jsp` pages) containing items that can feed values to one another.

The tag library provides the following basic functionality:

Authentication: You can set the user, password, and security provider and share across any items or prompts defined on the page. Authentication is required to access the items in the IBM SPSS Collaboration and Deployment Services Repository and for data source authentication.

Items: You can specify the definition of items, including the target "container" (`<div>` or `<iframe>` element). The items will run using a POST request for IFRAME targets and using Ajax (Asynchronous JavaScript and XML) for DIV targets.

Prompts: You can use prompts to dynamically adjust the parameters used to run items. The prompt location is only restricted to a location on the current page. Prompts can either be user defined or a selected parameter from an existing item definition.

Linking Relationships: You can define relationships between the following items:

- Source report items and target report, job, scoring, or custom dialog items
- A list of prompts and a target item. Both the activation location (`DIV` or `IFRAME`) and the timing (`ONDEMAND`, `ONLOAD`, or `NONE`) are supported.

The tag library framework is made up of the following main parts:

- Public JavaScript API.
- Custom tags and their interactions with each other.
- Tag library beans for data set retrieval.

This document describes each tag function available in the JSP tag library and includes usage examples. After reading this document, we recommend reviewing the sample *.jsp* files shipped with the tag library before creating your own custom pages.

Upgrading to IBM SPSS Collaboration and Deployment Services Tag Library

Note that previous versions of IBM SPSS Collaboration and Deployment Services used a *.tld* file named *reporting-taglib.tld* or *pasw-taglib.tld*. Any existing JSP pages using either of those names should be updated to reference *IBMSPSSTaglib.tld*.

In addition, you should verify that any custom pages created using previous versions of IBM SPSS Collaboration and Deployment Services perform as expected in the current version. Some modifications may be required. For example, changes in report processing may require changes to pages that incorporate reports.

JavaServer Pages architecture

The figure illustrates the architecture underlying the use the tag library. The application server hosting the IBM SPSS Collaboration and Deployment Services Repository includes a servlet engine that transforms the information contained in the library tags into input for web services included in IBM SPSS Collaboration and Deployment Services.

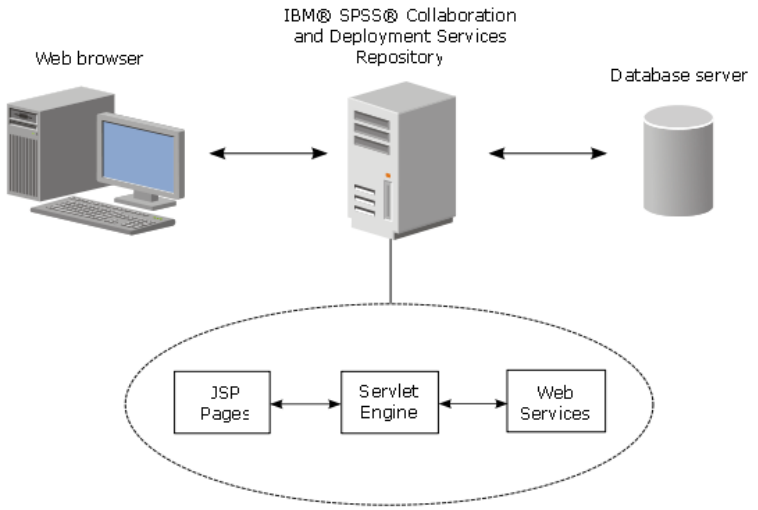

Figure 1. JSP Architecture

In general, the process of running items using the tag library is as follows:

1. The JSP developer uses custom tags to define credentials, prompts, items, and item relationships in a *.jsp* file and stores the file on the application server hosting IBM SPSS Collaboration and Deployment Services Repository.
2. When a client accesses the *.jsp* page, the server evaluates the tags and generates XML data islands or HTML elements as appropriate, which are used by the JavaScript components of the framework to identify and manage relationships between items and prompts.
3. Validations are carried out in each tag handler and appropriate error messages are displayed on the page so the user is aware of any errors at each stage of page creation.
4. A servlet provides support for running items and processing and returning the output.
5. The web service associated with the item type is invoked to run the item and perform various validations.

Supported items

A variety of repository items can be referenced in JSP pages using the IBM SPSS Collaboration and Deployment Services Tag Library. When processing the page, the MIME type of the item determines how the item gets processed.

Valid items include the following:

- Reports
- Jobs
- Scoring models
- Custom interface definitions

Reports

For a report, the repository item must reference a Visualization definition (`*.viz`)

The following properties should be considered when working with report items:

Output. A report item typically generates a single output. Visualization reports, however, generate an image map in addition to the visualization. The output for the item can be delivered in a variety of formats that depend on the report type. Available formats include the following:

- HyperText Markup Language (`*.html`)
- Portable document format (`*.pdf`)

- Report document (*.rptdocument)
- HTML complete (*.htmlc)
- MIME HTML (*.mht)
- Microsoft Word document (*.doc)
- Microsoft PowerPoint (*.ppt)
- Portable Network Graphic (*.png)
- Enhanced Metafile (*.emf)
- Joint Photographic Experts Group (*.jpeg)

Prompts. When processed, the item will prompt for values for any variables defined in the report.

Location restrictions. Output of the *.rptdocument type can only be displayed in an IFRAME.

Item linking. Report items can be used as sources for subsequent items or as targets of other items.

Supported tags. Report items do not support the outputLocation tag. All other tags in the tag library are supported.

The item may include additional information controlling the output display, such as the window title or the presence of a toolbar.

Jobs

For a job, the repository item must reference a job in IBM SPSS Collaboration and Deployment Services, which has a MIME type of application/x-vnd.spss-prms-job.

The following properties should be considered when working with job items:

Output. A job item can generate any number of outputs of varying types. The output produced depends on the steps contained within the job.

Prompts. When processed, the item will prompt for values for any job parameters defined for the job.

Location restrictions. Output from the individual steps within the job must be explicitly defined.

Item linking. Job items can be used as targets of other items but not as sources.

Supported tags. Job items do not support the actionHandler tag. All other tags in the tag library are supported.

Scoring models

For a scoring model, the repository item must reference a file configured for scoring.

Valid types of files include:

- IBM SPSS Modeler stream (*.str)
- Predictive Model Markup Language (PMML)
- Real Time predictive application definition

The following properties should be considered when working with scoring items:

Output. A scoring item produces HTML output.

Prompts. When processed, the item can prompt for values for parameters, a data file, and a model name.

Item linking. Scoring items can be used as targets of other items but not as sources.

Supported tags. Scoring items do not support the outputLocation and actionHandler tags. All other tags in the tag library are supported.

Custom dialogs

For a custom web interface, the repository item must reference a dialog definition (*.spd).

The following properties should be considered when working with custom dialog items:

Output. A custom dialog item generates one of the following:

- A single output file (`*.spw`) that must be targeted to a frame or window
- HTML that can be targeted to a frame/window or a DIV

Prompts. When processed, the item will prompt for values for any prompts defined in the dialog definition. The item can also prompt for data sets. However, any help for prompts defined in the `.spd` file is not used. The application should include its own help references.

Location restrictions. The output can be viewed in a frame, DIV, or a new window.

Item linking. Dialog items can be used as targets of other items but not as sources.

Supported tags. Dialog items do not support the `actionHandler` tag. All other tags in the tag library are supported.

The web deployment properties described for use in a URL referencing a custom dialog item can be specified in the tag library either as properties nested in the `repositoryItem` tag or using the `sourceLinkPrompt` tag.

The `dataset.uri` and `dataset.table` properties should always be defined. In contrast, the `javascript.url`, `javascript.name`, `stylesheet.url`, and `stylesheet.name` properties are all ignored. Values for those properties should be defined within the JSP itself.

Note:

This functionality requires IBM SPSS Statistics adapters in the IBM SPSS Collaboration and Deployment Services environment. For more information, see the IBM SPSS Statistics installation documentation.

Building an application

Each JSP page in a custom application must define some standard directives to allow the tag library to be used and referenced properly.

The first, the `page` directive, sets properties for the entire page itself. These properties include:

- The *language* attribute defining the scripting language used by the page
- The *contentType* attribute specifying the MIME type and character set used for responses to clients
- The *session* attribute indicating whether or not the tag library stores information on the session

The second directive, `taglib`, indicates which tags will be used by the JSP page. Properties defined for this directive include:

- The `uri` attribute specifying the proper path to *IBMSPSSTaglib.tld*
- The `prefix` attribute defining a scope for the tags

Note that previous versions of IBM SPSS Collaboration and Deployment Services used a *.tld* file named *reporting-taglib.tld* or *pasw-taglib.tld*. Any existing JSP pages using either of those names should be updated to reference *IBMSPSSTaglib.tld*.

