Release Notes

ALERI STREAMING PLATFORM 2.2

Aleri, Inc
2 Prudential Plaza
41st Floor
Chicago, IL
USA
+1 (312) 540 0100
Documentation E-mail: docs@aleri.com
Website: http://www.aleri.com

Published June, 2007

Table of Contents

List of Examples

Release Notes for Aleri Streaming Platform

Table of Contents

Introduction

These Release Notes describe the new features, enhancements, noteworthy bug fixes, any backward compatibility issues, and known issues and limitations for Release 2.2 of the Aleri Streaming Platform™. We also include this information for prior releases up to the last major release, Release 2.0. For a complete description of all the Platform functionality please refer to the various documents in the docset.

The Release 2.2 package consists of the following:

1. Aleri Streaming Platform software executables, scripts, utilities, libraries, and examples
2. User Documentation (PDF format)
3. Release Notes (this document, PDF format)
4. Installation and Configuration Guide (PDF format)
5. Installation Tool (`install.sh` script for Unix, InstallShield for Windows)

Major Features and Fixes for Release 2.2

Platform Server

- **RSA Authentication for High Availability** - RSA public/private key authentication was extended to include High Availability configurations.
- **Dynamic Service Modifications** - Allows for model changes to be made and introduced to a running Platform server without bringing the server out of service.

Platform Authoring

- **Secondary Join Indices** - Reintroduced the ability to specify a secondary index to be used for joins to fields or field combinations that are not the primary key.
- **SQL Scripting** - Stream retention windows were added for Unions.
- **Print Function** - A print function was added to print strings to the console. This is very helpful in debugging, especially in SPLASH.
- **Break & Continue Statements added to SPLASH** - Break and Continue statements were added to SPLASH scripting in Flexstreams. These are extremely helpful in debugging.

Platform Studio

- **Improved Syntax Checking Messages** - Improved the error messages provided in model and SPLASH syntax checking.
- **Pop-up Help Text for Platform Library Functions**.
- **SPLASH Editor** - Added GUI support for FlexStream SPLASH editing.
- **Aleri SQL Script Editor**- Added GUI support for Aleri SQL Script editing.
- **JavaScript™ Support** - Added support in Studio for JavaScript editing and execution for scripting Studio execution.
- **Most Recently Used Files -** Added support on toolbar to list most recently used files.

Platform Tools

- **SQL-based Subscribes** - Subscribes now supports select, project, and compute attributes on the `subscribe` statement.
- **Aleri Dashboard** - Real-time customizable dashboard for extremely versatile graphical visualizations of subscribed data from the Platform. This is packaged & sold separately from the Aleri Platform.
- **Aleri RT for Microsoft Excel®** - The following new features have been added:
  - improved packaging,
  - performance improvements,
  - added support for SQL-based Select, Project, and Compute to subscriptions
  - added digital signature.

Major Features and Fixes for Release 2.1

Platform Server

- **Online Backup** - A consistent backup copy of the log store files can now be created on a running Platform instance with the **sp_cli** backup command.
- **Memory Improvements** - A number of optimizations have been made to reduce memory consumption.

Platform Authoring

- **Java™ Call-out Functions** - The ability to call user defined Java functions has been added and implemented through the JNI and existing shared libraries.
- **Retention Windows Element** - Data retention on BaseStreams and CopyStreams has been generalized to a concept of retention windows on input streams. Currently all streams in memory and stateless stores, except for JoinStream and FlexStream™, support the retention window element. This restriction (on JoinStream and FlexStream) will be removed in a future version of the Platform, but can currently be worked around by front-ending the stream with a CopyStream that has a retention window defined on it. For streams defined that use a log store, only CopyStreams and BaseStreams may have retention windows. This restriction will not go away, and is a result of the fine-grained recovery capabilities inherent in the Platform's log store technology. This last restriction on streams defined with log stores also may be easily overcome by front-ending a stream with a CopyStream that has a retention window defined on it. If a stream has multiple inputs, an optional retention window element may be defined on each of the stream's inputs.

