IBM SPSS Analytics Toolkit for IBM InfoSphere Streams

Version 2 Release 0

User's Guide

Note
Before using this information and the product it supports, read the information in "Notices" on page 17.

Product Information

This edition applies to version 2, release 0, modification 0 of IBM SPSS Analytics Toolkit for IBM InfoSphere Streams and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. SPSS Analytics Toolkit overview

The IBM® SPSS® Analytics Toolkit contains IBM InfoSphere® Streams operators that integrate with IBM SPSS Modeler and IBM SPSS Collaboration and Deployment Services products to implement various aspects of predictive analytics in your IBM InfoSphere Streams applications.

The toolkit includes the following operators:

SPSSScoring operator
: Integrates with IBM SPSS Modeler Solution Publisher to enable the scoring of your IBM SPSS Modeler predictive models in IBM InfoSphere Streams applications.

SPSSForecast operator
: Merges a data stream of actual values with another data stream of future predictor values as a window of input to be scored in the **SPSSScoring** operator.

SPSSPublish operator
: Automates the IBM SPSS Modeler Solution Publisher publish function, which generates the required executable images that are needed to refresh the model that is used in your IBM InfoSphere Streams applications from the logical definition of a scoring branch that is defined in an IBM SPSS Modeler file.

SPSSRepository operator
: Detects notification events that indicate changes to deployed models managed in the IBM SPSS Collaboration and Deployment Services Repository and retrieves the indicated IBM SPSS Modeler file version for automated publish and preparation for use in IBM InfoSphere Streams applications.

The granularity of the operators that are implemented in this toolkit enables the following basic implementation strategies:

SPSSScoring operator in the IBM InfoSphere Streams application
: Site manages changes to IBM SPSS Modeler files that are used in the application, but places promotion and related publish of updated models outside of the IBM InfoSphere Streams application domain.

SPSSScoring and SPSSPublish operators used in the IBM InfoSphere Streams application
: Site manages IBM SPSS Modeler file versions outside of the IBM InfoSphere Streams application, but uses the automation of publish functionality, refreshing the models that are used in deployed applications.

SPSSScoring, SPSSPublish, and SPSSRepository operators used in the IBM InfoSphere Streams application
: IBM SPSS Modeler assets are managed in IBM SPSS Collaboration and Deployment Services Repository. The IBM InfoSphere Streams application refreshes the IBM SPSS Modeler files that are used by operators while jobs are running. All download, publish, and refresh automation is based on promotion event notifications that are sent from the IBM SPSS Collaboration and Deployment Services Repository.

Supported product versions

The IBM SPSS Analytics Toolkit for IBM InfoSphere Streams is designed to run with IBM InfoSphere Streams version 3 or later and IBM SPSS Modeler Solution Publisher version16 or later.

The IBM SPSS Analytics Toolkit is installed by IBM SPSS Modeler Solution Publisher, which is bundled with IBM SPSS Collaboration and Deployment Services 6.0 and later.

Chapter 2. IBM SPSS Analytics Toolkit installation

The IBM SPSS Analytics Toolkit installation assets are in the `InfoSphere` subfolder under the root directory of your IBM SPSS Modeler Solution Publisher installation.

The installation assets include the following files:

- The IBM SPSS Analytics Toolkit for IBM InfoSphere Streams installation archive `SpssAnalyticsToolkit.tar.gz`
- The toolkit installation helper script `installToolkit.sh`
- A readme file that describes where to find this documentation and a short description of the operators in this toolkit

Installation prerequisites

Verify that you have a functioning IBM InfoSphere Streams installation before you attempt to install the IBM SPSS Analytics Toolkit as an enhancement to that environment.

The following criteria must be satisfied:

- IBM InfoSphere Streams is installed and all fix packs are applied
- The **`STREAMS_INSTALL`** environment variable is set to point to the IBM InfoSphere Streams installation
- IBM InfoSphere Streams is working properly in your development and production environments
- IBM SPSS Modeler Solution Publisher is installed in your environment and can be accessed by all systems that are building or running applications that use operators from this toolkit

After all of the prerequisites are satisfied, you can proceed with the installation of the IBM SPSS Analytics Toolkit for access in your development, test, and production IBM InfoSphere Streams environments.

Installation considerations

To use this toolkit after it is installed, you must reference the toolkit installation directory or a parent directory of the toolkit installation directory as part of your IBM Streams Processing Language (SPL) application build environment.