The following sample uses the `page` directive to define the content type as text/html using the UTF-8 character set, the scripting language as Java, and use of the session object as true. The `taglib` directive identifies the location of the reporting `.tld` file and specifies a prefix of *r* for all tags defined within.

```
<%@ page contentType="text/html;charset=utf-8"
    language="java" session="true" %>

<%@ taglib uri="/WEB-INF/tlds/IBMSPSSTaglib.tld" prefix="r" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    http://www.w3.org/TR/html4/loose.dtd">

<html>
<!-- Rest of HTML / JSP goes here -->
</html>
```

To put your application into production you should plan on creating a web application archive (*.war*) file containing the *.jsp* files and deploy it as a separate web application on your application server. This is the preferred method.

For example, the structure of expanded sample reporting tag library application archive (*IBMSPSSTaglib.tld.war*) included in the default installation of IBM SPSS Collaboration and Deployment Services is as follows:

```
IBMSPSSTaglib
|     index.html
|     setup.html
|---js
|   <JavaScript files>|
|---jsp
|   <Java Server Page files>
|---META-INF
|       MANIFEST.MF
|---WEB-INF
|   |   jboss-classloading.xml
|   |   jboss-deployment-structure.xml
|   |   web.xml
|   |---lib
|   |   <Java archive files>
|   |---tlds
|           IBMSPSSTaglib.tld
|           reporting-taglib.tld
|---xsl
        <Extensible Stylesheet Language files>
```

Note that the TLD (Tag Library descriptions) file and libraries (*.jar* files) are included in the deployed *.war* file. The TLD file is also referenced in the application descriptor file (*web.xml*):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
     <servlet>
         <servlet-name>ReportingTaglibServlet</servlet-name>
         <display-name>
             Servlet responsible for fulfilling all requests from
             reporting taglibs
         </display-name>
         <servlet-class>
             com.spss.report.taglib.servlet.ReportingTaglibServlet
         </servlet-class>
     </servlet
>
      <servlet-mapping>
         <servlet-name>ReportingTaglibServlet</servlet-name>
         <url-pattern>/reportingTaglib/*</url-pattern>
     </servlet-mapping>

      <servlet-mapping>
         <servlet-name>ReportingTaglibServlet</servlet-name>
         <url-pattern>/tagLib/*</url-pattern>
     </servlet-mapping>

     <error-page>
       <error-code>500</error-code>
       <location>/jsp/error/error.jsp</location>
  </error-page>
</web-app>
```

The application descriptor also specifies that *ReportingTaglibServlet* servlet is mapped to the */taglib* and */reportingTagLib* URL patterns, and either URL would call the servlet. The servlet Java class is *com.spss.report.taglib.servlet.ReportingTaglibServlet*. Optional single sign-on functionality is enabled by a servlet filter *Common Authentication Filter* which uses *com.spss.report.taglib.filter.AuthFilter* class and is mapped to the servlet by URL. The filter is initialized with encoding and SSO adapter class parameters.

See your application server's documentation for additional information and requirements for `.war` files.

Important! Application *.war* files that are not deployed by IBM SPSS Collaboration and Deployment Services installation scripts or IBM SPSS Collaboration and Deployment Services Package Manager, such as tag library or custom applications, may need to have class loader order modified. For example, for reporting and scoring tag library applications on WebSphere, class loader order must be set to *Classes loaded with application class loader first* and *.war* class loader policy to *Single class loader for application*.

Implementation details

Users creating custom *.jsp* pages should be aware of the following information:

- Each time the server stops, any *.jsp* that are placed directly in the *tmp/deploy* directory files are lost. To preserve your *.jsp* files, save backup copies to a local drive and copy them back to the server after each restart. Production applications should by packaged in `.war` files.
- For reports containing images or charts, the `.mht` output format combines all the images/charts and HTML in a single file compatible with Internet Explorer only.

- The server pre-validates all tags to the extent possible and writes error messages to the HTTP response as they are identified. This provides the JSP developer assistance in resolving problems as a page is being created. For example, the following items are validated:
  - Verify all required credentials are defined
  - Verify credentials
  - Verify report parameters exist
  - Verify column names exists for a report object
  - Verify the linkage between items is logically sound
- The tags require a repository server.
- The tag library supports links between prompts and items, between row clicks and target items, between reports and drill-down reports in the same frame, and between prompts/table rows and target items opened in separate windows.
- All linkage behavior is hidden from the user and is defined using `sourceLinkRepositoryItem` or `sourceLinkPrompt` JSP tags. The user is not required to understand any technology beyond JSP tag usage.
- All target items must be predefined with parameters to receive the passed parameters.
- For custom dialogs, the standard CSS defines default styles that are included automatically at the point the `repositoryItem` tag is used. To override those styles, include a custom style sheet after the `repositoryItem` tag. For example:

```
<r:repositoryItem name="sample" inputURI="spsscr:///myDialog.spd"
    ...more here...
</r:repositoryItem>

<link rel="stylesheet" type="text/css" href="MyStyles.css">
```

Public JavaScript API

The framework provides JavaScript functions for processing repository items, retrieving bookmarked report values, and retrieving cascading prompt values.

The runRepositoryItem function

The public JavaScript function provided by the framework for running an item is `runRepositoryItem`. It allows the developer to run an item by connecting this JavaScript to an event handler, and activate an item when using prompts.

However, when running an item directly using this function, the normal prompt validation is bypassed. It is the application's responsibility to validate any parameters before invoking function.

The function accepts the following three arguments.

- A string corresponding to the name of the item to execute. The name must have been defined using the `name` attribute of the `repositoryItem` tag.
- An Array of data values to use as parameter values when running the item. The array has the following structure:

```
var thisVar = new Object();
thisVar.value = "param_value";
thisVar.columnName = "param_name";
var linkedData = new Array(thisVar);
```

- An optional parameter specifying a target location for the item output. This follows the same rules as the `location` attribute of the `repositoryItem` tag. It may be the ID of a DIV, the name of an IFRAME or FRAME, **NEW* or **windowName*. If omitted, the default location from the `repositoryItem` is used.

The data value array many be specified in one of the following ways:

- As null (or omitted). In this case, any necessary values are retrieved using any `sourceLinkPrompt` tags defined for the item.

- As the link data from actionHandler. The `actionHandler` tags define functions to call and the parameter values which are captured as part of the event. Those parameters can be passed directly to the `runRepositoryItem` API.
- As user defined. The JavaScript calling the `runRepositoryItem` API can define the values of the array as necessary. The *columnName* is the name of the column defined in the `sourceLinkVariable` tag. The value is the value to pass as the parameter.

The getBookmarkedValues function

The `getBookmarkedValues` function retrieves the values of cells that have been bookmarked in a report.

The id attribute of the cell should be set to the bookmark value. This function can be used for linking complex reports involving crosstabs and nested tables.

The `getBookmarkedValues` function accepts the following arguments:

- A parent node in the DOM of the report that the function needs to traverse to get the cell values matching the items specified in the bookmarks array
- An array of bookmarks defined in the report whose values are needed. For example, [“bookid1”,”bookid2”].

The function returns an array of data values to use as parameter values when running the report. The array has the following structure:

```
var thisVar = new Object();
    thisVar.value = "cell_value";
    thisVar.columnName = "bookmark";
```

The *columnName* is the name of the bookmark. The *value* is the value of the specified cell that is bookmarked.

The retrievePromptValues function

The `retrievePromptValues` function should be called when using parameters with custom controls, and supports both cascading and non-cascading prompts.

Call this function in the body `onLoad` handler to load the initial values of the prompt (or the parent prompt in the case of a cascading prompt). Call this function in the `onChange` handler of the control used to define the cascading parameter. The function will make calls to the server and get the prompt values to fill the parameter controls with updated values depending on the parent parameter value selected.

The `retrievePromptValues` function accepts the following arguments:

- A string denoting the name of the report containing the definitions of the cascading parameters. The name must have been defined as the `name` attribute of a `repositoryItem` tag.
- A string corresponding to the name of the parameter in the report. For cascading parameters, this string is the name of the cascading parameter group. The cascading group must be present in the report.
- A user-defined function that accepts an array of value and display text for the new options. The array can be null, in which case the function should clear the control. This function will be called by `retrieveCascadingPromptValues` to populate the parameter controls with new values.