Platform Studio

- **Studio Dashboard** - A dashboard has been added that will show the results of a single stream's numeric column calculated as a percent of the total value for all rows in the stream. Multiple dashboards can be created for one or more streams.
- **Palette** - A Memory Store, Log Store, and Stateless Store have been added.
- **Studio Online Help** - Studio is now shipped with the complete table of contents for online help already generated.
- **Search Model Action and Window** - A new search feature that allows a user to search a model for string properties contained within elements, and then display the results in a new Search Window, has been added. The matching elements can then be modified individually using the Properties Window, or by selecting multiple matching items in the Search Window and performing a replace action.
- **Expression Editor** - The expression editor (Editor Block) has the following new features:
    - For a FlexStream method, the expression editor now displays the line and columon position of the cursor. Also, when a syntax check is performed, the cursor will be positioned on the line where the error occurred, if any.
    - When editing a column expression for a ComputeStream, JoinStream or AggregateStream, the expression editor will display the corresponding column name, if any.
    - Hot keys have been added to display help, stream columns, data types, operators, local columns, variables, and to save the expression(s).
- **Version Control Header** - A new user preference has been added to Preferences, for adding a version control header into a newly created Aleri XML file.
- **Allow or Disallow Duplicate Diagram Elements** - A new user preference has been added to Preferences, to allow duplicate elements on a diagram.
- **Toolbar** - The following new features have been added:
    - A new button has been added to allow a user to quickly layout a diagram as top-down, or from left to right.

    - A new zoom button has been added.
- **Related Streams Feature** - Using the context menu, a user can now select a stream and expand the diagram to include one or all sources and targets for the selected stream.
- **Execution Prespective Optional Arguments** - An "Additional Options" text box was added that can be used for additional command line arguments to be passed to the Aleri Streaming Platform Server.
- **Join Constraints Editor** - A Join Constraint Editor allows a user to edit a join constraint by selecting columns of two streams.
- **Required Properties** - Properties that are deemed "Required" in the Platform.xsd file are now prefixed with an asterisk "*" in the Properties Window.
- **Stream Insert Action** - When a FlexStream, JoinStream, or UnionStream is selected a user will now have the option to add a new stream as the target stream using the Collection Window or the context menu. For an AggregateStream, CopyStream, ComputeStream or FilterStream, a user will have the option to select a stream as the target stream using the context menu or Properties Window only (since these stream types only have one input stream, there is no Collection Page for these streams).
- **Unassign Action** - In verbose mode, when a target stream has an input stream compartment selected, the user will have an option to delete the istream for the selected target stream, removing the istream for the selected target stream. This feature is also available in the Collection Windows for supported streams. This same action is available to unassign the Store or RowDef of a selected stream.
- **Quick Image Toggling** - A user can now double-click any shape in a diagram, above the label, to toggle an image from an iconic to a verbose display.
- **Retention Window Support** - For any stream (except a JoinStream, FlexStream and AutoStream), a user can add, edit, or delete a Retention Window.
- **Context Menu Changes** - The context menu has been reorganized to group features and for easier access to functionalities.

Platform Tools

- **Aleri RT for Microsoft Excel®** - The following new features have been added:
    - Auto Publishing - AleriRt can now automatically publish data to the Aleri Streaming Platform Server from an Excel spreadsheet whenever data changes in a range of cells are detected.
    - Manual Publishing - A user can now manually publish data to the Aleri Streaming Platform Server by selecting a range of cells and using the AleriRt Publish Wizard.
    - Hot Failover Support - An optional entry that specifies the machine on which an Aleri Streaming Platform Secondary Hot Spare Server is running has been added. Therefore, when the primary server fails, data is automatically retrieved from the secondary server.
    - RSA Authentication
    - RTD Wizard - AleriRt now has an improved RTD Wizard, including an RTD filter that can be specified by using a combination of the Filter table, Max Rows, and Window Size properties.
- **Data Archiving** -

- The **sp_archive** executable now archives data from one or more streams in the Platform to Sybase IQ™ either in batch mode or in real-time.
- **sp_archive** now supports error recovery in insert, update, and delete modes.
- **sp_archive** now uses the persistent subscribe pattern to support error recovery instead of using the record rowid mechanism.
- **sp_archive** also now supports RSA authentication.