To reference the toolkit, modify the **`STREAMS_SPLPATH`** environment variable or use the `–t` option on the **`sc`** compiler command.

If you are using multiple toolkits, install all toolkits into a common parent directory to minimize the path information that must be specified. If you use a common directory, you need to specify the common directory to make all toolkits that are installed in any subdirectories available for use.

Installing IBM SPSS Analytics Toolkit

The `installToolkit.sh` helper script in the `InfoSphere` directory of your IBM SPSS Modeler Solution Publisher installation extracts the toolkit content and places it in the target toolkit directory.

Run the script from the `InfoSphere` directory of your IBM SPSS Modeler Solution Publisher installation. The script takes a single command-line parameter, the file path for the toolkit installation.

The following example places the IBM SPSS Analytics Toolkit under the common IBM InfoSphere Streams `toolkits` root directory of your installation by using the default **`STREAMS_TOOLKIT_INSTALL`** environment variable:

```
./installToolkit.sh
```

If the **STREAMS_TOOLKIT_INSTALL** environment variable is not set to the parent directory for all IBM InfoSphere Streams toolkit installations, pass the destination path as a parameter in the `installToolkit.sh` command line:

```
./installToolkit.sh /home/streamsdev/toolkits
```

This script performs some basic validation of the IBM InfoSphere Streams installation and your indicated target directory before it attempts the installation. If any portion of this validation fails, the script exits with an error message. If the script succeeds, it displays a notification of this fact before it ends.

Important: Make sure the **execute** privilege is set for the objects in the IBM SPSS Analytics Toolkit.

After the installation

After installing the toolkit into your IBM InfoSphere Streams environment, define the **CLEMRUNTIME** environment variable on all systems that are building or running applications that use operators from this toolkit.

The **CLEMRUNTIME** environment variable specifies the path to the IBM SPSS Modeler Solution Publisher installation.

You also need to ensure that the **LD_LIBRARY_PATH** variable is correctly set for all systems on which the IBM InfoSphere Streams operator is deployed to enable dynamic library load of all necessary IBM SPSS Modeler Solution Publisher libraries. The `setupEnv.sh` script in the root of the toolkit provides an example and might be sourced directly by editing the content to match the information unique to your IBM SPSS Modeler Solution Publisher installation.

Important: To use the `setupEnv.sh` script, you must first either define the **CLEMRUNTIME** environment variable externally or by editing this script. You must source the script that enables these operators after the `streamsprofile.sh` script. Source these settings in the same manner as IBM InfoSphere Streams in all environments.

The following helper script files are available in the toolkit:

publishHelper.sh
: Publishes IBM SPSS Modeler files for use in IBM InfoSphere Streams applications.

passwordEncoder.sh
: Encodes the IBM SPSS Collaboration and Deployment Services password when used in the operator configurations of your IBM InfoSphere Streams applications.

Important: To use these sample scripts, set the **SPSS_TOOLKIT_INSTALL** environment variable to the directory where the IBM SPSS Analytics Toolkit is installed. See the `setupEnv.sh` script for an example.

Chapter 3. Using the IBM SPSS Analytics Toolkit

IBM InfoSphere Streams applications that use the IBM SPSS Analytics Toolkit can be compiled in IBM InfoSphere Streams Studio or by using the SPL compiler command, **sc**.

To compile an IBM InfoSphere Streams application by using the SPL compiler command, specify the toolkit installation directory either in the **STREAMS_SPLPATH** environment variable or in the `–t` option on the **sc** compiler command.

The following is an example of adding this toolkit to the **STREAMS_SPLPATH** environment variable:

```
export STREAMS_SPLPATH=/home/myuserid/toolkits/com.ibm.spss.streams.analytics
```

Adding IBM SPSS Analytics Toolkit to **STREAMS_SPLPATH** makes the toolkit available by default in both IBM InfoSphere Streams Studio and in **sc** command compilation. It is a good practice to load all commonly referenced toolkits in this manner.

To explicitly add the IBM SPSS Analytics Toolkit to your IBM InfoSphere Streams Studio environment, add it to the Toolkit Locations view of your InfoSphere Streams Explorer.

The IBM SPSS Analytics Toolkit can also be specified in the SPL compiler command as illustrated in the following example:

```
sc –t /home/myuserid/toolkits/com.ibm.spss.streams.analytics –M MyApp
```

Operators

The operators in the IBM SPSS Analytics Toolkit are all defined under the `com.ibm.spss.streams.analytics` namespace.

