IBM SPSS Analytics Toolkit for InfoSphere Streams Version 2.0

SPSS Analytics Toolkit

Note:
Before using this information and the product it supports, read the general information in Appendix A.

Edition Notice:

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

You can order IBM publications online or through your local IBM representative.

•To order publications online, go to the IBM Publications Center at
www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss

•To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Table of Contents

1 Overview

The SPSS Analytics Toolkit contains InfoSphere Streams[1] operators that integrate with IBM SPSS Modeler[2] and SPSS Collaboration and Deployment Services[3] products to implement various aspects of SPSS Modeler predictive analytics in your InfoSphere Streams applications:

- SPSSScoring operator - integrates with SPSS Modeler Solution Publisher to the enable the scoring of your SPSS Modeler designed predictive models in InfoSphere Streams applications
- SPSSPublish operator - automates the SPSS Modeler Solution Publisher 'publish' function which generates the required executable images needed to refresh the model used in your InfoSphere Streams applications from the logical definition of an SPSS Modeler scoring branch defined in a SPSS Modeler file
- SPSSRepository operator - detects notification events indicating changes to the deployed models managed in the SPSS Collaboration and Deployment Services repository and retrieves the indicated Modeler file version for automated publish and preparation for use in your InfoSphere Streams applications
- SPSSForecast operator – used to merge 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.

The granularity of the operators implemented in this toolkit enable the following basic implementation strategies:

- **SPSSScoring operator only in the Streams application** – site management of changes to SPSS Modeler files used in application placing 'promotion' and related 'publish' of updated models outside of InfoSphere Streams application domain
- **SPSSScoring plus SPSSPublish operators used in the Streams application** – site manages SPSS Modeler file versions outside of stream application but leverages of automation of 'publish' functionality refreshing the models used in deployed stream applications
- **SPSSScoring, SPSSPublish and SPSSRepository operators all used in Streams application** – SPSS Modeler assets managed in SPSS Collaboration and Deployment Services repository, InfoSphere Streams application refreshes the Modeler files used by its operators while jobs executing. All download, publish and refresh automation based on promotion event notifications issued from the SPSS Collaboration and Deployment Services repository.

2 Supported Product Versions

The *IBM SPSS Analytics Toolkit for InfoSphere Streams version 2.0* is designed to run with InfoSphere Streams version 2 and later and SPSS Modeler Solution Publisher version 15 and later. The SPSS

[1] For information on IBM InfoSphere Streams see http://www.ibm.com/software/data/infosphere/stream-computing/
[2] For information on IBM SPSS Modeler see http://www.ibm.com/software/analytics/spss/products/modeler/
[3] For information on IBM SPSS Collaboration and Deployment Services see http://www.ibm.com/software/analytics/spss/products/deployment/cds/

Analytics Toolkit is shipped with SPSS Modeler Solution Publisher product, which is bundled with SPSS Collaboration and Deployment Services release 5.0 and later.

3 Installing the SPSS Analytics Toolkit

The following SPSS Analytics Toolkit install assets can be found in the ***InfoSphere*** sub-folder under the root directory of your SPSS Modeler Solution Publisher install.

- The SPSS Analytics Toolkit for InfoSphere Streams toolkit 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

3.1 Before You Begin

Please verify that you have a functioning InfoSphere Streams installation before attempting the install of the SPSS Analytics Toolkit as an enhancement to your InfoSphere Streams environment by checking the following:

- InfoSphere Streams has been installed and all fixpacks have been applied
- The STREAMS_INSTALL environment variable is set to point to this installation
- InfoSphere Streams is working properly in your development and production environments
- SPSS Modeler Solution Publisher is installed in your environment and can be accessed by all systems that will be building or executing applications using operators from this toolkit

Once all of the pre-requisites above are met you can proceed with the installation of the SPSS Analytics Toolkit for access in your development, test and production InfoSphere Streams environments.

3.2 Installation Considerations

To use this toolkit after it is installed you will need to reference this toolkit's installation directory or a parent directory of the toolkit's installation directory as part of your IBM Streams Processing Language (SPL) application build environment. This is done by modifying the STREAMS_SPLPATH environment variable or using the ***–t*** option on the ***sc*** compiler command.