Major Features and Fixes for Release 2.0.1d

Platform Server

- **XML Schema File Specification** - Until now the XML Schema definition (.xsd) file path had to be specified in XML authoring (.xml) files. Now one can also specify the schema path via a `-F` parameter when starting the Platform Server. One can also set the `PLATFORM_HOME` environment variable; it will be used to find the XML Schema definition file. The priority order that the Platform commands use to find the .xsd path specification is: `-F` command parameter, `PLATFORM_HOME` environment variable, and then inside the XML authoring file.
- **Hot Failover** -
    - The recovery time may vary depending upon any large difference in data between the primary and secondary servers.
    - A race condition in High Availability (HA) mode has been fixed that could cause synchronization to stall and not finish. This could cause the secondary node to crash when the primary node failed.
    - An authentication issue with HA. In HA mode, both the primary and the secondary servers are required to authenticate on both the Command and Control and Gateway interfaces to properly establish and maintain data replication from the primary to the secondary. The authentication on the Gateway interface used an uninitialized variable, and thus was likely to fail (unless authentication was turned off at the PAM level).
- **User Password** - The user and password are no longer logged as messages on the system log files.
- **Persistent Subscribe** -
    - The persistent subscribe pattern created in Studio 2.0.1 is not correct. It is possible to get into a scenario where publishing valid data into the truncate side of the FlexStream would incorrectly fail to truncate the log stream. Studio 2.1 resolves this by modifying the `truncateMethod`, replacing
    - `    i := Trades_truncate.sequenceNumber;`

    -
      ```
                        if ((low &lt;= i ) and (i &lt; high)) {
                         while (low &lt;= i) {
      ```

    with

    ```
    i := Trades_truncate.sequenceNumber;
                   if (i &gt;= high)
                       i := high - 1;
                    if ((low &lt;= i ) and (i &lt; high)) {
    ```

```
while (low &lt;= i) {
```

    - The sequence number data type was changed to int64 (64-bit integer).

Platform Authoring

- **FlexStream** -
    - FlexStream has been enhanced to allow access to its internal records. Prior to this, a FlexStream could only access records from its input streams.
    - The RowLocalStorage capability, which existed in ComputeStream, has been added to FlexStream as well. This allows one to keep local variables for groups of records (e.g., moving average for each stock symbol).
- **Persistent Subscribe** - With the enhancement above, FlexStream can now be used to model persistent subscribes. Persistent subscribes augment normal subscribes by providing a persistent log of all subscribed data for a single user with sequential unique tags, so that the subscriber can easily restart a previous subscribe where it left off. Persistent subscribe of an existing stream is supported via two additional stream instances (one FlexStream and one BaseStream) stored in a Log Store for persistence. A sample model of persistent subscribe was added to the examples. Studio added a helper function (i.e., pattern) that allows you to click on a stream and right click to select that you want a persistent subscribe pattern for this Stream generated automatically.

Platform Studio

A number of small fixes were made.

- Studio expression & method dialog content assists were too small (and dialog was too small).
- Subscribe mechanism was using US ASCII for string encodings instead of UTF-8.
- Removed byteswap option from Studio upload page given remote execution executes models existing on its machine, hence no need for byteswap support.
- Studio performance monitor did not always shutdown properly.
- Added gesture to Sync keys of UnionStream with its input stream.
- Removed `xsl:noNamespaceSchemaLocation="/PATH/Platform.xsd"` when creating new .xml file from Studio.
- Added line and column number labels to FlexStream method editor.
- Studio FileSaveAs did not save if there was a missing `.notation` on the end.

Platform APIs

- **Memory Leaks** - A memory leak was fixed in the C++ publish API, and in the Microsoft® .NET publish API.

Major Features and Fixes for Release 2.0

Platform Server

- **Distributed Services** - The Aleri Streaming Platform now provides the ability to author distributed models by defining modules (i.e., model subsets) and clusters (modules to machine/node mappings). The primary purpose of this feature is to enable large and complex models to scale across multiple servers. The new tool **sp_clustermgr** supports the management of distributed services.
- **Cold Spares** - When using distributed services, if there is a failure of an active node, idle spare nodes(i.e. Platform instances) can be brought into active use in a distributed cluster. If persistence is utilized in distributed services, then all cluster nodes must have access to a common disk share. The tool **sp_clustermon** supports the active management of Cold Spares.
- **High Availability** - Any authored model that is designed to run on a single Platform instance (i..e not the distributed services) may now be run with a hot failover for high availability. Aleri's pub/sub APIs (Java, C++ and .NET) have full support for high availability.
- **Access Control** - Models can be defined to restrict access to individual streams and Platform instances on the basis of a user's roles. User role membership is defined via Unix or NIS groups. Access can currently be restricted for the On-demand Query Interface, Subscribe (not Publish) Interface, and Command & Control Interface.
- **RSA Authentication** - RSA authentication is now supported (in addition to previously existing username/password based authentication) to enable client access through a private/public key (similar to SSH). RSA authentication removes the need to put clear text passwords in system management scripts.
- **Version Checking** - The versions of authored models are now checked against the Platform version to ensure that the XML Schema Definitions have not changed. In situations where the XML Schema Definitions have changed, a warning message is generated to notify the user that the model needs to be updated. In future releases, automatic model upgrades will be provided when changes are made to the XML Schema Definition.