```
function callback(options) {
   // logic to clear the control
   // logic to add value and display text to control
   for(var i = 0; i <options.length; i++) {
      control.value = options[i].value;
      Display Text for control = options[i].displayText;
   }
}
```

- An array of the selected preceding values present in the cascading group. This array is only needed for cascading parameters and should be omitted for a non-cascading parameter. The parameters must be in sequential order. To get the list of the parent cascading parameter, specify preceding values:

```
var precedingvals= new Array();
```

The preceding values array has the following structure. For example, to get the list of cities in MN:

```
precedingvals= new Array();
precedingvals[0]= "USA";
precedingVals[1]="MN";
```

IBM SPSS Collaboration and Deployment Services Tag Library tag reference

The various tags included in the IBM SPSS Collaboration and Deployment Services Tag Library are dependent on each other and, for validation purposes, need to know that the references are correctly met. The tags must also be defined in the correct sequence.

The following sections describe each available tag in detail.

This tag library depends upon JSP 1.2.

The credential tag

The `credential` tag defines both a data source login credential and a repository login credential.

The credential is referenced by name for all items and prompts defined on the page. It should be defined before any tags that may reference the credential. In normal use, it would be the first tag referenced in the JSP.

The `credential` tag can contain `properties` elements. For, example, in the case of J. D. Edwards (JDE) enabled data sources, the `credential` looks like the following:

```
<credential>
   <properties>
      <property name="JDE_LIBRARY_LIST_SELECTED" value="liblist_name"/>
   </properties>
   </credential>
```

Table 4. Attributes for the credential tag

Name	Required	Description
name	true	Either an internal name for the IBM SPSS Collaboration and Deployment Services Repository credential or the name of a data source used in a repository item object. This is used to link items and prompts to this credential and to satisfy any data source logins required by referenced items. • For repository credentials, this must match the name provided on the `repositoryCredentialName` attribute of the `repositoryItem` tag. • For database credentials, the name must match the data source name as it is referenced by the item using that data source. This name is used to store the credential values in a session variable. All credentials must have a unique name.

Table 4. Attributes for the credential tag (continued)

Name	Required	Description
`useSSO`	false	Indicates if Single Sign On credential for Kerberos should be used. If this attribute is set to true, then the `username`, `password`, and `provider` attributes must not be specified. When using SSO, the Authentication Filter must be configured in the web.xml file.
`credentialDefinitionName`	false	The name of a credential defined as a resource in the repository. If this value is specified, the `username`, `password`, and `provider` attributes do not need to be defined as the credential resource includes this information.
`provider`	false	For repository credentials, this is the optional security provider name. Valid values include: • *Native* for the built-in provider • *AD_<name>/<domain>* for Active Directory, where *<name>* corresponds to the security provider name within the system and *<domain>* corresponds to the DNS namespace • *ADL_<name>/<domain>* for Active Directory with local override, where *<name>* corresponds to the security provider name within the system and *<domain>* corresponds to the DNS namespace • *iSeries* for IBM i • *ldap_<name>* for OpenLDAP, where *<name>* corresponds to the security provider name within the system If not specified, the built-in repository security provider is used. This attribute is ignored for database credentials.
`username`	false	The user name to use for authentication.

Table 4. Attributes for the credential tag (continued)		
Name	**Required**	**Description**
password	false	The password for the specified username. The password is used internally by the tag library. It is NOT written to the JSP result.

Tag nesting

None

Expected output

None. This tag provides authentication information. The tag does not produce output, but caches the credentials using the name attribute as a key for later use with a report or prompt tag.

Sample usage

The following sample specifies three credentials. This first is for accessing the IBM SPSS Collaboration and Deployment Services Repository with a specified username and password. The value of *Native* for *provider* indicates that the username/password pair for validation is defined in the native local security provider. The second credential employs single sign-on for the IBM SPSS Collaboration and Deployment Services Repository using the user's previously authenticated credentials. The third credential is for a data source named *Northwind*.

```
<r:credential name="repositoryCredential" provider="Native"
   username='admin' password='password'/>

<r:credential name="repositorySSO" useSSO="true" />

<r:credential name="Northwind" username='sa' password='sa'/>
```

The repositoryItem tag

The repositoryItem tag is the main tag for defining repository item definitions that will be used by the application.

The repositoryItem tag may reference visualization reports, jobs, scoring items, or SPD files. The repository items may be run directly, used to provide prompts, or ran programmatically.

Any sourceLinkPrompt and sourceLinkRepositoryItem tags should be nested within the repositoryItem tag as follows:

- Use a nested sourceLinkRepositoryItem tag if this item will be run when the user clicks on a different item.
- Use sourceLinkPrompt when the parameter values will come from prompts defined on the page or defined directly in the item.

You may optionally specify additional properties that are specific to a type of repository item. The property names must be in lowercase for them to work in the Firefox browser. These property values will be passed to the URL to run the repository item. The properties are specified as a nested XML block.

Table 5. Attributes for the repositoryItem tag		
Name	**Required**	**Description**
name	true	Defines a unique name for the item. The name can then be referenced by other tags or via the runRepositoryItem() JavaScript API.

Table 5. Attributes for the repositoryItem tag (continued)

Name	Required	Description
`inputURI`	true	The item definition to be used to render the report output. This value must specify a URI that may be used to locate the item definition. The following URI schemes are supported: • *file*: References a specific file on the application server or a network file location • *spsscr*: References a file in the IBM SPSS Collaboration and Deployment Services Repository. This scheme allows files to be referenced by identifier or hierarchical path within the repository. Specific version markers can be specified. If no version or label is specified, the latest version is used. • *scoring*: References a model configuration from the repository. Scoring configurations are referenced by name from the tag libraries. If a scoring configuration is renamed, the tag library reference must also be modified.
`activate`	true	Specifies when the item will be activated. Options include: • *ONDEMAND*: Runs the item when activated by a row-click for a source report. • *ONLOAD*: Runs the item when the page initially loads. • *NONE*: Item does not run automatically. In this case, the item is used to provide prompts or prompt values. Regardless of the activate setting, any report may be run programmatically by using the public JavaScript `runRepositoryItem()` API.

Table 5. Attributes for the repositoryItem tag (continued)		
Name	**Required**	**Description**
`location`	false	The destination for the output resulting from running the item. The usage varies slightly depending on what the target type is. • For DIV targets, the location should specify the ID of the DIV tag where the output is to be placed. • For IFRAME targets, the location must specify the name of the frame. • To open the output in a new window, specify a location of `*NEW`. • To direct the output to a named window, use an asterisk (`*`) followed by the window name. For example, `*MYWINDOW` will open a new window called *MYWINDOW* and will reuse that window on each activation of the link.
`repositoryCredentialName`	true	The name of the credential that should be used when accessing the item from the repository and running the item via the reporting service. The credential should have been previously defined using the `credential` tag.
`outputType`	false	The type of output to generate. The supported output types vary by item type. Normally this will be either HTML or PNG but other options include: **Visualization Reports:** *PNG*, *EMF*, *JPEG*, *HTML* If not specified, the output will default to HTML (or PNG for visualization reports). To display reports using the viewer, specify a type of *RPTDocument*. For this type, the target must be an IFRAME.

Table 5. Attributes for the repositoryItem tag (continued)		
Name	**Required**	**Description**
`width`	false	This is the width of the output image. The width must be greater than 0 and specified in conjunction with the height. If not specified, the default width and height are used.
`height`	false	This is the height to use when output is an image. The width must also be specified or the setting will have no effect. The value must be greater than 0.

Tag nesting

This tag may include one `sourceLinkRepositoryItem` and multiple `sourceLinkPrompt` and `outputLocation` tags.

The repositoryItemPrompt tag

The `repositoryItemPrompt` tag generates the HTML for a prompt variable that is defined in the referenced item.

The item that the prompt is referencing must be defined using the `repositoryItem` tag before this tag can be used. Use this when you want prompt controls such as those used in IBM SPSS Collaboration and Deployment Services Deployment Portal to be used in your application.

This tag generates the HTML prompt controls in the location corresponding to where the tag is used. The tag must be associated with a particular parameter of an item to be useful. The association with parameters is done using the `sourceLinkPrompt` tag, where the `promptID` of the `sourceLinkPrompt` must match the `promptID` of this tag.

Table 6. Attributes for the repositoryItemPrompt tag		
Name	**Required**	**Description**
`promptId`	false	A unique identifier that can be referenced from the `promptId` attribute of the `sourceLinkPrompt` tag.
`repositoryItemName`	true	A reference to the name of the item as defined in the `name` attribute of the `repositoryItem` tag.
`parameterName`	false	Name of the prompt variable as defined in the item.

Tag nesting

None

Expected output

An HTML element that allows the user to select or type personal values depending on the `promptType`, which is selected as `parameterName`. The `repositoryItemPrompt` tag supports all parameters IBM SPSS Collaboration and Deployment Services Deployment Portal supports. As a result, all types of prompts are supported and the appropriate HTML element is generated.

Sample usage

The following sample prompts for a value for the *EmployeeID* parameter in the *Employees* report.