To use the toolkit operators in an IBM InfoSphere Streams application, include the following `use` clause in your SPL source file:

```
use com.ibm.spss.streams.analytics::*;
```

You can also be more specific in your `use` clause by calling out individual operators, replacing the asterisk (*) with the specific operator your application requires.

Details on each operator in this toolkit, including their configuration and usage options, are covered in the following sections.

SPSSScoring operator

The **SPSSScoring** operator is an IBM InfoSphere Streams generic primitive operator that scores stream data.

The implementation of the operator is optimized by your configuration through code generation to match the published IBM SPSS Modeler scoring branch that it is configured to run. Your application will score the data in the stream through the integration of this operator with IBM SPSS Modeler Solution Publisher.

You can use this operator without the other operators in this toolkit by generating deployment metadata files for the new or modified scoring branch. Publish either from an export node from the IBM SPSS Modeler client or by using the supplied publish script included in this toolkit to produce the required `PIM`, `PAR`, and `XML` files.

Important: The release version of the environment that generates the PIM, PAR, and XML files must match the release version of IBM SPSS Modeler Solution Publisher you are using in your IBM InfoSphere Streams run time environment.

It is possible to use a **DirectoryScan** operator from the IBM InfoSphere Streams standard toolkit to trigger a model refresh by the **SPSSScoring** operator when you update the PIM and PAR files in the source directory.

The parameters for this operator follow:

pimfile
The full path to the executable image file generated by the publish of the IBM SPSS Modeler file scoring branch.

parfile
The full path to the file of parameters to be used in preparation of the executable image file, generated by the publish of the IBM SPSS Modeler stream file scoring branch.

xmlfile
The full path to the XML file that describes the inputs and outputs of the published IBM SPSS Modeler stream file scoring branch, used to validate input parameters.

modelFields
A list of strings that reference the scoring branch input field names as defined in the input section of the XML file that is passed in the **xmlfile** parameter.

streamAttributes
A list of expressions that define the input tuple attribute expressions to be mapped to the model fields in the order entered. Data types must match expected data types as defined in the XML file that is passed in the **xmlfile** parameter.

scoreOnWindowPunc
An optional Boolean parameter that indicates whether the scoring branch is expecting a set of inputs that are demarcated by window punctuation (true) or not. The default (false) is the score-each-input-tuple mode if this parameter is not specified.

maxTransactionOutput
An optional limit to the number of output tuples that are submitted for each window of input tuples scored (**scoreOnWindowPunc** set to true) to suppress some of the extra details from model algorithms that can be configured to score in transaction mode. The default is to submit all model outputs if this parameter is not specified.

implicitNULL
An optional Boolean parameter that indicates whether detection for not-a-number (NaN) in floating point inputs is detected and translated to NULL on the scoring call or not. The default is false if not specified.

Input ports

This operator defines the required input port where the tuples that contain the data to be scored flow. This port is non-windowed (single tuple per score restriction on scoring branch) and potentially mutates the attributes of the input tuple.

Note: Although the scoring input port is marked as oblivious to window punctuation in its configuration, it is expecting these punctuations in the data stream if you set **scoreOnWindowPunc** to true. Also, final punctuation does not trigger a score; the window punctuation is required in all cases.

This operator also defines one optional input port where notification of a modified PIM file from this toolkit **SPSSPublish** operator, the **DirectoryScan** operator from the IBM InfoSphere Streams standard toolkit, or another operator can trigger a worker thread to prepare the new scoring branch for execution

and then swap this prepared instance for the current instance without blocking the scoring flow. These refresh events are logged at the `L_INFO` level.

Output ports

This operator has one output port and defines helper functions for you to indicate the following characteristics:

fromModel(attribute)
: Submits an attribute that is referenced in this output function as returned by the scoring branch. The values can be modified.

fromModel (attribute, default value)
: Submits attribute referenced output function as returned by the scoring branch if a value was returned. Otherwise, it returns the default value indicated.

Important:

- All input attributes are submitted over the output port unless replaced by **`fromModel`** output function expressions in the output configuration. More output attributes that are generated by the scoring branch that you want passed downstream can be specified in **`fromModel`** expressions in your configured output specification.
- When you score in `Window Punctuation mode` (**`scoreOnWindowPunc`** parameter true), only the first input tuple is used to initialize all output tuples to emphasize the fact that there can be no assumptions of direct correlation between input attribute values and output attribute values in this mode.