Platform Authoring

- **FlexStream** - The Aleri Streaming Platform now provides a programmable stream called a FlexStream. Users can write small programs in a language called SPLASH, which resembles the C language. These programs respond to input events (records with insert, update, or delete operations) and produce output events. This feature gives (programmer-oriented) users the ability to build specialized streams for their specific application needs when existing stream primitives don't efficiently provide what's needed. In addition to Aleri XML-based and Studio-based authoring, FlexStreams are available within Aleri SQL Authoring using Program Views.
- **Model Modularity** - Models can be separated into multiple modules (albeit a single authored XML or SQL definition/file with modules defined within that definition), similar in definition to distributed services, but also for non-distributed services. For testing & debugging purposes, a single module can be executed as long as its Source Streams do not contain references to streams in other modules (as can be the case for

distributed services only). This feature can be used to divide large, complex models in functional modules for readability, maintainability, and testability.

- **AutoStream Heartbeat** - AutoStreams (i.e., Insert-only streams with optional filtering) now support a heartbeat attribute that will generate a new record with all non-key data fields NULL every N specified seconds. One potential use is to have an AutoStream, with heartbeat, feeding a FlexStream so that the FlexStream can periodically check for conditions of interest in the AutoStream data (e.g., detection of non-events).
- **RowLocalStorage (RLS)** - RLS was added to Aleri SQL Authoring (i.e., it already existed in Aleri XML and Aleri Studio authoring). RLS is an indexed collection or sets of values, which allows for maintaining state in a ComputeStream and correlating arbitrary records based on the data contained in them. There may be one set for every row in the stream if no keys are provided or one set for every unique combination of key values if one is provided.
- **New Library Functions:**
    - **avg** - The "avg" function finds the average of the non-null values in the set.
    - **cbrt** - The cube root function accepts a single numeric argument and produces the cube root of the argument as a double.
    - **ceil** - The "ceil" function accepts a single double precision argument and rounds it up to the nearest integer value, returned as a double.
    - **cosine** - The "cosine" function accepts one parameter of type double (specified in radians), and returns a double representing the cosine of the specified parameter.
    - **exp** - The "exp" function accepts one argument (call it "x") of type double, and produces a double, the value of which is the base of natural logarithms raised to the power of x.
    - **floor** - The "floor" function accepts a single double precision argument and rounds it down to the nearest integer value, returned as a double.
    - **length** - The "length" function accepts one parameter of type string, and returns an integer value representing its length.
    - **ln** - The "ln" function accepts a single argument (call it "x") of type double, and returns the natural logarithm of x as a double.
    - **log** - The "log" function accepts a single argument (call it "x") of type double, and returns the base 10 logarithm of x as a double.
    - **lower** - The "lower" function accepts one parameter of type string, and returns a new string with all of the characters represented in lower case.
    - **lwm_avg** - The "lwm_avg" function finds the linearly weighted average of the non-null values in the set. If there are N non-null values in the set, the most recent is weighted N, the next (N-1), and so forth; the least recent value is weighted 1. The sum of the weighted values is divided by the sum of N, (N-1), ..., 1.
    - **sine** - The "sine" function accepts one parameter of type double (specified in radians), and returns a double representing the sine of the specified parameter.
    - **sqrt** - The square root function accepts a single numeric argument and produces the square root of the argument as a double.
    - **stddev_pop** - The "stddev_pop" function finds the population standard deviation of the non-null values in the set.
    - **stddev_samp** - The "stddev_samp" function finds the sample standard deviation of the non-null values in the set.

    - **tangent** - The "tangent" function accepts one parameter of type double (specified in radians), and returns a double representing the tangent of the specified parameter.
    - **upper** - The "upper" function accepts one parameter of type string, and returns a new string with all of the characters raised to upper case.