If you have more than one extension toolkit installed in your environment you are encouraged to follow the 'best practices' recommendation of installing all toolkits into a common parent directory to minimize the path information that must be specified. If you follow this pattern only the parent directory is named and all toolkits installed in child directories of this parent are available for use.

3.3 Installation

The ***installToolkit.sh*** helper script in the *InfoSphere* directory of your SPSS Solution Publisher install is provided to help you extract the toolkit's content and place it in the desired target toolkit directory. The script must be executed from the *InfoSphere* directory of your SPSS Solution Publisher install and takes a single command-line parameter; the desired file path for the toolkit installation.

In the following example we are placing the SPSS Analytics Toolkit under the common InfoSphere Streams 'toolkits' root directory of your InfoSphere Streams install using the default STREAMS_TOOLKIT_INSTALL environment variable:

```
./installToolkit.sh
```

If the STREAMS_TOOLKIT_INSTALL environment variable is not set to the desired parent directory for all InfoSphere Streams toolkit installs you may pass the destination path as a parameter in the 'installToolkit.sh' command line:

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

This script will perform some basic validation of the InfoSphere Streams installation and your indicated target directory before attempting the install. If any portion of this validation fails the script will exit with an error message. If the script succeeds it will display a notification of this fact before it terminates.

Note: it is important to make sure the 'execute' privilege is set for the objects in the SPSS Analytics Toolkit.

3.4 After the Installation

Once the toolkit has been installed into your InfoSphere Streams environment(s) you will need to define the following environment variable on all systems that will be building or executing applications using operators from this toolkit:

`CLEMRUNTIME` : The path to the SPSS Modeler Solution Publisher installation

You also need to ensure that the LD_LIBRARY_PATH is correctly set on all systems that the InfoSphere Streams operator will be deployed on to enable dynamic library load of all necessary SPSS Modeler Solution Publisher libraries. The ***setupEnv.sh*** script has been provided in the root of the toolkit as an example of accomplishing this and could be sourced directly after editing the information unique to your SPSS Modeler Solution Publisher installation if you like.

IMPORTANT NOTE: 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. The best way to assure this is to source these settings in the same manner as InfoSphere Streams in all environments.

Two additional helper script files are placed in your toolkit by the install:

- **publishHelper.sh** – can be used to 'publish' Modeler files for use in InfoSphere Streams applications
- **passwordEncoder.sh** – can be used to encode the C&DS password when used in the operator configurations of your Streams applications

IMPORTANT NOTE: To use these sample scripts, you will need to set the **SPSS_TOOLKIT_INSTALL** environment variable to the directory where the SPSS Analytics Toolkit is installed, this is also demonstrated in the ***setupEnv.sh*** script.

4 How to use the SPSS Analytics Toolkit

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

To compile an InfoSphere Streams application using the SPL compiler command you must specify the toolkit install 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 the SPSS Analytics Toolkit to the STREAMS_SPLPATH makes the toolkit available by default in both in 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 SPSS Analytics Toolkit to your InfoSphere Streams Studio environment you would add it to the Toolkit Locations view of your InfoSphere Streams Explorer.