XML file generated by publish action

Detailed knowledge of the inputs and outputs of the scoring branch is required in the configuration description. This information is communicated in the `XML` file that is generated during the publish operation.

Remember: The input fields that are required to run the configured scoring branch and the output fields it produces define the data contract for a configuration of this operator in your IBM InfoSphere Streams application.

The **`inputDataSources`** element of this `XML` file defines the input fields that are required for each data source of the scoring branch.

Note: This release restricts this specification to one data source so the input fields of interest are all listed in the **`fields`** element under the first *<inputDataSource name="<node ID>" type="Delimited">* entry. For each field listed, note the *storage* value that defines its data type and the *name* defined.

An example input data contract description follows:

```
<inputDataSources>
    <inputDataSource name="file0" type="Delimited">
         ... ignore <parameters>
        <fields>
            <field storage="string" type="flag">
                 <name>sex</name>
                 ... ignore value range / categorical values, etc.
            </field>
            <field storage="integer" type="range">
                 <name>income</name>
                 ... ignore value range / categorical values, etc.
            </field>
        </fields>
    </inputDataSource>
</inputDataSources>
```

The **outputDataSources** element of this XML file defines the output fields that are produced by the execution of this scoring branch. You indicate a single terminal node when you publish the scoring branch, resulting in a single *<outputDataSource name="<node ID>" type="Delimited">* element. The output fields of interest are all listed in the **fields** element under the first output data source entry. For each field listed, note the *storage* value that defines its data type and the *name* defined.

An example output data contract description follows:

```
<outputDataSources>
    <outputDataSource name="file3" type="Delimited">
         ... ignore <parameters>
        <fields>
            <field storage="string" type="flag">
                 <name>sex</name>
                 ... ignore value range / flag / categorical values, etc.
            </field>
            <field storage="integer" type="range">
                 <name>income</name>
                 ... ignore value range / flag / categorical values, etc.
            </field>
            <field storage="string" type="flag">
                 <name>$C-beer_beans_pizza</name>
                 ... ignore value range / flag / categorical values, etc.
            </field>
            <field storage="real" type="range">
                 <name>$CC-beer_beans_pizza</name>
                 ... ignore value range / flag / categorical values, etc.
            </field>
        </fields>
    </outputDataSource>
</outputDataSources>
```

Give special consideration to Date and Timestamp predictors in the scoring branch. The IBM InfoSphere Streams `timestamp` primitive data type is based on Epoch (00:00:00, January 1, 1970 Coordinated Universal Time). The IBM SPSS Modeler `date` and `timestamp` data types are also based on midnight January 1, 1970, but make no Coordinated Universal Time adjustment. The most efficient way to pass IBM InfoSphere Streams `timestamp` attributes into the **SPSSScoring** operator that implements the scoring branch is by `int64` seconds. Any adjustment to this value must be determined by investigation of the system settings of the compute nodes that run the IBM InfoSphere Streams application to avoid any unplanned time zone influence on this transformation during scoring.

Example usage

In this example, a set of input data that is sourced from a `CSV` file is scored. The **SPSSScoring** operator listens for a new version of its predictive model and refreshes its executable image without blocking the scoring of the data stream.

```
composite SPSSScoringExample {
 type
  static DataSchema =
   rstring s_sex,
   int64 baseSalary,
   int64 bonusSalary;
  static DataSchemaPlus =
   DataSchema, tuple<int64 income, rstring predLabel, float64 confidence>;
 graph
  stream<DataSchema> data = FileSource() {
   param file: "input.csv";
  }
  stream<rstring fileName> notifier = DirectoryScan() {
   param directory : "/home/streamsadmin/is/temp/small";
  }
  stream<DataSchemaPlus> scorer = com.ibm.spss.streams.analytics::SPSSScoring(data;notifier) {
   param
```