Platform Studio

- **Solaris™ x86** - Aleri Studio is now available on the Solaris x86 operating system.
- **Improved Welcome Screen** - The Aleri Studio has an improved Welcome Screen, which allows the user to view documentation or start off authoring with one of the packaged example models. The Welcome Screen is automatically displayed on startup and can be disabled when desired.
- **Data Input View** - The new Data Input View provides the ability to publish data manually (by means of an Insert, Update, Upsert, or Delete) to the Aleri Streaming Platform by selecting the BaseStream or AutoStream from a drop down list.
- **Collection View** - A new Collection View has been added to allow a user to select a stream or RowDefinition and display, edit, or delete the child elements of the selected item in this view. Each child element also has a corresponding icon displayed in the first column of the element's details.
- **Multiple Streamviewers** - Multiple streams can now be viewed concurrently from Aleri Studio (i.e., previously only single streams could be viewed).
- **Expression Editor** - An expression editor (i.e., Editor Block) can be easily launched from the context menu for a ColumnExpression, Rule InLine Expression, or FlexStream method. Users can now perform in-line editing at the cursor position by executing Ctl-Spacebar to insert supported functions and syntax, or Shift-Spacebar to insert the column names of Input Streams when editing a ColumnExpression.
- **Extensive Help Features** - A suite of Help features have been added to Aleri Studio, allowing the complete documentation for the Aleri Streaming Platform and Aleri Studio to be easily accessible from the authoring environment.
    - **Tool Tips** - Tool tips have been added to the: 1) Platform Command Execution 2) Problem, Collection, and Data Input views 3) Palette 4) Platform Monitoring 5) Platform Streamviewer 6) Platform Record-Playback and 7) Platform Debugger.
    - **Help Contents** - A user can now access extensive online help (launched in a separate window) for both the Aleri Streaming Platform and Aleri Studio. This feature enables a user to search for help in any Aleri documentation, including the Installation and Configuration Guide, Platform Overview, Authoring Guide, Guide to Programming Interfaces and the Administrator's Guide.
    - **Search View and Dynamic Help** - This new view, displayed inside Aleri Studio, allows that same extensive access to online help as the Help Contents.
    - **Context Help** - Selecting any shape on a Platform Authoring Diagram, and then pressing F1 (for Windows) or Control-F1 (for Unix) immediately displays help for the selected item in the Search View. This feature is also available in the Platform Command Execution, Data Input, Platform Monitoring, Platform Streamviewer, Platform Record-Playback and Platform Debugger views.

Platform Tools

- **Sybase IQ™ Migrations** - The Sybase IQ migration tool, **sp_archive**, was updated to deal with error recovery in the case of Insert migrations. A future release of the tool will also deal with error recovery for delete, update and upsert migrations.

Backward Compatibility Issues

The following notes describe how to convert previous XML authoring files into newer ones. Platform users who also migrate data to Aleri's OLAP Historical Database should consult the Utilities Guide; the **sp_histport** command requires changes to the data migration configuration file. New users of the Platform can skip this section.

Release 2.0 to 2.1

The XML Authoring configuration file format was changed in Release 2.1 for the changes in data retention.

Here is an example of the old format:

Example 1. Sample XML Configuration for Retention, Version 2.0

```
<?xml version="1.0" encoding="UTF-8"?>

<Platform xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="2.0">

    <Store id="store" file="store" />

    <RowDefinition id="inputRowDef">
      <Column name="a" datatype="int32" />
      <Column name="b" datatype="string" />
    </RowDefinition>

    <BaseStream id="input"
       store="store"
       rowdef="inputRowDef"
       retentionType="record"
       retentionValue="1000"
       retentionSlop="10"
       keys="a" />

    <CopyStream id="copy"
       store="store"
       rowdef="inputRowDef"
       istream="input"
       retentionType="time"
       retentionValue="30"
       keys="a" />
</Platform>
```

Following is an equivalent configuration in the new format:

Example 2. Sample XML Configuration for Retention, Version 2.1

```
<?xml version="1.0" encoding="UTF-8"?>

<Platform xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="2.1">

    <Store id="store" file="store" />

    <RowDefinition id="inputRowDef">
      <Column name="a" datatype="int32" />
      <Column name="b" datatype="string" />
    </RowDefinition>

    <BaseStream id="input"
       store="store"
       rowdef="inputRowDef"
       keys="a">
    <RetentionWindow type="record" value="1000" slack="10"/>
    </BaseStream>

    <CopyStream id="copy"
       store="store"
       rowdef="inputRowDef"
       istream="input"
       keys="a" >
    <RetentionWindow stream="input" type="time" value="30"/>
    </CopyStream>
</Platform>
```

Changes from 1.0 to 2.0

The XML Authoring configuration file format was changed in Release 2.0 to improve Studio authoring usability. It includes version numbering checking, improvements in the direct use of expressions (as opposed to Rules), and a change in sub-expressions from if-then-else to case-when-then-else-end.

In order to import existing models from releases prior to Release 2.0 into Aleri Studio, or to run models, users will need to manually convert the XML files.