The 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
```

4.1 Operators

The operators in the SPSS Analytics Toolkit are all defined under the ***com.ibm.spss.streams.analytics*** namespace. To use this toolkit's operators in an InfoSphere Streams application you must include the following 'use' clause in your SPL source file:

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

You may 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.

4.1.1 The Scoring Operator

The SPSSScoring operator is an InfoSphere Streams 'generic' primitive operator. This operator's implementation will be optimized by your configuration through code generation to match the 'published' SPSS Modeler scoring branch it is configured to execute. Your application will score the data in the stream through this operator's integration with SPSS Modeler Solution Publisher.

This operator can be used without the other operators in this toolkit. To accomplish this, deployment metadata files must be generated for the new or modified scoring branch. This is done either by performing a 'publish' from an 'export node' from SPSS Modeler client or using the supplied 'publish' script included in this toolkit to produce the required PIM, PAR and XML files (description below).

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

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

The parameters for this operator are:

- **pimfile** – The full path to the executable image file generated by the publish of the SPSS Modeler file scoring branch
- **parfile** – The full path to the file of parameters to be used in preparation of the executable image file above, generated by the publish of the SPSS Modeler stream file scoring branch
- **xmlfile** – The full path to the XML file describing the inputs and outputs of the published SPSS Modeler stream file scoring branch, used to validate input parameters
- **modelFields** – A list of strings referencing the scoring branch input field names as defined in the input section of the XML file passed in the '**xmlfile**' parameter
- **streamAttributes** – A list of expressions defining the input tuple attribute expressions to be mapped to the '**modelFields**' in the order entered. Data types must match expected data types as defined in the XML file passed in the '**xmlfile**' parameter.
- **scoreOnWindowPunc** – An optional Boolean parameter indicating if the scoring branch is expecting a set of inputs 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 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 indicating if detection for 'not a number' (NAN) in floating point inputs should be 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 holding the data to be scored will flow. This is a non-windowed port (single tuple per score restriction on scoring branch) and will potentially mutate the attributes of the input tuple.

Windowing Notes: Although the scoring input port is marked as Oblivious to window punctuations in its configuration it will actually be expecting these punctuations in the data stream if you set 'scoreOnWindowPunc' to true. Also, final punctuation will 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's Publish operator (described below), the DirectoryScan operator from the InfoSphere Streams standard toolkit or other 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:

- **fromModel(attribute)** – Submits attribute referenced in this output function as returned by the scoring branch, values may 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 Notes:

I. All input attributes are submitted over the output port unless replaced by 'fromModel' output function expressions in the output configuration. Additional output attributes generated by the scoring branch that you want passed downstream should be specified in 'fromModel' expressions in your configured output specification.

II. When scoring in 'Window Punctuation' mode (scoreOnWindowPunc parameter true) only the first input tuple will be 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

As you can see in the configuration description above detailed knowledge of the inputs and outputs of the scoring branch is required. This information is communicated in the XML file generated during the publish operation.

Reminder: the input fields required to execute the configured scoring branch and the output fields it produces define the 'data contract' for a given configuration of this operator in your InfoSphere Streams application.

Input Contract:

The <inputDataSources> element of this XML file defines the input fields required for each data source of the scoring branch. **NOTE:** This release restricts this to one data source so the input fields of interest will all be listed in the <fields> element under the first <inputDataSource name="<node ID>" type="Delimited"> entry. For each <field> listed note the 'storage' value defining its data type and the 'name' defined.

Example input data contract description:

```
<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>
```

Output Contract:

The <outputDataSources> element of this XML file defines the output fields produced by the execution of this scoring branch. You indicate a single terminal node when publishing the scoring branch and so this section will always have a single <outputDataSource name="<node ID>" type="Delimited"> element. The output fields of interest will all be listed in the <fields> element under the of the first output data source entry. For each <field> listed note the 'storage' value defining its data type and the 'name' defined.

Example output data contract description:

```
<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>
        </inputDataSource>
</inputDataSources>
```

Notes on Date and Timestamp predictors in the scoring branch:

The InfoSphere Streams ***timestamp*** primitive data type is based on Epoch (00:00:00, January 1, 1970 UTC). The Modeler ***date*** and ***timestamp*** data types are also based on midnight January 1, 1970 but make no UTC adjustment. The most efficient way to pass InfoSphere Streams timestamp attributes into the SPSSScoring operator implementing scoring branch is by int64 seconds. Any adjustment to this value should be determined by investigation of the system settings of the compute nodes running the Streams application to avoid any unplanned time zone influence on this transformation during scoring.

Example usage:

In this example we are firing a set of input data to be scored sourced from a 'CSV' file. The SPSSScoring operator will listen for a new version to its predictive model and refresh its executable image without blocking the scoring of the data stream.

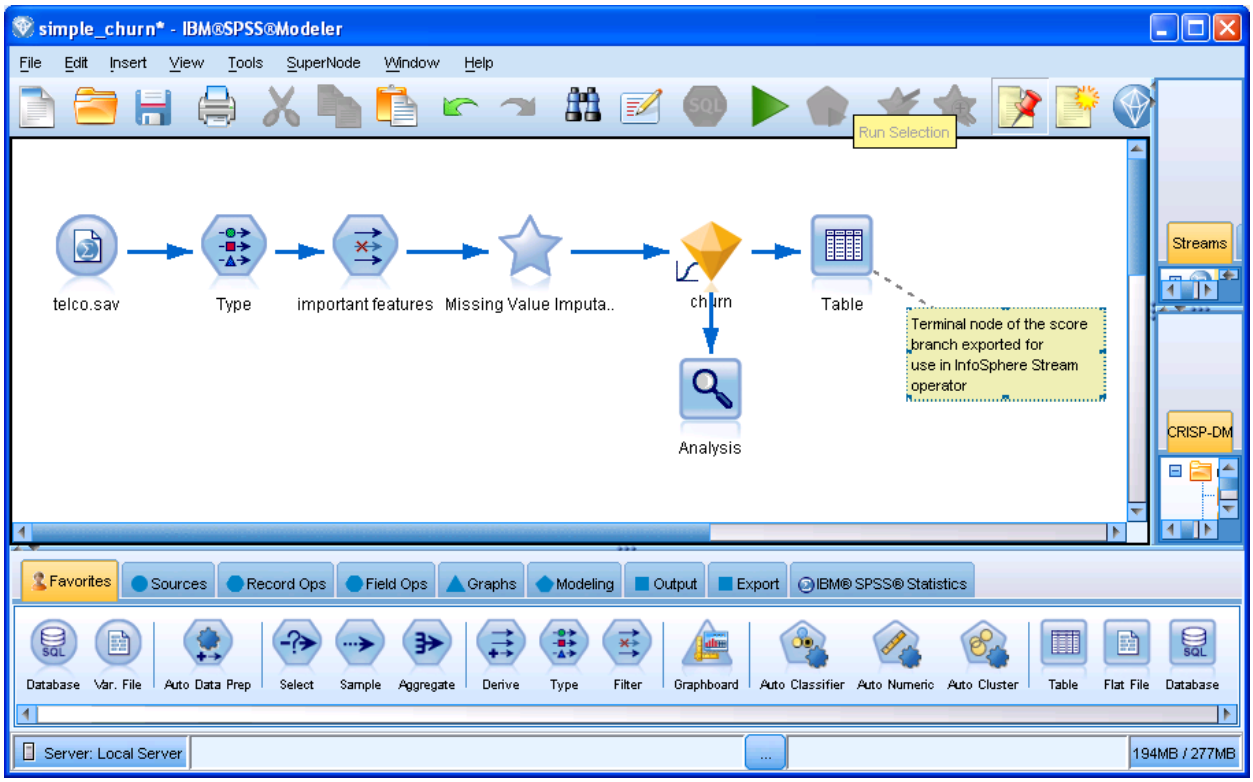
```
composite SPSSScoringExample {
        type
                static DataSchema =
                        rstring s_sex,
                        int64 baseSalary,
                        int64 bonusSalary;
                static DataSchemaPlus =
                        DataSchema, tuple<int64 income, 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 have been included in the operator model for the SPSSScoring for your convenience.

Scoring terminology and background information:

The term ‘score’ is to be taken to mean the act of executing the set of process nodes defined in the designed path of a SPSS Modeler file that implements the plan for producing the desired predictive analytics.

In the graphic below we can trace the scoring branch from a specific terminal node (‘Table’ as noted in the screen shot) to the left through the various process nodes to the source node ‘telco.sav’ in this example.



The illustration above highlights some important concepts:

1. A scoring branch seldom has a simple source / model nugget / terminal design and for this and other reasons we should avoid using the over-simplified term ‘predictive model’ or ‘model’ for an implementation of a scoring plan that will produce the desired predictive analytics.
2. 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 may radically change your scoring branch implementation and still use it in an InfoSphere Streams application configured against another version of your scoring plan as long as this data contract is maintained.
3. In this graphic the ‘churn’ node (yellow gem in display) is a ‘model nugget’ which is a predictive model constructed using data mining techniques in the ‘build branch’ of a

Modeler file. It is much more common to periodically 'retrain' the predictive models in a scoring branch using new data than it is to redesign the scoring branch itself.

4. There may be many processing branches in your SPSS Modeler file but you publish ONE scoring branch to prepare the executable image for use in your InfoSphere Streams applications.
5. To 'publish' the scoring branch for use in an InfoSphere Streams application leveraging SPSS Modeler Solution Publisher you perform an 'export' of the branch from the SPSS Modeler client or use the publish functionality provided by the toolkit. This generates the executable image ('.pim' extension) file, the execution parameters ('.par' extension) file and a XML file describing the required inputs and resulting outputs of the scoring branch.