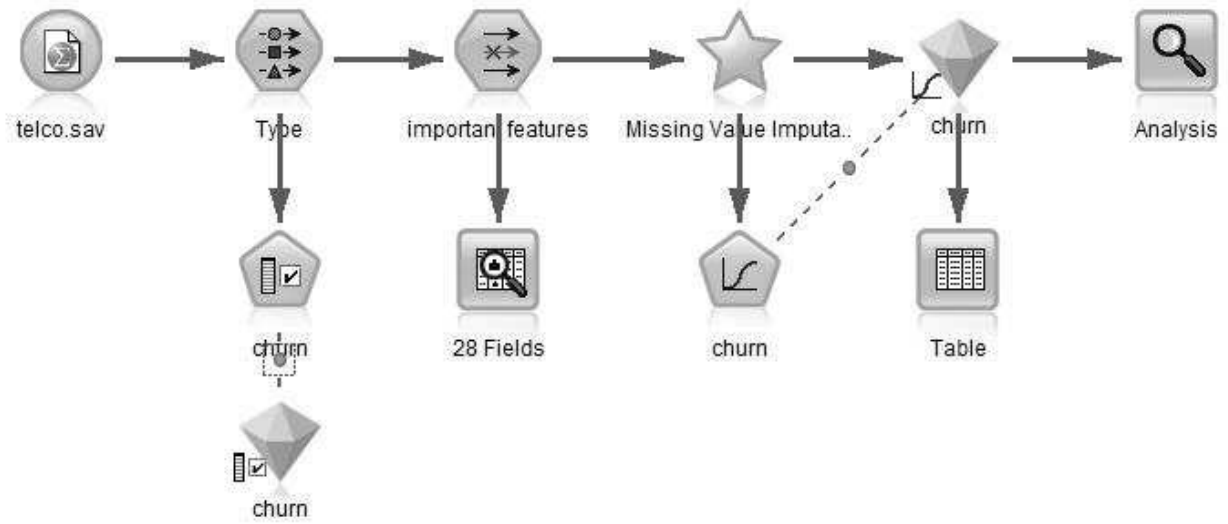
```
    pimfile: "model.pim";
    parfile: "model.par";
    xmlfile: "model.xml";
    modelFields: "sex","income";
    streamAttributes: s_sex, baseSalary+bonusSalary;
   output scorer:
    income = fromModel("income"),
    predLabel = fromModel("$C-beer_beans_pizza"),
    confidence = fromModel("$CC-beer_beans_pizza");
  }
  ...
}
```

Configuration templates are included in the operator model for your convenience.

Scoring terminology and background information

Scoring is the act of running the set of process nodes that are defined in the designed path of IBM SPSS Modeler files that implement the plan for producing the predictive analytics.

In the following image, you can trace the scoring branch from a specific terminal node, such as *Table*, to the left through the various process nodes to the source node `telco.sav`.



This IBM SPSS Modeler file highlights some important concepts:

- A scoring branch seldom has a simple source / model nugget / terminal design. Avoid the use of the oversimplified term *predictive model* or *model* for an implementation of a scoring plan that produces the predictive analytics.
- The input data attributes defined by the source node and the output data attributes defined by the terminal node define the data contract of the scoring branch. You can radically change your scoring branch implementation and still use it in an IBM InfoSphere Streams application that is configured against another version of your scoring plan if this data contract is maintained.
- In this graphic, the churn node is a model nugget, which is a predictive model that is constructed by using data mining techniques in the build branch of an IBM SPSS Modeler file. It is much more common to periodically retrain the predictive models in a scoring branch by using new data than it is to redesign the scoring branch itself.
- There can be many processing branches in your IBM SPSS Modeler file, but you publish one scoring branch to prepare the executable image for use in your IBM InfoSphere Streams applications.

- To publish the scoring branch for use in an IBM InfoSphere Streams application that uses IBM SPSS Modeler Solution Publisher, export the branch from the IBM SPSS Modeler client or use the publish functionality that is provided by the toolkit. Publishing generates the executable image (`.pim` extension) file, the execution parameters (`.par` extension) file, and an `XML` file that describes the required inputs and resulting outputs of the scoring branch.

SPSSForecast operator

The **`SPSSForecast`** operator is a primitive operator that merges a data stream of actual values (port 0) with another data stream of future predictor values (port 1) adding window punctuation. This operator triggers transactional scoring in the **`SPSSScoring`** operator. Both input data streams must be sliding windows.

When the actual input window triggers, the matching tuples from the future predictor window will be merged into the output tuples generated according to the output port configuration. Remaining tuples in the future predictor window will then be submitted by using the default actual tuple that is defined in the configuration followed by window punctuation.

This operator exists to feed an **`SPSSScoring`** operator that is configured for time series processing that needs to see both the actual measures and associated future predictor values for a specified time period. This process effectively does a model refresh as it does its scoring.