Here is an example of the old format:

Example 3. Sample XML Configuration, Version 1.0

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Platform xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <Store id="store" file="store" />

    <RowDefinition id="inputRowDef">
      <Column name="a" datatype="int32" />
      <Column name="b" datatype="string" />
    </RowDefinition>

    <BaseStream id="input"
                store="store"
                rowdef="inputRowDef">
      <Order keys="a" />
    </BaseStream>

    <ComputeStream id="compute"
                   store="store"
                   istream="input"
                   rowdef="inputRowDef">
      <Order keys="a"/>
      <Compute rule="copyval.a" />
      <Compute rule="copyval.b" />
    </ComputeStream>

<Rule id="copyval.a" type="int32" params="v:inputRowDef">
     <InlineExpr value="if v.a then 1 else 2"/>
</Rule>

<Rule id="copyval.a" type="int32" params="v:inputRowDef">
      <InlineExpr value="v.b"/>
</Rule>

</Platform>
```

Following is an equivalent configuration in the new format:

Example 4. Sample XML Configuration, Version 2.0

```
<?xml version="1.0" encoding="UTF-8"?>

<Platform xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="2.0">

    <Store id="store" file="store" />

    <RowDefinition id="inputRowDef">
      <Column name="a" datatype="int32" />
      <Column name="b" datatype="string" />
    </RowDefinition>

    <BaseStream id="input"
                store="store"
               rowdef="inputRowDef"
               keys="a" />

    <ComputeStream id="compute"
            store="store"
            istream="input"
            rowdef="inputRowDef"
            keys="a">
      <ColumnExpression value="call(copyval_a)" />
      <ColumnExpression value="input.b" />
    </ComputeStream>

<Rule id="copyval_a" type="int32" params="v:inputRowDef">
   <InlineExpr value="case when v.a then 1 else 2 end"/>
    </Rule>