4.1.2 The Forecast 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; used to trigger 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 using the default actual tuple defined in the configuration followed by window punctuation.

This operator exists to feed a SPSSScoring operator configured perform Time Series processing that needs to see both the actual measures and associated future predictor values for a given period of time; effectively doing a model refresh as it does its scoring.

The parameters for this operator are:

- **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 must be sliding windows.
- Port 0 – the tuples representing the actual values, this sliding window triggers output submissions
- Port 1 – the tuples representing the future predictor values, contents of this sliding window are used in matches to actuals and also define the additional future predictor outputs

Output Ports:

- This operator has one output port which accepts expressions referencing tuple attributes from both input ports.

Example usage:
In this example we'll take input from two Beacons, one representing the actual values from some sensor and the other the future predictor values generated from some source. We'll use a simple integer key value that we'll increment 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. We'll trigger the output from this operator on every actual input. Note the input tuple 'match' expression used to join the inputs 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;
                }
```

...
}

4.1.3 The Publish Operator

The SPSSPublish operator is a Java primitive operator that automates the 'publish' of a Modeler file's scoring branch and summarizes the generated files so down-stream operators can refresh their scoring implementation with the PIM, PAR and XML files 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 are:

- **sourceFile** – The fully qualified name of the SPSS Modeler file to be published
- **terminalNodeID** – ID of the terminal node in the SPSS Modeler file to be published that defines the scoring branch, required if the SPSS Modeler file has not been deployed with its scoring branch denoted in its meta data otherwise optional
- **targetPath** – Directory path the generated execution image (PIM), execution parameters (PAR) and execution description (XML) files are to be written to, optional parameter with default target path being the same as the source
- **cdsServer** – The address of the SPSS Collaboration and Deployment Services server, 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 referenced in the SPSS Collaboration and Deployment Services repository, required if 'cdsServer' is required
- **password** - Password of the user authorized to access the objects referenced in the SPSS Collaboration and Deployment Services repository, required if 'cdsServer' is required
- **encodedPassword** – an optional Boolean parameter that indicates if the password above has been encoded (true) using the mechanism supplied in this toolkit or not (false), if not specified it is assumed to be false
- **maxMemory** – an optional unsigned integer parameter that can be used to indicate the mega-bytes of memory to be allocated for the Java VM launched to perform the publish of the scoring branch

Input Ports:

- 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 will be published. This is a non-mutating, non-windowed input port.

Output Ports:

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

Example usage:

In this example we'll listen to the file changes in a specific directory in the file system and publish the configured SPSS Modeler file's 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);
        }
}
```

4.1.4 The Repository 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 SPSS Collaboration and Deployment Services repository. When a notification occurs indicating that the object this operator is configured to monitor has changed, the associated file version is retrieved from the repository and written to the configured target directory. On successful download an output tuple describing the file updated is submitted to communicate this event to down-stream operators.

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

The parameters for this operator are:

- **cdsServer** – the address of the SPSS Collaboration and Deployment Services server
- **userID** – ID of the user authorized to access the server and objects in the SPSS Collaboration and Deployment Services repository
- **password** - Password of the user authorized to access the server and objects referenced in the SPSS Collaboration and Deployment Services repository
- **resourceURI** – URI string referencing the Modeler file to be monitored in the SPSS Collaboration and Deployment Services repository

- **versionLabelName** – The name of the 'label' used to identify promoted resource versions to be monitored, if omitted any new version will trigger download
- **targetFilePath** – Path of the target directory to which file versions downloaded by this operator will be written
- **detectionPeriod** – Optional, detection period in seconds determining how frequently this operator looks through the notifications from the SPSS Collaboration and Deployment Services repository, if not specified an internal default of 10 minutes will be used
- **encodedPassword** – an optional Boolean parameter that indicates if the password above has been encoded (true) using the mechanism supplied in this toolkit or not (false), if not specified it is assumed to be false

Input Ports:

- None, this is a source operator

Output Ports:

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

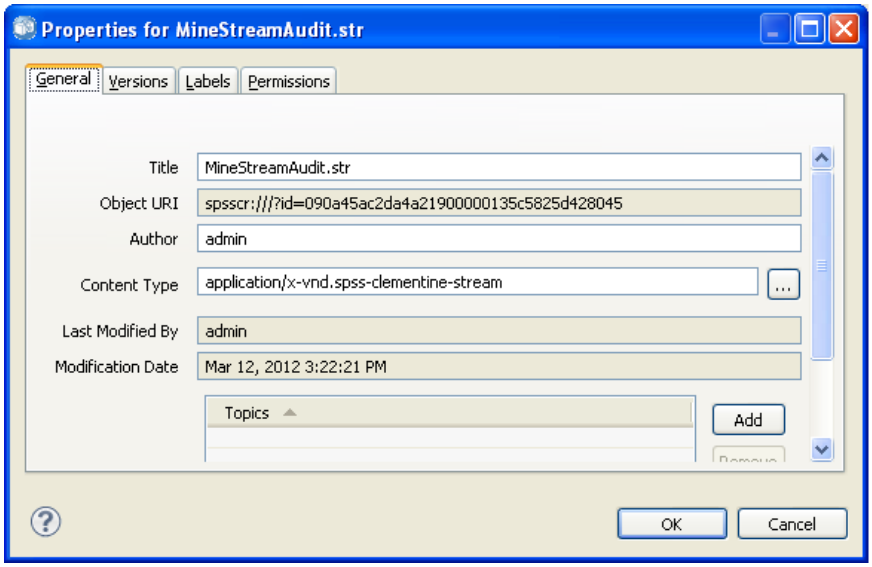
Example usage:

In this example we'll listen for notifications on the association of the label named 'PRODUCTION' to an SPSS Modeler file integrated in the repository under the URI " spsscr:///?id=09895272b9c1042e00000133fad8111192f4" (details on how to obtain this value will follow this example). When the notification is detected (default detectionPeriod) the file version will be 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);
        }
}
```

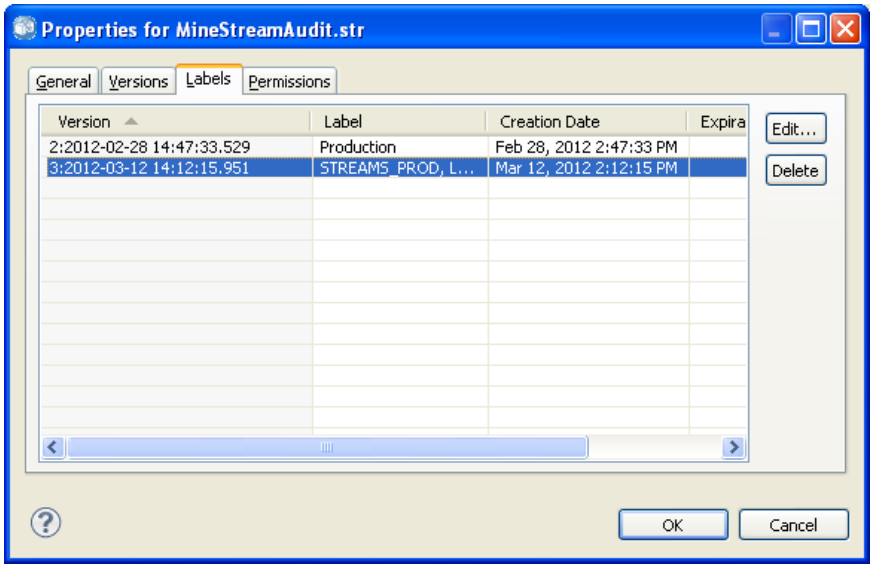
4.1.4.1 Determining the resource URI for the Modeler file managed in the SPSS Collaboration and Deployment Services repository

To get the URI to a given object in the repository use the IBM SPSS Collaboration and Deployment Services Deployment Manager client and right-click to get the pop-up menu that will give you the 'Properties…' option that will give you a detailed summary of the object. From this dialog you can easily copy the 'Object URI' value to paste into the configuration of your Streams application.



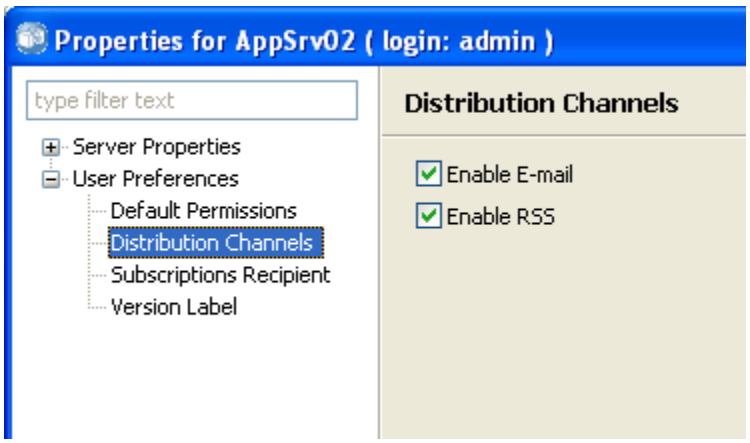
4.1.4.2 Best Practices for Notifications Monitored by InfoSphere Streams Applications

Use a meaningful Label name with clear and well communicated purpose on all Modeler files placed into production in your Streams applications. In the example below the STREAMS_PROD label is being used.

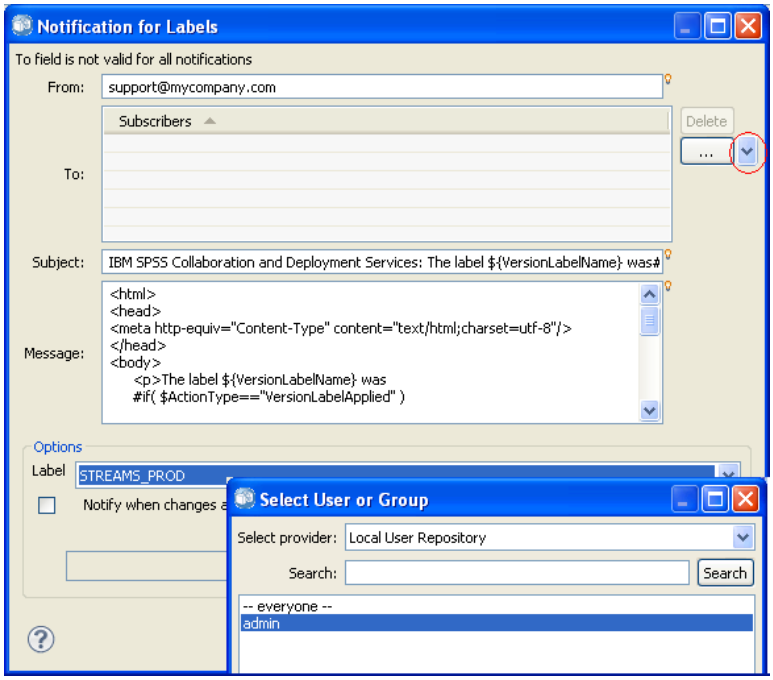


To use Label move notifications to indicate that a specific version of a Modeler file is to be used to refresh your running InfoSphere Streams applications you would set the label using the interface above.

You must also make sure that your user is configured to receive notifications over RSS distribution channel in their user preferences:



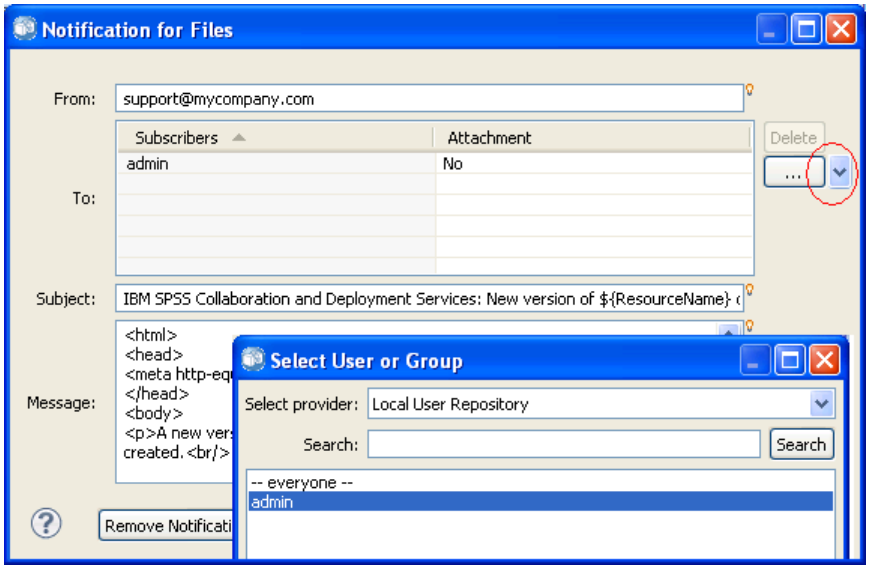
Finally you must indicate that notifications are to be issued whenever the label in question is 'set' or 'moved' on an object by right clicking on the Content Repository root folder in the display and taking the 'Label Events…' option. In the dialog presented you can add a 'Security Subscriber' of the user used to read the RSS feed for these 'label move' notification events.



Although not a 'best practice' some sites use the act of creating a new file version as a 'promotion' indicator. This is somewhat natural for a 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 promoted to the target repository. NOTE: the pseudo-label 'LATEST' will effectively 'move back' to a later version if

the most recent version of an object is deleted; always implicitly associated with the 'latest' version held by the repository.

If you are going to use file version creation notifications (Note: will not monitor delete events to 'move back' to a previous file version on a delete) instead of label based monitoring right click on the Modeler file object to be monitored selecting the 'Notifications…' option and define a 'Security Subscriber' notification type using the drop list circled in red in the snapshot below to make notifications available to the 'userID' authorized to these object in the Collaboration and Deployment Services repository.



5 Sample Applications

The IBM SPSS Analytics Toolkit for 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 info.xml file to describe the sample in InfoSphere Streams Studio and reference a common 'data' subdirectory with the sample data and other assets needed to run the sample.

NOTE: To use these samples you will have to publish the 'model.str' file generating the required PIM, PAR and XML files. The **publishHelper.sh** script has been included to help you do this by hand if you have not done this using SPSS Modeler client.

