



Authoring Guide

Sybase Aleri Streaming Platform 3.2

DOCUMENT ID: DC01295-01-0320-02

LAST REVISED: December, 2010

Copyright © 2010 Sybase, Inc.

All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Bloomberg is a trademark of Bloomberg Finance L.P., a Delaware limited partnership, or its subsidiaries.

DB2, IBM and Websphere are registered trademarks of International Business Machines Corporation.

Eclipse is a trademark of Eclipse Foundation, Inc.

Excel, Internet Explorer, Microsoft, ODBC, SQL Server, Visual C++, and Windows are trademarks or registered trademarks of Microsoft Corp.

Intel is a registered trademark of Intel Corporation.

Kerberos is a trademark of the Massachusetts Institute of Technology.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Mozilla and Firefox are registered trademarks of the Mozilla Foundation.

Netezza is a registered trademark of Netezza Corporation in the United States and/or other countries.

Novell and SUSE are registered trademarks of Novell, Inc. in the U.S. and other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

Reuters is a registered trademark and trademark of the Thomson Reuters group of companies around the world.

SPARC is a registered trademark of SPARC International, Inc. Products bearing SPARC trademarks are based on an architecture developed by Sun Microsystems, Inc.

Teradata is a registered trademark of Teradata Corporation and/or its affiliates in the U.S. and other

countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Group Ltd.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Table of Contents

About This Guide	vii
1. Purpose	vii
2. Organization	vii
3. Related Documents	vii
1. Introduction to Authoring	1
1.1. Dataflow Programming	1
1.2. Applying Relational Concepts to Event Streams	1
1.3. Streams - the Basic Building Blocks of a Data Model	2
1.3.1. The Anatomy of a Stream	3
1.3.2. Stream Operations - Types of Derived Streams	4
1.4. Choice of Authoring Environments	5
2. Authoring in the Aleri Studio	6
2.1. Data Files Used by the Aleri Studio	6
2.2. Opening and Viewing an Example	7
2.3. Running a Data Model	7
2.4. Creating or Opening a Data Model in the Aleri Studio	7
2.4.1. Create a New Data Model	7
2.4.2. Open An Existing Data Model in the Aleri Studio	8
2.5. The Visual Authoring Workspace	8
2.5.1. Visual Authoring: Modeling Using Diagrams	9
2.5.2. Working With Stream Shapes in the Visual Editor	10
2.5.2.1. Adding Streams to the Model	10
2.5.2.2. Collapse/Expand Stream Shapes	10
2.5.2.3. Connecting Streams to Control Data Flow	11
2.5.2.4. Edit the Properties and Behavior of a Stream	11
2.5.2.5. Automatically Lay Out a Diagram	12
2.5.2.6. Placeholder Shapes	12
2.5.2.7. The Diagram Overview	13
2.5.2.8. Visual Feedback: Red Borders	13
2.5.2.9. Delete Elements from a Data Model	14
2.5.3. The Properties View	14
2.5.4. Aleri Studio Shortcuts	15
2.5.5. The Outline View	15
2.5.6. Search, Problems, and Console Views	16
2.5.7. Adding Other Elements to the Diagram	16
2.5.8. Online Help	16
2.5.9. Customizing the Workspace	16
2.6. Connections	16
2.6.1. Add Input Connections	17
2.6.2. Output Connections	17
2.7. Data Location	17
2.7.1. Adding Data Locations	18
2.8. Data Location Explorer	19
2.8.1. Using The Data Location Explorer	19
2.9. Adding Global Variables and Functions	20
2.10. Defining Streams	20
2.10.1. Source Stream	21
2.10.2. Adding Streams From A Template Model	22
2.10.3. Aggregate Stream	23
2.10.4. Compute Stream	24
2.10.5. Extend Stream	24
2.10.6. Copy Stream	25
2.10.7. Filter Stream	26