The parameters for this operator follow:

match A required Boolean expression that defines how input tuples in the actual and future predictor windows are to be matched to one another.

defaultActualTuple
An SPL tuple definition of constant values to be used in outputs of future predictor tuples beyond the last match from the actual input tuples in the window at the time it triggers.

Input ports

This operator requires two input ports, both of which must be sliding windows.

Port 0 The tuples that represent the actual values. This sliding window triggers output submissions.

Port 1 The tuples that represent the future predictor values. The contents of this sliding window are used in matches to actual values and also define the additional future predictor outputs.

Output port

This operator has one output port that accepts expressions that reference tuple attributes from both input ports.

Example usage

This example uses input from two Beacons, one representing the actual values from some sensor and the other representing the future predictor values that are generated from some source. A simple integer key value increments with each input tuple generated by the Beacon.

The configuration of the **`SPSSForecast`** operator instance will hold both the actual and the future predictor inputs in sliding windows that will have at most 10 tuples in them. The example triggers the output from this operator on every actual input. Note the input tuple match expression that is used to join the inputs that are held by the widows and the **`defaultActualTuple`** definition to be used in the future predictor outputs.

```
composite ForecastTest {
 type
  static DataSchema = int64 k, float64 v;
```

```
  static DataSchemaJoined = int64 key, float64 actual, float64 fPred;
 graph
  stream<DataSchema> sActual = Beacon() {
   logic state : mutable int64 i = 0;
   param
    period: 0.1;
    iterations: 100u;
   output
    sActual: k = i++, v = (float64)(random() * 200000.0);
  }
  stream<DataSchema> sFPred = Beacon() {
   logic state : mutable int64 i = 1;
   param
    period: 0.1;
    iterations: 100u;
   output
    sFPred: k = i++, v = (float64)(random() * 200000.0);
  }
  stream<DataSchemaJoined> forecastOutput =
   com.ibm.spss.streams.analytics::SPSSForecast(sActual as A; sFPred as F) {
   window
    A : sliding, count(10), count(1);
    F : sliding, count(10);
   param
    match : A.k == F.k;
    defaultActualTuple : { k=(int64)0, v=nanl() };
   output
    forecastOutput: key = F.k, actual = A.v, fPred = F.v;
  }
  ...
}
```

SPSSPublish operator

The **SPSSPublish** operator is a Java primitive operator that automates the publish of an IBM SPSS Modeler file scoring branch and summarizes the generated files so downstream operators can refresh their scoring implementation with the PIM, PAR, and XML files that are created or updated by the publish operation.

This operator might be used with other operators but its designed purpose is to be attached to the optional notification port of the **SPSSScoring** operator to trigger a model refresh. In normal usage, the input to the **SPSSPublish** operator would come from a **DirectoryScan** operator or the **SPSSRepository** operator in this toolkit.

The parameters for this operator follow:

sourceFile
: The fully qualified name of the IBM SPSS Modeler file to be published.

terminalNodeID
: ID of the terminal node in the IBM SPSS Modeler file to be published that defines the scoring branch. This value is required if the IBM SPSS Modeler file is not deployed with its scoring branch denoted in its metadata; otherwise, this value is optional.

targetPath
: Directory path at which the generated execution image (PIM), execution parameters (PAR), and execution description (XML) files are written. This parameter is optional, with the default target path being the same as the source.

cdsServer
: The address of the IBM SPSS Collaboration and Deployment Services Repository server. This value is required only if the scoring branch to be published contains references to other objects stored in the repository.

userID
ID of the user authorized to access the objects that are referenced in the IBM SPSS Collaboration and Deployment Services Repository. This value is required if **cdsServer** is specified.

password
Password of the user authorized to access the objects that are referenced in theIBM SPSS Collaboration and Deployment Services Repository. This value is required if **cdsServer** is specified.

encodedPassword
An optional Boolean parameter that indicates whether the password is encoded (true) using the mechanism that is supplied in the toolkit or not (false). If not specified, this parameter is assumed to be false.

maxMemory
An optional unsigned integer parameter that can be used to indicate the megabytes of memory to be allocated for the Java VM started to publish the scoring branch.

Input port

This operator defines one required input port where the tuples received as input describe the file to be considered for the publish automation. Only files that match the **sourceFile** parameter are published. This input port is non-mutating and non-windowed.