Short description of each sample:

- **SPSSScoring** – This sample application uses a simple SPSS Modeler scoring branch defined in the 'model.str' file, a small data file of data to be scored in the 'input.csv' file and creates an output file containing the predictions in the configured file sink.

- **SPSSForecastScoring** – This sample application uses a simple SPSS Modeler scoring branch defined in 'timeseries.str' and inputs generated by Beacon source operators feeding a SPSSForecast operator to pass a window of information in to the SPSSScoring operator with the outputs directed to a file sink.

- **SPSSPublishScoring** – This sample will 'publish' any SPSS Modeler file it is asked to according to its configuration and in turn trigger the 'refresh' of the scoring in the SPSSScoring operator. The sample does not define the SPSS Collaboration and Deployment Services repository connectivity information so the 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 all three operators from this toolkit; SPSSRepository, SPSSPublish and SPSSScoring. This sample will require an 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 drop a second version of the object into the repository if not using the label recognition to get the notification that will trigger the download and ultimately the 'publish' and 'refresh' operations. **NOTE**: this application will not terminate on its own as it continually monitors notifications from the SPSS Collaboration and Deployment Services repository.

5.1 Working with the samples in the command-line environment

To compile one of the samples from the command line, you will need to set the SPSS_TOOLKIT_INSTALL environment variable to the directory where the SPSS Analytics Toolkit is installed. You can then run make from within one of the samples subdirectories (e.g. SPSSScoring).

By default, the sample is compiled as a distributed application. If you wish 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”.

5.2 Working with the samples in InfoSphere Streams Studio

To import a sample into InfoSphere Streams Studio, you must first add the SPSS Analytics Toolkit to the Toolkit Locations section. To accomplish this, go to the **Streams Explorer**, right-click **Toolkit Locations** and select **Add Toolkit Location**. Enter the directory or click **Directory** to select the install location of the SPSS Analytics Toolkit, and click **OK**. This only needs to be done once.

After you have finished adding this toolkit location to your development environment, select **Import** from the **File** menu, expand the **InfoSphereStreams** folder, and select **SPL Project**. Enter the directory or click **Browse** to select the directory of the sample you wish to import, and click **Finish**.

Appendix A: Notices

This information was developed for products and services offered in the U.S.A. Information about non-IBM products is based on information available at the time of first publication of this document and is subject to change.

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma,
Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country/region where suchprovisions are inconsistent with local law:

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

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

IBM, the IBM logo, ibm.com, SPSS and InfoSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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