2.10.8. Join Stream	27
2.10.9. Union Stream	28
2.10.10. Pattern Stream	29
2.10.11. FlexStream	30
2.11. Optional Stream Elements	32
2.11.1. Input Windows	32
2.11.2. Local Variables and Functions	32
2.11.3. Automatically Generated Columns	33
2.12. Checking a Model For Errors	34
2.13. Advanced Visual Authoring	36
2.13.1. Data Storage Managers	36
2.13.2. Working With Diagrams	37
2.13.2.1. Create a New Diagram	37
2.13.2.2. Populate a Diagram with All Streams	38
2.13.2.3. Add Existing Elements to a Diagram	38
2.13.2.4. Show Related Streams	39
2.13.2.5. Show Shared Components of a Stream	39
2.13.3. Create Modules	39
2.13.4. Search the Model	40
2.13.5. View the Properties of the Model	41
2.13.6. Modify Studio Preferences	41
2.14. SQL Authoring	44
2.14.1. The Studio SQL Editor	44
2.14.2. Running and Testing an SQL Model	45
2.14.3. Convert an Aleri SQL model for Visual Authoring	45
3. Expressions	48
3.1. Expression Types	48
3.2. Editing Expressions	48
3.3. More Editing Tips	49
4. Pattern Matching	50
4.1. Creating Pattern-Matching Syntax	50
4.2. Examples of Pattern Matching	51
5. Advanced Authoring Concepts	53
5.1. Retention	53
5.2. Expiry	53
5.3. Stores	54
5.4. Joins	55
5.5. Distributed Models and Clustering	58
5.5.1. Modify a Model to Run in a Cluster Configuration	58
5.5.2. Mapping Modules to Servers	61
5.5.3. Create a Cluster	62
5.5.3.1. Create a Cluster using the Outline view:	63
5.5.3.2. Create a Node for a Cluster	63
5.5.3.3. Create a Node using the Outline view:	63
5.6. Creating a Persistent Subscribe Pattern	64
6. Running and Testing a Data Model	65
6.1. The Run-Test Perspective	65
6.1.1. Configure the Run-Test Perspective	66
6.1.2. Start	67
6.1.3. RSA/PAM Settings	68
6.1.4. Kerberos/GSSAPI Authentication	70
6.1.5. Kerberos User IDs and Role-Based Authorization	70
6.1.6. Path Aliases	71
6.1.7. Data Input view	72
6.1.8. Run an Ad-hoc SQL Query	72
6.1.9. Send Commands to the Stream Processor	73
6.1.10. Aleri Studio Playback	74
6.1.11. Upload Test Data	77

6.2. Monitor	77
6.2.1. Run the Monitor	78
6.2.2. Save the contents of a performance diagram as an image	80
6.3. Streamviewer	80
6.3.1. Copy and Paste from the Streamviewer Window	81
6.4. Debugging with Breakpoints	81
6.4.1. Managing Breakpoints	83
6.4.2. Managing Watches	84
6.5. Debugger	85
6.5.1. Run the Debugger	86
6.6. Running/Testing a Sybase Aleri Streaming Platform Data Model from the Command Line	87
7. Monitoring, Tuning and Optimization Techniques	88
7.1. General Optimization Techniques	88
7.2. Log Store Optimization Techniques	88
8. Aleri Studio Execution Scenarios	89
8.1. Creating Execution Scenarios	89
8.1.1. Create an Execution Scenario	90
8.2. Adding Child Elements to an Execution Scenario	91
8.2.1. To add child elements to an Execution Scenario:	92
8.3. Working with DataFeedScenarios	92
8.3.1. Create a Data Feed Scenario	93
8.3.2. Parameters for Data Feed Scenarios	93
9. Studio Scripting	95
9.1. Create a New Script File	95
9.2. Run Scripts	95
9.3. Register a Script	95
9.4. Studio Scripting Example	96
A. Aleri Studio Keyboard Shortcuts	97
Index	100

About This Guide

1. Purpose

This document is a guide for creating and editing data models with the Aleri Studio. Most people initially prefer the tools in the Aleri Studio but you can also use XML or SQL.

A data model contains the event-processing logic applied to incoming data streams to produce one or more output streams or result sets. It is essentially the application that runs on the Sybase® Aleri Streaming Platform containing the business logic applied for delivering insight or taking action in response to incoming events.

Advanced users who are familiar with programming concepts and the Aleri Studio will find the *Authoring Reference* to be a valuable resource. If you are looking for more general information, such as an introduction to Complex Event Processing (CEP), the *Product Overview* is a good starting point.

2. Organization

This guide includes the following chapters:

- [Chapter 1, *Introduction to Authoring*](#) provides an overview of the Sybase Aleri Streaming Platform and introduces the authoring tools and languages.
- [Chapter 2, *Authoring in the Aleri Studio*](#) provides instructions for creating a Sybase Aleri Streaming Platform data model using the Aleri Studio.
- [Chapter 3, *Expressions*](#) introduces the Aleri Expression Editor, a convenient way to author complex expressions associated with streams.
- [Chapter 4, *Pattern Matching*](#) explains how to create and implement Pattern Streams to detect patterns of events across one or more streams in the Sybase Aleri Streaming Platform.
- [Chapter 5, *Advanced Authoring Concepts*](#) explores more advanced Sybase Aleri Streaming Platform elements and configurations.
- [Chapter 6, *Running and Testing a Data Model*](#) shows how to test, debug and examine a data model.
- [Chapter 7, *Monitoring, Tuning and Optimization Techniques*](#) provides tips to optimize data models for maximum performance.
- [Chapter 8, *Aleri Studio Execution Scenarios*](#) shows how this perspective can define, save and edit multiple execution scenarios.
- [Chapter 9, *Studio Scripting*](#) explains how the Aleri Studio runs Javascript™-based scripts within its environment.

The following appendix has been added at the back of this guide:

- [Appendix A, *Aleri Studio Keyboard Shortcuts*](#)

3. Related Documents

This guide is part of a set. The following list briefly describes each document in the set.