Output port

This operator has one output port where the descriptions of the files that are generated by the publish action are submitted.

Example usage

This example monitors file changes in a specific directory of the file system and publishes the configured IBM SPSS Modeler file scoring branch when it is modified.

```
composite SPSSPublishExample {
type
 outputTuple = tuple<rstring fileName>;
graph
 stream<rstring fileName> file = DirectoryScan() {
  param
   directory : "/home/streamsadmin/demo";
   pattern: "stream.str";
 }
 stream<outputTuple> Output = com.ibm.spss.streams.analytics::SPSSPublish(File){
  param
   sourceFile:" /home/streamsadmin/demo/stream.str";
 }
 () as sink = Custom(Output){
  logic
   onTuple Output: printStringLn("File Path: "+ Output.fileName);
 }
}
```

SPSSRepository operator

The **SPSSRepository** operator is a primitive Java source operator that is configured to listen for specific change notifications to an object deployed in the IBM SPSS Collaboration and Deployment Services Repository.

When a notification occurs indicating that the object this operator is configured to monitor changed, the associated file version is retrieved from the repository and written to the configured target directory. On successful download, an output tuple that describes the file that is updated is submitted to communicate this event to downstream operators.

This operator might be used with other operators, but normally it is attached to the input port of the **SPSSPublish** operator to trigger the publish generation of the files that are needed to accomplish a model refresh in the **SPSSScoring** operator.

The parameters for this operator follow:

cdsServer
The address of the IBM SPSS Collaboration and Deployment Services server.

userID
ID of the user authorized to access the server and objects in the IBM SPSS Collaboration and Deployment Services Repository

password
Password of the user authorized to access the server and objects that are referenced in the IBM SPSS Collaboration and Deployment Services Repository

resourceURI
URI string that references the IBM SPSS Modeler file to be monitored in the IBM SPSS Collaboration and Deployment Services Repository

versionLabelName
The name of the label that is used to identify promoted resource versions to be monitored. If omitted, any new version triggers download.

targetFilePath
Path of the target directory to which file versions downloaded by this operator are written.

detectionPeriod
Optional detection period, in seconds, determining how frequently this operator looks through the notifications from the IBM SPSS Collaboration and Deployment Services Repository. If not specified, an internal default of 10 minutes is used.

encodedPassword
An optional Boolean parameter that indicates whether the password is encoded (true) using the mechanism that is supplied in the toolkit or not (false). If not specified, this parameter is assumed to be false.

Input ports

None. This operator is a source operator.

Output port

This operator has one output port where the description of the file that is downloaded is submitted.

Example usage

This example monitors notifications on the association of the label *PRODUCTION* to an IBM SPSS Modeler file integrated in the repository under the URI *spsscr:///?id=09895272b9c1042e00000133fad8111192f4*. When the notification is detected (default **detectionPeriod**), the file version is downloaded to the */home/streamsadmin/cdsFileFolder* directory.

```
composite SPSSRepositoryExample {
type
 outputTuple = tuple<rstring filePath>;
```

```
graph
 stream<outputTuple> Output = com.ibm.spss.streams.analytics::SPSSRepository(){
  param
   cdsServer: "http://9.119.82.114:9081";
   userID:"admin";
   password:"12345678";
   resourceURI:" spsscr:///?id=09895272b9c1042e00000133fad8111192f4";
   versionLabelName:"PRODUCTION";
   targetFilePath:"/home/streamsadmin/cdsFileFolder";
 }
 () as sink = Custom(Output){
  logic
   onTuple Output: printStringLn("File Path: "+ Output.filePath);
 }
}
```

To get the URI for an object in the repository, use IBM SPSS Deployment Manager client:

1. Right-click the object in the Content Explorer and select **Properties** to view a detailed summary of the object.
2. In the Properties dialog box, copy the **Object URI** value.
3. In the configuration of your IBM InfoSphere Streams application, paste the value.

Best Practices for Notifications Monitored by IBM InfoSphere Streams Applications

Use a meaningful label name with a clear and well-communicated purpose on all IBM SPSS Modeler files that are placed into production in your IBM InfoSphere Streams applications. For example, you might use a label of *STREAMS_PROD* for the production version of a file that is used in an IBM InfoSphere Streams application.

To use label move notifications to indicate that a specific version of a IBM SPSS Modeler file is to be used to refresh your running IBM InfoSphere Streams applications, you would set the label by using the Properties dialog box in IBM SPSS Deployment Manager.