```
<repositoryItem name="Employees"
   inputURI="file:///d:/yourDS/ReportTaglib/Employees.dbq"
   repositoryCredentialName="localhost" activate="NONE" />

<repositoryItemPrompt promptId="EmployeeIdPrompt"
   repositoryItemName="Employees" parameterName="EmployeeID" />
```

The report tag

This tag is deprecated. Use the `repositoryItem` tag instead.

The reportPrompt tag

This tag is deprecated. Use the `repositoryItemPrompt` tag instead.

The outputLocation tag

This tag associates the generated output that exists in the repository with the location on the page where the output is displayed. When the item runs, the output is retrieved from the repository and displayed at the specified target location on the page.

This tag must always be nested within a `repositoryItem` tag.

Table 7. Attributes for the outputLocation tag

Name	Required	Description
`outputId`	false	This is the path to the output that exists in the repository. For custom dialogs, this attribute should be omitted. The output from running the syntax is automatically detected.

Table 7. Attributes for the outputLocation tag (continued)		
Name	**Required**	**Description**
`location`	true	This attribute specifies where the output should be placed on the page. • For DIV targets, the location should specify the ID of the DIV tag where the output is to be placed. • For IFRAME targets, the location must specify the name of the frame. • To open the report output in a new window, specify a location of `*NEW`. • To direct the output to a named window, use an asterisk (*) followed by the window name. For example, `*MYREPORTS` will open a new window called *MYREPORTS* and will reuse that window on each activation of the link. HTML outputs may target a DIV. All other outputs should target an IFRAME or a window.
`partId`	false	This is used to identify the specific part or item of the SPW archive output. This functionality requires IBM SPSS Statistics adapters in the IBM SPSS Collaboration and Deployment Services environment. For more information, see the IBM SPSS Statistics installation documentation.

Tag nesting

None

Sample usage

The following sample specifies an output location for a chart stored in the IBM SPSS Collaboration and Deployment Services Repository using the *ChartFRAME* IFRAME tag.

```
<outputLocation outputId="spsscr:///output/output_chart.png"
   location="ChartFRAME"/>
```

If the attribute values depend on parameter values, use the *sourceLinkPrompt* tag to define matches for the parameters. If a match is found, it is substituted for the parameter. For example, the following sample defines two *outputLocation* tags with file names that depend on parameters.

```
<repositoryItem name= “Call_Center_Score”
   inputURI= ”spsscr:///job/Call Center”
   repositoryCredentialName=”localhost”
   activate=”ONDEMAND”/>
   <outputLocation outputId =”spsscr:///output/output_tab_${JobParam1}.png”
      location=”ChartFRAME”/>
   <outputLocation outputId=”/output/output_chart_${JobParam2}.html”
      location=”ReportDIV”/>
   <sourceLinkPrompt promptId=”JobParam1” parameterValue=”Jan” />
   <sourceLinkPrompt promptId=”JobParam2”
      targetNameParameter=”html_id_for_the_value” />
</repositoryItem>
```

For *JobParam1*, a value of *Jan* is substituted in the name, resulting in *output_chart_Jan.png* appearing at *ChartFRAME*.

For *JobParam2*, the value associated with the html control for the parameter is substituted in the name. If that value is *Illinois*, the file *output_tab_Illinois.html* appears at *ReportDIV*.

The sourceLinkPrompt tag

The `sourceLinkPrompt` tag associates the item parameters with the prompts providing their values. These could be user defined HTML elements, javaScript functions, prompts created using the `repositoryItemPrompt` tag, or directly specified values.

The `sourceLinkPrompt` tag must always be nested within a `repositoryItem` tag. When the item runs, the parameter values are retrieved using the `sourceLinkPrompts`.

Table 8. Attributes for the sourceLinkPrompt tag

Name	Required	Description
`targetNameParameter`	true	Name of the parameter as it is defined in the repository item. For scoring models that use multiple tables, specify the value as *table.parameter* where *table* is the table name and *parameter* is the parameter name.
`promptId`	false	The `promptId` could be the ID of a `reportPrompt` tag or the name of an HTML control. When a prompt value is needed, the `reportPrompt` or the HTML control will be used to determine the prompt value. Either `promptId`, `parameterValue` or `getValueJSFunction` should be specified.

Table 8. Attributes for the sourceLinkPrompt tag (continued)		
Name	**Required**	**Description**
`parameterValue`	false	Specifies a value for the parameter instead of prompting for one. This should be specified when the application knows the parameter value when the JSP is being processed. In that case, the value can be specified directly using this attribute. If `parameterValue` is specified, then `promptId` and `getValueJSFunction` should not be used.
`getValueJSFunction`	false	Identifies a function to call to retrieve the prompt value(s). The function should return either a single value or return an array of values. This attribute should include the function name, parentheses and any parameters as necessary. For example, for a function called *MyGetValues* that takes one parameter, set the attribute to `MyGetValues('myPromptID')`.
`validateJSFunction`	false	Identifies a function to call to provide validation of the prompt. The function should return `true` if the prompts are valid. This attribute should include the function name, parentheses and any parameters as necessary. For example, for a function called *MyValidate* that takes one parameter, set the attribute to `MyValidate('myPromptID')`.

Tag nesting

None

Validations performed

None

Expected output

None

Sample usage

The following report sample prompts for two parameter values using `repositoryItemPrompt` tags. The `sourceLinkPrompt` tags for the *CountrySales* report use the identifiers for those prompts to supply their values to the report.

```
<r:repositoryItem name="CountrySales"
   reportDefinitionURI="spsscr:///rpts/CountryCity_cascadingParameter.rptdesign"
   repositoryCredentialName="repositoryCredential"
   outputType="HTML" activate="ONDEMAND" location="ReportDIV">
   <r:sourceLinkPrompt targetNameParameter="ShipCountry" promptId="IDFilter"/>
   <r:sourceLinkPrompt targetNameParameter="ShipCity" promptId="IDFilter1"/>
</r:repositoryItem>

<table width="95%" cellspacing="1" bgcolor="black">
   <tr bgcolor="white">
      <r:repositoryItemPrompt promptId="IDFilter"
       repositoryItemName="CountrySales1" parameterName="ShipCountry"/>
   </tr>
   <tr bgcolor="white">
      <r:repositoryItemPrompt promptId="IDFilter1"
       repositoryItemName="CountrySales1" parameterName="ShipCity"/>
   </tr>
</table>
```

A similar approach can be used for scoring models. The following scoring sample prompts for five parameter values using `input` tags. The `sourceLinkPrompt` tags for the *Configuration* item use the identifiers for those prompts to supply their values.

```
<table>
   <tr>
      <td>Age</td>
      <td><input name="Age" id="Age" type="text"/></td>
   </tr>
   <tr>
      <td>Blood Pressure</td>
      <td><input name="BP" id="BP" type="text"/></td>
   </tr>
   <tr>
      <td>Cholesterol</td>
      <td><input name="Cholesterol" id="Cholesterol" type="text"/></td>
   </tr>
   <tr>
      <td>K</td>
      <td><input name="K" id="K" type="text"/></td>
   </tr>
   <tr>
      <td>Na</td>
      <td><input name="Na" id="Na" type="text"/></td>
   </tr>
</table>

<r:repositoryItem name="MyConfiguration" inputURI="scoring:///KMeans"
   repositoryCredentialName="repositoryCredential" outputType="HTML"
   activate="ONDEMAND" location="ReportIframe">
   <r:sourceLinkPrompt targetNameParameter="Age" promptId="Age"/>
   <r:sourceLinkPrompt targetNameParameter="BP" promptId="BP"/>
   <r:sourceLinkPrompt targetNameParameter="Cholesterol"
     promptId="Cholesterol"/>
   <r:sourceLinkPrompt targetNameParameter="Drug" parameterValue="DrugX"/>
   <r:sourceLinkPrompt targetNameParameter="K" promptId="K"/>
   <r:sourceLinkPrompt targetNameParameter="Na" promptId="Na"/>
</r:repositoryItem>
```

The value for the *Drug* parameter is specified within the page as *DrugX* by using the `parameterValue` attribute.

If the parameters used by the scoring configuration are defined in separate tables, prefix the parameter name with the table name, separating the two using a period. For example, if the *Age* and *BP* parameters are defined in *Table1* and the remaining parameters are defined in *Table2*, the `sourceLinkPrompt` elements would be specified as follows:

```
<r:repositoryItem name="MyConfiguration" inputURI="scoring:///KMeans"
   repositoryCredentialName="repositoryCredential" outputType="HTML"
   activate="ONDEMAND" location="ReportIframe">
   <r:sourceLinkPrompt targetNameParameter="Table1.Age" promptId="Age"/>
   <r:sourceLinkPrompt targetNameParameter="Table1.BP" promptId="BP"/>
   <r:sourceLinkPrompt targetNameParameter="Table2.Cholesterol"
     promptId="Cholesterol"/>
   <r:sourceLinkPrompt targetNameParameter="Table2.Drug" parameterValue="DrugX"/>
   <r:sourceLinkPrompt targetNameParameter="Table2.K" promptId="K"/>
   <r:sourceLinkPrompt targetNameParameter="Table2.Na" promptId="Na"/>
</r:repositoryItem>
```

The sourceLinkRepositoryItem tag

The `sourceLinkRepositoryItem` tag identifies the source item and variables used to satisfy the item's defined parameters. Using this mechanism, when the source item is clicked, the parent item runs using the parameters defined within the nested `sourceLinkVariable` tags.

This tag must always be nested within a `repositoryItem` tag. It should contain one or more nested `sourceLinkVariable` tags.

Table 9. Attributes for the sourceLinkRepositoryItem tag

Name	Required	Description
`sourceName`	true	Name of the `repositoryItem` that will serve as the source of the relationship
`linkType`	false	Determines what action on the source report will trigger the running of the current report. Currently there is only one supported `linkType`, *row*. For this type, when a row in the source report is clicked, the target report runs. In future releases, additional linkTypes may be added.

Tag nesting

The `sourceLinkRepositoryItem` tag contains one or more `sourceLinkVariable` tags that identify the source column and the target parameter names.

Expected output

None

Sample usage

The following sample identifies *CityDetails* as the report to run in response to a user action in the *AllCountries* report.