</Platform>
```

In order to convert the older XML configuration into the new format, one needs to perform the following steps:

1. Add the “version” attribute to the `<Platform>` declaration.
2. Remove the `<Order .../>` declarations and move the “keys” attribute into the definition of the stream.
3. Remove the `</BaseStream>` end marker and use a `<BaseStream ... />` form of the declaration. A stream definition that does not contain any sub-elements, i.e., BaseStreams, CopyStreams, and UnionStreams, will all use a similar form of declaration. AutoStreams that perform no filter must also take the form

   ```
   <AutoStream id="..." ... />
   ```

4. Make the following changes:
   - Change

     ```
     <Compute rule="..." />
     ```

     to

     ```
     <ColumnExpression value="call(...)"/>
     ```

   - Change

     ```
     <Compute rule="..."/>
     ```

     to

     ```
     <FilterExpression value="call(...)"/>
     ```

    - Change

      ```
      <Group rule="..."/>
      ```

      to

      ```
      <Group value="call(...)"/>
      ```

    - Change

      ```
      <GroupOrder rule="..."/>
      ```

      to

      ```
      <GroupFilter value="call(...)"/>
      ```

    - Change

      ```
      <GroupOrder rule="..."/>
      ```

      to

      ```
      <GroupOrder value="call(...)"/>
      ```

5. Notice that `Compute` changes into either `ColumnExpression` within ComputeStreams, Join Streams, and AggregateStreams, and into `FilterExpression` within Auto Streams and Filter Streams. One can use any expression within these elements and not call a rule. So, for instance, one can write
6. `<ColumnExpression value="InputStream.f + 1"/>`
7. if there is a column named “f” in the Input Stream "InputStream".
8. Eliminate periods in the names of rules. Periods in names are no longer valid, because they clash with the period used in the syntax of expressions (for field lookup).
9. Change subexpressions of the form

   ```
   if ... then ... else ...
   ```

   to

   ```
   case when ... then ... else ...end
   ```

Known Issues and Limitations

Platform Server

- Performance monitoring causes a small memory leak on the server.
- PAM authentication on Linux via the `pam_unix.so` module has a small memory leak.

- Distributed processing (i.e., clusters) does not currently support RSA authentication, only authentication via username & password.
- For streams in a memory store, retention windows are allowed on all streams except (currently) Join and Flex streams. Workaround: use a Copy stream in front of the Join or Flex stream and provide a retention window to the Copy stream. This limitation will be removed shortly.
- For streams in a log store, retention windows are allowed only on Base & Copy streams. Workaround: use a Copy stream in front of any derived stream and provide a retention window to the Copy stream.
- For dynamic service changes:
  - To modify the "file" attribute of a log store in the current release, you must specify a new not-yet-existing directory for its new location. In the next release you'll be able to specify an existing directory as long as there are no log stores in it.
  - The current release does not support dynamic modifications on the clustered or high-availability configurations. These restrictions will be removed in future releases.
  - Any dynamic modification invalidates all the registered on-demand SQL queries.
- To successfully run `install.sh` to install the Platform, you must specify the absolute, fully qualified paths to the installation directory and the Platform package.

Platform Authoring

- The 2.2 release of the Platform ships with a 2.2 version of the .xsd file (used to parse .xml files, including the data model and other related Platform files). Release 2.2 also ships with a complete set of example .xml files; these files explicitly reference the 2.2 version of the xsd file. This reference is usually in the second line of the .xml file, and looks like the following example:
-
-
- `            <Platform xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" \`
- `            version="2.2">`
-

  The example .xml files included with the previous release of the Platform (release 2.1) explicitly reference the 2.1 version of the .xsd file. (The corresponding line in these files ends with `version="2.1"`). The 2.1 examples will not with on an installation of the 2.2 release of the Platform.

  More importantly: if you have developed data models and other .xml files by editing the 2.1 examples, and these files still say `version="2.1"`, they will not work on an installation of the 2.2 release of the Platform.

  There are two ways to fix .xml files that do not work because of this imcompatibility:

    - Change this line so that it reads `version="2.2"`.
    - Remove the `version="2.1"` element entirely.
- The **sp_sql2xml** translator
    - traps most but not all errors before generating the XML. The untrapped errors will be caught by the Platform when it loads the XML for processing. The User Guide describes the errors that are not trapped under the section “Limits on Verification” in the **sp_sql2xml** section of the Utilities Guide.
    - does not verify that all the Row Local Storage elements used in a rule are defined. It also does not verify if the data types of the expressions used in the RowLocalStorage manipulation functions actually matches the data type of the RowLocalStorage.
    - does not ensure that key fields are consistent between a target table and input table. If the model contains such inconsistencies, however, the Platform will detect them and reject the generated xml model.
    - does not support Secondary Index definition.

Platform Studio

- When editing a SQL file or script, the "save as" option is inactive. This feature will be added in the next release. You can currently name the file when creating it, so you can effectively do a "save as" by creating a new file, copying the contents of the original into it, and then saving the new file with a new name.
- Studio search currently only searches the open diagram, in future releases it will also search all open editors.
- Streamviewer's "Get Stream" function currently returns all streams including internal streams. Future releases will enable you to filter out the internal streams so the list will not be so cluttered.
- Upon exiting Studio immediately after uploading data in the execution perspective, you may see an error similar to following. This is not a serious error and is not associated with any data loss in Studio.
- `ENTRY org.eclipse.osgi 4 0 2007-01-05 18:14:30.029`
- `    !MESSAGE Application error !STACK 1`
- `org.eclipse.swt.SWTException: Widget is disposed`
- `at org.eclipse.swt.SWT.error(SWT.java:3374)`
- `at org.eclipse.swt.SWT.error(SWT.java:3297)`
- `at org.eclipse.swt.SWT.error(SWT.java:3268)`
- `at org.eclipse.swt.widgets.Widget.error(Widget.java:434)`
- `at org.eclipse.swt.widgets.Widget.checkWidget(Widget.java:372)`

```
...(abbreviated error log message )
```

- RSA authentication is not a supported feature of Studio execution.

Platform Tools

- Streamviewer (**sp_viewer)**:
    - is not certified to work under X-Windows with the server running on a Solaris 10 machine.

    - requires Sun's version of Java for execution. This is a problem on the 64 bit Linux version of the Platform, which ships with IBM's version of Java. In order to get **sp_viewer** to run on a machine that has the 64 bit Linux version of the Platform installed, install Sun's version of Java 1.4.x on the machine. Then, make a copy of the `sp_viewer` script and modify the script, changing

      ```
      ${PLATFORM_HOME}/authoring/eclipse/jre/bin/java
      ```

      to point to Sun's version of Java 1.4.x instead of IBM's. Use this new script to execute the stream viewer.

- Excel connectivity via Aleri RT for *Microsoft Excel®*:
    - The use of multiple Excel Workbooks (i.e., not Worksheets) within the same instance of Excel is not currently reliable and may cause unpredictable results.
    - The total length of an AleriRT query statement is currently restricted to 32K.
    - AleriRT cannot subscribe to a stream with more than 64 columns.
    - Using a MaxRows value greater than a few hundred rows can cause the CPU to get pegged for extended periods of time when the Platform is pushing data at high rates.
- The **sp_monitor** command currently does not support encryption.
- The **sp_archive** command (which archives data to Sybase IQ™) may not keep up with the Platform if the data has a high percentage of updates & deletes (because it must apply them to Sybase IQ using SQL statements via ODBC, which is inherently slow). Inserts are applied via micro-batches and so are performed relatively quickly.

Platform APIs

- Writing Java client applications that use RSA authentication against the Aleri Streaming Platform: below is a code sample showing how to encrypt the login message using the private RSA key file of the user. Please see the appropriate section of the Administrators Guide for setting up the keys and Java environment.
- `  /*`
- `   * This method can be used by a Java client to create the RSA encrypted`
- `   * login message expected by the Aleri Streaming Platform Server.  The`
- `   * method returns the encrypted message as a String.  The String can then`
- `   * be used to RSA authenticate to both the Platform's Command and Control`
- `   * Process, and the Gateway I/O Process.`
- `   *`
- `   * The client application encrypts the login message using the private`
- `   * RSA key file of the user, and sends it to the server.  The server then`
- `   * attempts to decrypt the message using the user's public rsa key file,`
- `   * and compares the decrypted message to the expected result.  If they are`