You must also make sure that your user is configured to receive notifications over the RSS distribution channel in their user preferences. For more information, see the IBM SPSS Deployment Manager documentation.

Finally, indicate that notifications are to be sent whenever the label in question is set or moved on an object by creating a security subscriber for the root folder of the content repository. The subscriber corresponds to the user who reads the RSS feed for these label move notification events. For more information, see the IBM SPSS Deployment Manager documentation.

Some sites use the act of creating a new file version as a promotion indicator. This approach is natural for an IBM SPSS Collaboration and Deployment Services installation that uses separate development, test, and production repositories where the act of promoting from one environment to another triggers the processes that are to act on the new object version that is promoted to the target repository.

Note: The internal label *LATEST* effectively moves back to a later version if the most recent version of an object is deleted. This label is always implicitly associated with the latest version available in the repository.

If you are going to use file version creation notifications instead of label-based monitoring, create a security subscriber for the file object to be monitored. However, this approach does not monitor deletion of versions and is unable to revert to a previous file version when version deletion occurs.

Chapter 4. Sample applications

The IBM SPSS Analytics Toolkit for IBM InfoSphere Streams contains a set of simple sample applications to demonstrate how to use the various operators.

Each of these sample directories contains an SPL source file that defines the sample application, and an `info.xml` file to describe the sample in InfoSphere Streams Studio. The `XML` file also references a common data subdirectory with the sample data and other assets needed to run the sample.

Note: To use these samples, you must publish the `model.str` file to generate the required `PIM`, `PAR`, and `XML` files. You can publish the file using either the IBM SPSS Modeler client or the included `publishHelper.sh` script.

The following samples are available:

SPSSScoring
: This sample uses an IBM SPSS Modeler scoring branch defined in the `model.str` file and a small data file of data to be scored in the `input.csv` file to create an output file containing the predictions in the configured file sink.

SPSSForecastScoring
: This sample uses an IBM SPSS Modeler scoring branch defined in the `timeseries.str` file and inputs generated by Beacon source operators feeding an **`SPSSForecast`** operator to pass a window of information to the **`SPSSScoring`** operator, directing the outputs to a file sink.

SPSSPublishScoring
: This sample publishes any specified IBM SPSS Modeler file according to its configuration and also triggers a refresh of the scoring in the **`SPSSScoring`** operator. The sample does not define the IBM SPSS Collaboration and Deployment Services Repository connectivity information so the IBM SPSS Modeler files presented cannot contain references to other objects in the repository. This sample must be edited to point to valid file directories on your development system.

  Note: This application will not terminate on its own due to the **`DirectoryScan`** source operator used.

SPSSRepositoryPublishScoring
: This sample uses three operators from this toolkit: **`SPSSRepository`**, **`SPSSPublish`**, and **`SPSSScoring`**. The sample requires an IBM SPSS Collaboration and Deployment Services installation to work. You can choose to modify the configuration to point to any file object you deploy to the repository. Once the sample is running, you can set the label configured or store a second version of the object to the repository if not using the label recognition to get the notification that will trigger the download, publish, and refresh operations.

  Note: This application will not terminate on its own as it continually monitors notifications from the IBM SPSS Collaboration and Deployment Services Repository.

Samples in the command-line environment

To compile one of the samples from the command line, set the *SPSS_TOOLKIT_INSTALL* environment variable to the directory where the IBM SPSS Analytics Toolkit is installed. You can then run **`make`** from within one of the sample subdirectories, such as `SPSSScoring`.

By default, the sample is compiled as a distributed application. If you want to compile the application as a stand-alone application, run **`make standalone`** instead. To remove all the generated files and return the sample to its original state, run **`make clean`**.

Samples in IBM InfoSphere Streams Studio

To import a sample into IBM InfoSphere Streams Studio, you must first add the IBM SPSS Analytics Toolkit to the **Toolkit Locations** section.

In the Streams Explorer, right-click **Toolkit Locations** and select **Add Toolkit Location**. Enter the directory or click **Directory** to select the installation location of the IBM SPSS Analytics Toolkit, and click **OK**. This addition needs to be done one time.

After you add this toolkit location to your development environment, select **File** > **Import**, expand the `InfoSphereStreams` folder, and select **SPL Project**. Enter the directory or click **Browse** to select the directory of the sample you want to import, and click **Finish**.

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

Printed in USA