```
<r:repositoryItem name="CityDetails"
   inputURI="spsscr:///SampleReports/BIRT/CountrySalesByCity.rptdesign"
   repositoryCredentialName="repositoryCredential"
   outputType="HTML" width="400" height="300"
   activate="ONDEMAND" location="SecondReportDIV">
   <r:sourceLinkRepositoryItem sourceReportName="AllCountries">
      <r:sourceLinkVariable columnName="ShipCountry"
         targetNameParameter="ShipCountry" />
   </r:sourceLinkRepositoryItem>
</r:repositoryItem>
```

The sourceLinkReport tag

This tag is deprecated. Use the `sourceLinkRepositoryItem` tag instead.

The sourceLinkVariable tag

The `sourceLinkVariable` tag defines the mapping between the variable or column to use in the source item and the parameter as defined in the target item. This tag must always be nested under a `sourceLinkRepositoryItem` tag.

Table 10. Attributes for the sourceLinkVariable tag

Name	Required	Description
`columnName`	true	For Visualization reports, this attribute contains the id of the `sourceVariable` or `derivedVariable` element of the Visualization specification. Currently only categorical variables are supported.
`targetNameParameter`	true	Name of the parameter in the target query

Tag nesting

None

Validations performed

None

Expected output

None

Sample usage

The following sample maps the *ShipCountry* variable in the *AllCountries* report to the *ShipCountry* parameter in the *CityDetails* report.

```
<r:repositoryItem name="CityDetails"
   inputURI="spsscr:///SampleReports/Vis/CitiesBarChart.viz"
   repositoryCredentialName="repositoryCredential"
   outputType="png" width="400" height="300"
   activate="ONDEMAND" location="SecondReportDIV">
   <r:sourceLinkRepositoryItem sourceName="AllCountries">
      <r:sourceLinkVariable columnName="ShipCountry"
        targetNameParameter="ShipCountry"/>
   </r:sourceLinkRepositoryItem>
</r:repositoryItem>
```

The actionHandler tag

Defines the action handlers that should be applied to the item. When action handlers are defined, the automatic linking setup using `sourceLinkRepositoryItem` no longer applies.

The application builder is responsible for running any target items using the `runRepositoryItem` public Java script API.

Table 11. Attributes for the actionHandler tag		
Name	**Required**	**Description**
`event`	true	The event name. Valid events include: • *onclick* • *onmouseover* • *onmouseout*
`function`	true	The name of the Java Script function to call when the event occurs. This should be the function name only, without () or any parameters.
`partId`	false	This is used to identify the specific part of the report that the actions should apply to.

Tag nesting

Any data values that need to be passed as parameters to the JavaScript function should be defined using nested `actionParameter` tags.

Sample usage

The following `repositoryItem` tag defines three action handlers, one for each type of event that could occur. Each handler calls a unique JavaScript function that defines the subsequent processing.

```
<r:repositoryItem name="AllCountries"
   inputURI="spsscr:///SampleReports/BIRT/CountrySales.rptdesign"
   repositoryCredentialName="repositoryCredential"
   outputType="HTML"
   width="400" height="300"
   activate="ONLOAD" location="ReportDIV">
   <r:actionHandler event="onclick" function="myOnClick">
      <r:actionParameter name="ShipCountry"/>
   </r:actionHandler>
   <r:actionHandler event="onmouseover" function="myOnOver">
      <r:actionParameter name="ShipCountry"/>
   </r:actionHandler>
   <r:actionHandler event="onmouseout" function="myOnOut" />
</r:repositoryItem>
```

The actionParameter tag

There should be an `actionParameter` for each data value from the item that needs to be passed to the `actionHandler` JavaScript function. This tag must be nested within the `actionHandler` tag.

Table 12. Attributes for the actionParameter tag

Name	Required	Description
name	true	Name of the column or variable that defines which value from the report results should be passed to the function. For visualization reports, the name is the `id` attribute of the `sourceVariable` or `derivedVariable` element. Currently only categorical variables are supported.

Tag nesting

None

Sample usage

The following sample defines an `actionParameter` named *ShipCountry* that gets passed to the JavaScript function `myOnClick` when the user clicks the report.

```
<r:actionHandler event="onclick" function="myOnClick">
   <r:actionParameter name="ShipCountry"/>
</r:actionHandler>
```

Tag library beans

The framework includes tag library beans that can be used together for a variety of purposes. For example, the beans can be used to retrieve a data set that can then be used to build custom HTML controls.

In order to use the beans, you must first declare references to them in the JSP. This is done through the `import` attribute of the page directive.

```
<%@ page contentType="text/html;charset=utf-8"
     language="java"
     session="true"
     import="java.util.Map"
     import="java.util.HashMap"
     import="com.spss.report.taglib.bean.ReportBean"
     import="com.spss.report.taglib.bean.Credential"
%>
```

The code samples for beans use the JavaServer Pages Standard Tag Library (JSTL) which should be included using the `taglib` directive.

```
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
```

For more information on JSTL, refer to the Sun documentation.

Credential bean

The `Credential` bean defines the credentials that will be used by other beans.

The following code sample creates two credentials and stores them in a `HashMap`. In the following sample, the *localhost* credential provides the logon information for the IBM SPSS Collaboration and

Deployment Services Repository. The *ps4008* credential is for a SQL Server data source called ps4008 that is referenced by the report definition.

```
<%
Map credentialMap = new HashMap();
Credential repositoryCredential =
  new Credential("localhost","Native","admin","spss",null);
Credential datasourceCredential = new Credential("ps4008",null,"sa","sa",null);
credentialMap.put("localhost",repositoryCredential);
credentialMap.put("ps4008",datasourceCredential);
%>
```

ReportBean bean

The ReportBean is used to retrieve the data for a data set that is defined in a report definition.

The following code uses the previously created credentialMap to retrieve a data set. Visualization reports do not support this function.

```
<%-- Creating JavaBeans --%>
<jsp:useBean id="report" class="com.spss.report.taglib.bean.ReportBean">
   <jsp:setProperty name="report" property="reportDefinitionURI"
      value="file:///d:/SPSS/ps4008/Test.dbq" />
   <jsp:setProperty name="report" property="repositoryCredentialName"
      value="localhost" />
   <jsp:setProperty name="report" property="host" value="localhost" />
   <jsp:setProperty name="report" property="port" value="8080" />
   <jsp:setProperty name="report" property="dataSetName"
      value="DataSet1" />
   <jsp:setProperty name="report" property="credentialMap"
      value="<%=credentialMap%>" />
</jsp:useBean>
```

The properties used in this code are the following:

- The *reportDefinitionURI* property specifying the location of the report
- The *repositoryCredentialName* property identifying the host
- The *port* property defining the port
- The *dataSetName* property specifying the name of the data set.
- The *credentialMap* property defining a reference to a HashMap containing the credentials to use

The ReportBean can then be run to return the data set. The data can be used to generate a list control as shown in the following code.

```
<SELECT style="WIDTH :250 px" ID="EmployeeID_Prompt" NAME="EmployeeID_Prompt"
   TABINDEX="2">
<c:forEach var="row" items="${report.rows}">
   <c:forEach var="column" items="${row.columns}">
      <c:if test='${column.name == "EmployeeID"}'>

         <OPTION VALUE='<c:out value="${column.value}" />'>
            <c:out value="${column.value}" />
         </OPTION>
      </c:if>
   </c:forEach>
</c:forEach>
</SELECT>
```

SearchBean bean

The SearchBean bean provides a query mechanism for locating content in the repository that meet specified criteria.

For example, the bean can retrieve a list of IBM SPSS Statistics data file (.sav) sources in the IBM SPSS Collaboration and Deployment Services Repository that match a specified search criterion. The following code defines bean properties to query for all IBM SPSS Statistics data sources using the MIME types associated with those sources.