```
   * equal, authentication is granted by the server.
   *
   * NOTE: For Java to read in an RSA private key file (originally
generated
   * from openssl), the private key has to be stored as PKCS8 in a DER
   * (binary) formatted file.
   *
   * NOTE: The server is using the C/C++ OpenSSL PEM libraries for the
RSA
   * functionality.  In the Java client, we had to encrypt the login
   * message, and then base 64 encode the result.  We also had to insert
   * line feed characters into the base 64 encoded message, in order to
get
   * the RSA authentication process to succeed (see the corresponding
   * "for-loop" code below).
   *
   */
  static String encryptPrivateRsaLoginMsg(String privateRsaKeyFile) {
    String rsaLoginMsg = null;
    if (privateRsaKeyFile == null) {
      return null;
    }

    File pkFile = new File(privateRsaKeyFile);

    if (pkFile.exists()) {
      FileInputStream in = null;
      try {
        /*
         * Read in the private rsa key file in the PKCS8 DER format
         */
        in = new FileInputStream(privateRsaKeyFile);
        byte[] privKeyBytes = new byte[in.available()];
        in.read(privKeyBytes);
        in.close(); in = null;

        PKCS8EncodedKeySpec privSpec =
           new PKCS8EncodedKeySpec(privKeyBytes);

        /*
         * For Java 1.4.x, we are using the Bouncy Castle Provider.  The
         * provider must be installed in the java
         * ./jre/lib/security/java.security file
         * and the Bouncy Castle provider jar file must be specified on
         * the Java classpath of the client application.
         */
        KeyFactory keyFactory = KeyFactory.getInstance("RSA", "BC");
        RSAPrivateKey privKey =
          (RSAPrivateKey)keyFactory.generatePrivate(privSpec);
        Cipher ecipher = Cipher.getInstance("RSA/NONE/PKCS1Padding", "BC");
        ecipher.init(Cipher.ENCRYPT_MODE, privKey);

        /*
         * This is the code that encrypts the login/authentication message
         * that is expected by the server.
         */
        String msg = "Aleri Streaming Platform RSA Authentication";

        ByteBuffer tt = ByteBuffer.allocate(100);
        for (int i = 0; i < 100; i++) tt.put(i, (byte)0);

        byte[] pw1 = msg.getBytes();
        tt.put(pw1, 0, pw1.length);
        byte[] encrypted = ecipher.doFinal(tt.array());
        byte[] pw3 = Base64.encode(encrypted);
        byte[] pw4 = new byte[400];
        pw4[0] = pw3[0];
        int j = 1;
        for (int i = 1; i < pw3.length; i++) {
          /*
           * NOTE: Need these line feeds inserted into the byte array
           * otherwise the check on the server fails.
           *
           * NOTE: There appears to be nothing special concerning where the
           * line feed characters are placed.  You just have to make sure
           * each string in the sequence of the base 64 encoded message
           * does not exceed 79 characters.
           */
          if ((j % 64) == 0) {
            pw4[j] = '\n';
            j++;
          }

          pw4[j] = pw3[i];
          j++;
        }
        rsaLoginMsg = new String(pw4, 0, j);

      } catch (Exception e) {
        e.printStackTrace();
      }
      finally {
        if (in != null) {
          try { in.close(); } catch (Exception ce) {}
        }
```

-
  ```
          }
  ```

-
  ```
      }
  ```
-

-
  ```
      return rsaLoginMsg;
  ```

```
}
```