```
<jsp:useBean id="data_sources"
   class="com.spss.report.taglib.bean.SearchBean" scope="page">
   <jsp:setProperty name="data_sources" property="request"
    value="<%= request %>" />
   <jsp:setProperty name="data_sources" property="credentialName"
      value="AuthenticationCredential" />
   <jsp:setProperty name="data_sources" property="searchQuery"
      value="<%=  "('$$search/mimetype'='application/x-vnd.spss-spss-data' or "
         + "'$$search/mimetype'='application/x-vnd.spss-statistics-data' )" %>" />
</jsp:useBean>
```

The properties used in this code are as follows:

- The *request* property defining an `HttpServletRequest` object.
- The *credentialName* property specifying the credential needed to connect to repository. In this case, the value corresponds to the credential *AuthenticationCredential* defined using the `credential` tag.
- The *searchQuery* property specifying a string denoting the search criterion.

The `SearchBean` can then be run to return the matching data sources. The following code presents the name, modification date, version label, and author metadata for the data sources in a table.

```
<Table  border="0" height="100%" width="100%" cellpadding="0" cellspacing="0">
   <tr>
      <td align="center" bgcolor="#EEEEEE">
         Data Source
      </td>
      <td align="center" bgcolor="#EEEEEE">
         Modified Date
      </td>
      <td align="center" bgcolor="#EEEEEE">
         Version Label
      </td>
      <td align="center" bgcolor="#EEEEEE">
         Author
      </td>
   </tr>
   <c:forEach var="data_source" items="${data_sources.records}"
    varStatus="status" begin="0" end="3" step="1">
      <tr>
         <td align="center" bgcolor="#EEEEEE">
            <c:out value="${data_source.title}" />
         </td>
         <td align="center" bgcolor="#EEEEEE">
            <c:out value="${data_source.modifiedDate}" />
         </td>
         <td align="center" bgcolor="#EEEEEE">
            <c:out value="${data_source.versionLabel}" />
         </td>
         <td align="center" bgcolor="#EEEEEE">
            <c:out value="${data_source.author}" />
         </td>
      </tr>
   </c:forEach>
</Table>
```

ScoringBean bean

The `ScoringBean` bean retrieves a list of scoring configurations for a specified model that are able to respond to a scoring request.

The `getScoringConfigurations` method of the bean accepts the following parameters:

- The *credential* parameter specifying the credentials for accessing the IBM SPSS Collaboration and Deployment Services Repository defined using the `Credential` bean.
- The *modelLocationUri* parameter identifying the URI for a model in the IBM SPSS Collaboration and Deployment Services Repository.

Alternatively, instead of supplying a `Credential` bean item, the following two parameters can be used for specifying credentials:

- The *request* parameter specifying an `HttpServletRequest` object.
- The *credentialName* parameter defining the credential needed to connect to the IBM SPSS Collaboration and Deployment Services Repository defined using the `credential` tag.

The following code retrieves the scoring configurations for the model *KMeans.xml* that can respond to a scoring request using a credential defined using the `credential` tag:

```
<r:credential name="repositoryCredential" provider="Native"
   username='<%= request.getParameter("userid")%>'
   password='<%= request.getParameter("password")%>' />

<%
   String[] configurations = ScoringBean.getScoringConfigurations(request,
      "repositoryCredential", "spsscr:///Sample/KMeans.xml");
%>
```

The array returned by the bean can be used to populate a form from which a user can select a scoring configuration to use for subsequent scoring.

```
<form id="selectConfigurationForm" target="ScoringIframe" method="POST">
   <div style="display:none">
```

```
      <input name="userid" type="text"
    value="<%= request.getParameter("userid")%>"/>
      <input name="password" type="text"
    value="<%= request.getParameter("password")%>"/>
   </div>
   Select Scoring Configuration:
   <select name="selectedConfiguration"
    onchange="onSelectConfiguration(this)">
      <option></option>
      <%
         for (int i=0; i < configurations.length; i++)
         {
      %>
            <option value="<%= configurations[i].replaceAll("[ ]", "%20")%>">
               <%= configurations[i] %></option>
      <%
         }
      %>
   </select>
</form>
```

JavaServer Pages samples

IBM SPSS Collaboration and Deployment Services includes a variety of JSP samples illustrating the use of the tag library.

The samples are grouped into the following categories:

- **Reporting.** Using visualization reports interactively, including running a second report in response to a selection. To access these samples, go to:

  ```
  http://<server-name>:<port>/IBMSPSSTagLib/index.html
  ```

- **Scoring.** Generating scores for a predictive model configured for scoring, including a variety of approaches to supplying configured models with data for scoring. To access these samples, go to:

```
http://<server-name>:<port>/scoringTagLib/index.html
```

If the URL for a set of samples fails to return an introduction page, the war file containing the samples may not be deployed to the IBM SPSS Collaboration and Deployment Services Repository server. The war files to be deployed and necessary sample files are under the `./components/taglib/Samples/TagLib` directory of the repository installation. Deploy the war files in accordance with the documentation for your application server.

Note: To avoid conflicts with other applications running on your server, you can specify a custom context root for the deployed samples. For more information, see the documentation for your application server.

On the introduction page for the samples, click **View Source** for any sample to examine its source code. To explore their functionality, you can run the samples from the page by clicking **Run**. However, successful execution requires the following:

- Sample resources in a specific folder structure in the IBM SPSS Collaboration and Deployment Services Repository
- Valid credentials for accessing the resources referenced in the samples

Instructions for configuring the environment for successful sample execution are available from the introduction page for the samples.

Chapter 4. Portal integration

The IBM SPSS Collaboration and Deployment Services Web services architecture provides the ability to integrate it with portal servers. This enables delivery of highly customized content through pluggable user interface components that use Web services to produce fragments of markup code that are aggregated into a portal page.

Typically, a portal page is displayed as a collection of non-overlapping windows, where each window displays a segment of content. Some examples of portal applications are email, weather reports, discussion forums, and news. Similarly, IBM SPSS Collaboration and Deployment Services portals can be used to deliver customized content, such as output of reports and analytical processing, charts, diagrams, etc.

The repository supports portal integration based on JSR 168 standard. JSR 168 proposed by Java Community Process group (http://jcp.org), enables interpretability for portlets between different Web portals. This specification defines a set of APIs for interaction between the portlet container and the portlet, addressing the areas of personalization, presentation and security. Implementation of JSR 168 include IBM Web Portal from WebSphere, Oracle Application Server Portal 10g, Vignette Portal, Sun Portal Server, and JBoss.

The repository also supports portal integration with Microsoft SharePoint server using Web Parts.

Officially supported portal environments include:

- WebSphere Portal Server 6.1
- GateIn 3.5.0 (JBoss AS7)
- Sun Java Enterprise System 5
- Microsoft Sharepoint 2010 Server
- Microsoft Sharepoint 2007 Server

The repository may also be integrated with other portal environments based on JSR 168 and J2SE 5.0.

IBM SPSS Collaboration and Deployment Services Portlet and IBM SPSS Collaboration and Deployment Services Web Part can be used to deliver repository content to portal users. The architecture also enables creation of custom JSR 168-compliant portlets and SharePoint Web Parts that use IBM SPSS Collaboration and Deployment Services Web services.

Restriction: To use portal integration for accessing repository content, your browser must allow cookies.

Installation

After downloading the IBM SPSS Collaboration and Deployment Services installation files, the portal components are in the `/PORTLET` directory of the repository installation download. They include `IBMSPSSPortlet.war` (portlet) and `IBMSPSSWebPart.wsp` (Web Part).

IBM SPSS Collaboration and Deployment Services Portlet installation

The procedure for installing `IBMSPSSPortlet.war` varies depending on the portal server type. Refer to portal server vendor documentation for details.

IBM SPSS Collaboration and Deployment Services Web Part installation

SharePoint Web Part installation prerequisites include:

- Microsoft SharePoint 2007
- Microsoft Web Service Enhancement 2.0 (WSE 2.0 SP3)

To install IBM SPSS Collaboration and Deployment Services Web Part:

1. Copy `IBMSPSSWebPart.wsp` from the repository installation download to a predefined location on the SharePoint host, for example, `c:\temp`.
2. From the `/bin` directory of the SharePoint server installation run the following commands:

```
stsadm -o addsolution -filename c:\tmp\IBMSPSSwebpart.wsp
stsadm -o deploysolution -name IBMSPSSwebpart.wsp -immediate
    -allowgacdeployment -url http://<hostname>
```

3. Use SharePoint administration utilities to add the Web Part to the Web Part gallery and to subsequently deploy it. For more information, see Microsoft SharePoint documentation.

Once the component has been installed, it must be configured to access a specific resource in the repository. Component preferences must also be set up.

Configuration

After the portal component has been installed and the portal page layout has been completed, you will be prompted to configure the component to access a repository resource. The general procedure for configuring portal access involves defining the repository server, specifying repository credentials, selecting the resource to be delivered to the portal, and if necessary, specifying data source credentials and default prompt values. You can also configure components' appearance and behavior by setting the preferences.

Configuring IBM SPSS Collaboration and Deployment Services Portlet

Open the portlet configuration page. The page may open differently depending on the portal server type.

1. Specify the repository server URL.
2. Specify the repository user credentials and security provider for login authentication.
3. Select the repository resource to be delivered to the portal. Make sure the correct resource version is specified.
4. If necessary, specify the credentials for the data source referenced by the resource; for example if a report uses a database, database credentials must be provided. Note that depending on the resource, it may be necessary to specify credentials for multiple data sources
5. If the resource includes prompts (for example, a report may allow for a dynamic selection of values), specify the default prompt settings.
6. Verify that configuration information is correct. To start over, click **Refresh**.
7. Click **Next** to proceed to viewing the resource.

Portlet settings can been edited after the initial configuration: for example, it can be pointed to a different repository resource if necessary.

Certain aspects of the appearance and behavior of the portlet are set through its preferences. The following preferences are available:

Preference	Description
expiration-cache	The expiration period for the portlet cache, i.e., the time in seconds after which the portlet output expires. -1 indicates that the output never expires. The default value is 600.
log-messages	Specifies whether portlet messages will be appended to the portal server log file. The default value is NO.
reenter-dsLogin	Specifies whether the user must provide the data source credentials for the portlet instance every time she logs in to the portal. The default value is NO.
reenter-parameter	Specifies whether the user must reenter the prompt values for the portlet instance every time she logs in to the portal. The default value is NO.

Preference	Description
refresh-parameter	Specifies whether the user can enter different parameter values and re-display the content based on those values. The default value is NO.
use-single-sign-on	Specifies whether the portlet will be used with single sign-on. The default value is NO.
validate-input-parameter	Enables user input parameter validation in order to protect against cross-site scripting attacks. The default value is YES.
window-height	The height of the portlet window (pixels). The default value is 750.
window-title	Descriptive name for the portlet instance.
window-width	The width of the portlet window (percent). The default value is 100%.

Preferences are set with portal server administration facilities and the way they are accessed will differ depending on the server type.

Configuring IBM SPSS Collaboration and Deployment Services Web Part

Web Part configuration involves the same basic steps as the portlet configuration: setting up access to the repository resource and configuration option. Note that the number of displayed items in the repository tree (when you select the resource) is controlled by an additional configuration option.

Single sign-on

IBM SPSS Collaboration and Deployment Services allows single sign-on access, and special configuration of the portal server may be required to enable it for the portlet or Web part. The procedures for enabling single sign-on will be different depending on the portal server. Consult your portal server documentation for more information.

Known issues

- When the portlet is used with JBoss portal, the repository tree view may not expand. In order to correct the problem, modify the *<JBoss installation folder>/bin/run.bat* (*run.sh* on UNIX) to increase the new generation and permanent generation size by adding the following arguments to JAVA_OPTS:

```
-XX:MaxNewSize=256m -XX:MaxPermSize=256m
```

- Cookie settings in the Safari browser may prevent some repository artifacts from being displayed in the portlet without first prompting for credentials. The browser cookie policy should be set to *Always* instead of *Only from sites I visit* to avoid repeated requests for credentials.

Chapter 5. HTML archive

An HTML report typically involves a number of HTML files displaying a variety of referenced images using style sheets to control the appearance of the output. Due to the number of files involved, managing and sharing this output can be a challenge. If one file is missing or incorrectly referenced, the pages do not display correctly.

The HTML Archive, or HTMLC, format addresses the issue of managing numerous intra-linked files by placing all associated HTML artifacts into a single, cross-browser archive file. The IBM SPSS Collaboration and Deployment Services Repository includes a viewer enabling a variety of client applications to display the contents of the archive. When accessing an HTMLC file stored in the repository, relative cross-references within the archive are silently replaced with full paths that reference the archive file. This allows links to files within the archive to resolve completely and display correctly.

File structure

An HTMLC archive file contains the following:

- A primary HTML file at the root of the archive. When rendering an HTMLC archive, the viewer uses the first file with an *.html* extension at the archive root as the primary file.
- Secondary files referenced by the primary file, such as cascading style sheets, images, javascript, or other HTML files. Secondary files can exist in any folder within the archive.

All references to files within the archive should use relative paths.

Creating HTMLC files

Custom HTMLC files can also be created using a file archiver such as the Java Archive tool or WinZip. To manually create an HTMLC file:

1. Create the structure for the files in the file system.
2. Create an archive containing those files and folders, specifying an extension of `.htmlc` for the output file.

The files in the archive may be created manually or automatically. In IBM SPSS Statistics, for example, you can export the results of an analysis as HTML. The resulting HTML and image files can be archived as an HTMLC file. Alternatively, you can use an HTML editor to manually create pages to be archived.

Custom HTMLC file example

For this example, consider a folder containing the file `gss.html` and the subfolders `css` and `images`. The HTML file references images contained in the `images` folder and uses styles contained in a cascading style sheet in the `css` folder. Using the Java Archive tool, the following command creates an HTMLC file named `custom.HTMLC` containing the files.

```
jar -cvfM custom.HTMLC gss.htm images css
```

Storing this single archive in the repository allows the `gss.html` page to be displayed in repository clients, such as the IBM SPSS Collaboration and Deployment Services Deployment Portal or IBM SPSS Deployment Manager, with its referenced graphics using the defined styles.

Chapter 6. Customization example

The Model Management page of IBM SPSS Collaboration and Deployment Services Deployment Portal provides the ability to monitor the ongoing performance of models deployed to IBM SPSS Collaboration and Deployment Services Repository. These model files are associated with jobs that can be executed on demand or scheduled. The files are created by using IBM SPSS Modeler. Model evaluation and champion challenger jobs are set up and executed using IBM SPSS Deployment Manager, and IBM SPSS Collaboration and Deployment Services Deployment Portal is used only to view the results. The information displayed as panels on the Model Management page can include the following:

- a listing of the best and worst performing models
- trends of model performance
- champion models
- a listing of all available model files

The options on the Configure panel can be used to specify display parameters and show or hide individual tabs.

For information on using the Model Management page, see the IBM SPSS Collaboration and Deployment Services Deployment Portal help system.

The user interface mainly consists of a single JavaServer Page (JSP), *MMDMaster.jsp*. The interface components rendered on the page are Visualization reports. These reports are rendered using the IBM SPSS Collaboration and Deployment Services Tag Library. The page itself is integrated into IBM SPSS Collaboration and Deployment Services Deployment Portal using the Tab Extension framework.

IBM SPSS Collaboration and Deployment Services Tag Library

The IBM SPSS Collaboration and Deployment Services Tag Library provides support for running Visualization reports that generate the bulk of the content on the Model Management page.

The tag library also supports interactivity between reports, allowing a source report to invoke a target report. The source report passes parameters to the target report for processing.

Report definitions

The Report definitions used by the Model Management page are stored in the following directory within the IBM SPSS Collaboration and Deployment Services Repository installation:

`<installation-directory>/components/peb-mmd/reports`

The visualization reports can be opened using the IBM SPSS Visualization Designer, or a text or XML editor.

The reports are provided for reference purposes, and should not be directly modified. Any modification of the reports will not be supported by IBM Corp.. However, you may copy the reports and modify the copies as needed.

Running visualization reports

The following usage notes must be considered when running visualization reports.

- Visualization reports use a value of *ONDEMAND* for the `activate` attribute of the `repositoryItem` tag.
- Parameters required for the visualization reports are passed by the master reports. See the topic "Visualization report interactivity" on page 54 for more information.

JavaScript API

The tag library has a framework built using JavaScript methods. These JavaScript methods provide both a sound validation framework and a handle to the user to run the reports on demand.

In order to run the reports on demand, the tag library provides a public API. This public API is available in the *reportTagLibPublicAPI.js* file within the *IBMSPSSTagLib.war*. The JavaScript file contains the following API:

```
function runRepositoryItem( reportName, linkData, targetId )
```

For Model Management, this function is used to invoke the child reports for the master report.

If the *linkData* in the API call is null, the report runs with the data available within the JavaServer Page supplied using the various IBM SPSS Collaboration and Deployment Services Tag Library tags. Just before calling `runRepositoryItem`, the JavaScript code stores the parameter values to the html hidden control. The tag library framework picks up these values and passes them as parameters to the report being run.

The *targetID* fields correspond to the individual `DIV` ids where the report is to be rendered.

Visualization report interactivity

The Performance versus Scenario graph generated by the visualization report for the Champions Tab supports interactivity. Whenever the user clicks on a bar in the graph, the details of the corresponding scenario are displayed in an adjacent area. The reports use the `actionHandler` and `actionParameter` tags to achieve this functionality.

Using the `actionHandler` tag is not necessarily required for visualization reports. Typically, the `sourceLinkRepositoryItem` tag would work just as well for visualization reports. However, in the case of the Model Management page, the visualization chart can occur multiple times on the page. The application needs special logic to be able to expand detail rows and to run the target reports with specific output locations. The `actionHandler` tags offer that additional level of control.

The section of the page that renders the *Performance versus Scenario* Visualization report follows:

```
<ibmspss-taglib:repositoryItem
name="Champions_Scenario_Index_Report"
   inputURI="ChampionsScenarioIndex.viz"
   repositoryCredentialName="localhost"
   activate="ONDEMAND"
   outputType="HTML"
   location="championsTabVisReport">
   <ibmspss-taglib:actionHandler event="onclick" function="selectCCScenario">
      <ibmspss-taglib:actionParameter name="filename" />
      <ibmspss-taglib:actionParameter name="filepath" />
      <ibmspss-taglib:actionParameter name="ccid" />
      <ibmspss-taglib:actionParameter name="equivalencekey" />
   </ibmspss-taglib:actionHandler>
</ibmspss-taglib:repositoryItem>
```

The `repositoryItem` tag gives details about the bar chart to be rendered. The nested `actionHandler` tag indicates that the JavaScript function `selectCCScenario` should be called whenever the `onClick` event occurs for the bars. The `actionParameter` tags nested within the `actionHandler` tag indicate that *filename*, *filepath*, *ccid*, and *equivalencekey* will be passed to the `selectCCScerario` function.

Each of these fields is defined within the visualization report XML. The definition for the *filename* variable is the following:

```
<sourceVariable
categorical="true"
   id="filename"
   source="delimitedFileSource_430"
   sourceName="ct_filename">
```

This tag indicates that the column defined as *ct_filename* within the data set will be used as *filename* by this report.

The JavaScript function `selectCCScerario` receives the id of the report on which the event occurred and an array of the parameter values. Internally, it calls `runReport` for dependent child reports and passes them the value array. See the topic "JavaScript API" on page 54 for more information.

URL fragments

The Model Management page displays some repository artifacts in an I-FRAME. These artifacts are the outputs generated by certain job runs.

An artifact is loaded by setting the source of the I-FRAME to the URL having the following format:

```
http://<servername>:<port>/peb/view?id=<artifact resource id>
```

Tab extension framework

The navigation tabs of IBM SPSS Collaboration and Deployment Services Deployment Portal can be expanded to include custom entries using the Tab Extension framework. The Model Management functionality uses this framework to add an entry point into the Model Management page.

IBM SPSS Collaboration and Deployment Services Deployment Portal reads extension files present in the following directory:

`<installation-directory>/components/peb/extensions`

These files are scanned to find all instances of the `peb-extension` elements. These elements will be individually rendered in the interface, provided the user credentials include any required actions. Any custom application must provide:

- Extension XML file or an entry in an existing extension XML for the application
- Appropriate entries in the localized text (`.tx`) file

The Model Management functionality is contained within the `peb-mmd.package` file in the *staging* directory of the repository installation. The package includes the file *mmd_extension.xml* in the *peb/extensions* directory. This XML file controls the appearance and functionality of the Model Management tab.

```
<file-viewer>
   <peb-extension>
      <tab-id>pebMmdTab</tab-id>
      <tab-key>mmd/pebMmdTabTitle</tab-key>
      <tab-url>
         /peb-mmd/controller?actionName=LoginToMMDAction
      </tab-url>
      <tab-icon>/image2?file=someIcon.gif</tab-icon>
      <tab-position>2</tab-position>
      <tab-security>
         <capability>RunReport</capability>
         <capability>ViewModelManagementDashboard</capability>
      </tab-security>
   </peb-extension>
</file-viewer>
```

Elements defined within this file include:

- The *tab-id* element which is the unique id for the tab. In this case it is *pebMmdTab*.
- The *tab-key* element which references the text appearing on the new tab. Model Management isolates any localized text in XML files having `.tx` extensions. The key identifies the element in the language file containing the text to be displayed. In this case, the *mmd/pebMmdTabTitle* key corresponds to the text *Model Management*.
- The *tab-url* element which specifies the URL invoked when the user clicks the tab. The URL can be either fully qualified (starting with a slash '/' character) or relative to the IBM SPSS Collaboration and Deployment Services Deployment Portal application. In the latter case, the context is assumed to be peb. The link must point to a valid URI, with the URI location specified being the responsibility of the custom application. For Model Management, the link includes a reference to the war file *peb-mmd.war*.
- The *tab-security* element which identifies the actions required to access the tab. If the current user does not have these actions, the tab will not be displayed in the header JSP. Model Management requires the *RunReport* and *ViewModelManagementDashboard* actions.

Chapter 7. Creating custom data service drivers

The IBM SPSS Collaboration and Deployment Services data service API provides public Java interfaces for implementing custom drivers to access non-standard data sources. For example, a custom driver may be necessary in cases when JDBC or ODBC access to a production database environment is not allowed because of security considerations. It may also be necessary for file-based data sources, cached data, or legacy data sources for which no JDBC or ODBC support exists.

This functionality is used primarily by the scoring service.

This section provides an overview of the data service API, instructions for creating a custom driver, and a link to download a custom driver example.

Data service API

The API for implementing a custom driver is provided in the package `adv-server.jar` as interface: `com.spss.adv.dataService.IDataAdapter`.

You can find the package `adv-server.jar` in `<Repository installation directory>/staging/scoring.package`.

Creating a custom driver

To create a custom driver using the data service API:

1. Extract `adv-server.jar` and make sure it's in the classpath.
2. Write the Java source code implementing the data service interfaces and compile the driver classes.
3. Create the driver package (as a JAR file).
4. Depending on the application server, deploy the package to your IBM SPSS Collaboration and Deployment Services installation:

   For **WebSphere**, copy it to the directory `<appserver_websphere_profile_path>/InstalledApps/<appserver_websphere_deploy_target_node>/IBM_SPSS_Collaboration_and_Deployment_Services_<Version>.ear/lib`

   For **WebSphere Liberty**, copy it to `lib` folder of the `.ear` file `<repository installation directory>/toDeploy/liberty/cds<version>.ear.`

   For **JBoss**, copy it to the `lib` folder of the `.ear` file `<JBoss installation directory>/standalone/deployments/cds<version>.ear`.

   For a **clustered installation**, refer to the clustering section of the installation and configuration documentation.
5. Depending on your application server, restart the IBM SPSS Collaboration and Deployment Services server.
6. To test the driver, create a data source with the type set as **Data Service Data Source** in Deployment Manager, then use the data source in a real-time data access plan in an Analytic Data View, and preview the data.

Custom driver example

You can download an example custom driver from here. It mainly includes:

- `SimplifiedADVAdapter.java` (implements `com.spss.adv.dataService.IDataAdapter`)
- `simplified-adv-ds.jar` (execution file)

The driver retrieves the data from a CSV file with commas as column separators. The information for the file is specified in the corresponding property when creating the driver.

To demonstrate the functionality of the driver:

1. Deploy `simplified-adv-ds.jar`.
2. Create a data service data source with the name **SimpleADVDS** and the following parameters:
   - **com.ibm.spss.cads.dataservice.adv.SimplifiedADVAdapter** as the driver class name
   - **DataLocation** driver property that points to the full path, including the file name, where the CSV data file is located. For example:

     ```
     DataLocation=c:\tmp\test.csv
     ```
3. Create a CSV file with the following columns and make sure it's in the location specified by the `DataLocation`:
   - **Input1** (String)
   - **Input2** (String)
4. In Modeler, create a stream called `test.str` using the same metadata from step 3 as the input data, then store the stream in the IBM SPSS Collaboration and Deployment Services Repository.
5. Create an Analytic Data View called **testADV** with a real-time Data Access Plan called **DataAccessPlanRT**. For **Table1**, update the **Override Source Nodes** by choosing the type **Data Service Data Source** and custom driver name **SimpleADVDS** that you created in step 2.
6. Browse the data by previewing the Source Node so you can test if the data can be fetched correctly.
7. Configure scoring using the same `test.str` stream you created in step 4 by specifying the type as **Data Access Plan**, the Analytic Data View as **testADV**, the Data Access Plan as **DataAccessPlanRT**, and the Analytic Data View Table as **Table1** in the Data Provider Settings.
8. Run the stream.

Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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.

Index

L

N

O

P

R

S

T

U

V

W