<i>Product Overview</i>	Introduces the Aleri Streaming Platform and related Aleri products.
<i>Getting Started - the Aleri Studio</i>	Provides the necessary information to start using the Aleri Studio for defining data models.
<i>Release Bulletin</i>	Describes the features, known issues and limitations of the latest Aleri Streaming Platform release.
<i>Installation Guide</i>	Provides instructions for installing and configuring the Streaming Processor and Aleri Studio, which collectively are called the Aleri Streaming Platform.
<i>Authoring Guide</i>	Provides detailed information about creating a data model in the Aleri Studio. Since this is a comprehensive guide, you should read the <i>Introduction to Data Modeling and the Aleri Studio</i> . first.
<i>Authoring Reference</i>	Provides detailed information about creating a data model for the Aleri Streaming Platform.
<i>Guide to Programming Interfaces</i>	<p>Provides instructions and reference information for developers who want to use Aleri programming interfaces to create their own applications to work with the Aleri Streaming Platform.</p> <p>These interfaces include:</p> <ul style="list-style-type: none">• the Publish/Subscribe (Pub/Sub) Application Programming Interface (API) for Java• the Pub/Sub API for C++• the Pub/Sub API for .NET• a proprietary Command & Control interface• an on-demand SQL query interface
<i>Utilities Guide</i>	Collects usage information (similar to UNIX® man pages) for all Aleri Streaming Platform command line tools.
<i>Administrators Guide</i>	Provides instructions for specific administrative tasks related to the Aleri Streaming Platform.
<i>Introduction to Data Modeling and the Aleri Studio</i>	Walks you through the process of building and testing an Aleri data model using the Aleri Studio.
<i>SPLASH Tutorial</i>	Introduces the SPLASH programming language and illustrates its capabilities through a series of examples.
<i>Frequently Asked Questions</i>	Answers some frequently asked questions about the Aleri Streaming Platform.

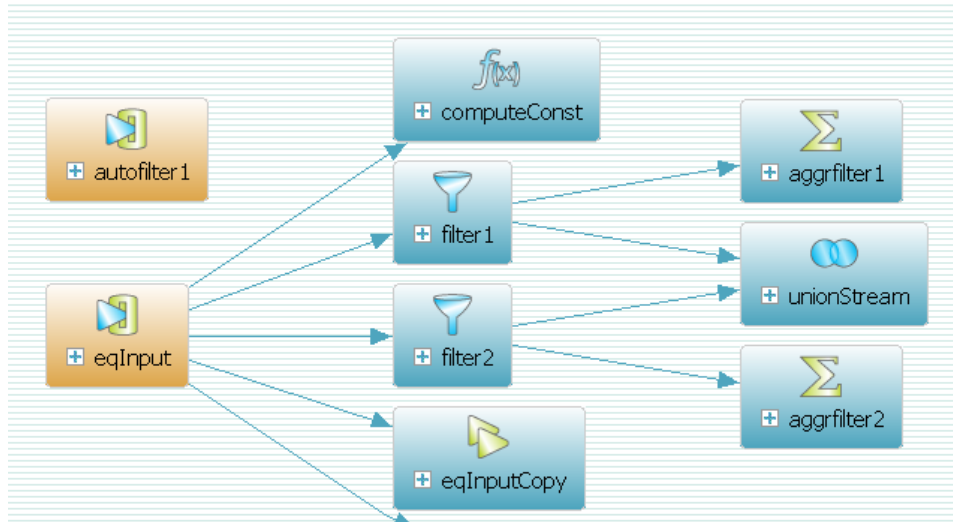
Chapter 1. Introduction to Authoring

The Sybase Aleri Streaming Platform is programmable CEP software that processes events according to a pre-defined *data model*. A data model contains information about the stream(s) of input events and the applied business logic for producing the desired output data.

The process of creating a data model is referred to as "authoring" since computer programming skills are not required. Users without programming skills can learn how to build simple models with relative ease. However, Sybase provides a very feature-rich environment with a range of power tools that mostly require basic programming skills.

1.1. Dataflow Programming

Sybase uses the concept of *Dataflow Programming*. The technique breaks a program into a series of operations with an arrow between each one. It can be conceptualized by a directed graph or flow diagram, which is how a data model is visually represented in the Aleri Studio.



Dataflow programming breaks a potentially complex computation into a sequence of operations with data flowing from one operation, or node, to the next. Besides simplifying the implementation of complex logic, this technique also provides scalability and possibilities for parallelization, since each operation is event driven and independently applied. That is, each node can be run on a separate processor, and needs to process an event only when it receives one from another node. No other coordination is needed between the nodes.

1.2. Applying Relational Concepts to Event Streams

The Sybase Aleri Streaming Platform borrows concepts from the relational database world for the task of processing event streams. Each stream of events can be considered as a series of updates to a table. Incoming events can add new rows to a table or modify (update or delete) existing rows. When defining a data model, you can apply relational operators to streams to produce new derived streams, which are analogous to materialized views in the database world. When authoring a data model, using the Aleri Studio or AleriML modeling language, "tables" are referred to as "source streams" and "materialized views" as "derived streams." In the Aleri SQL language, the use of "table" and "materialized view" is kept to be consistent with standard SQL.

The Sybase Aleri Streaming Platform takes the database concept of a static data set that you query when you need information, and turns it upside down. Here you define the queries first and as the data arrives, it passes through the queries. This architecture is entirely event-driven and designed to reduce the time for an event to flow through the model and update all the affected result sets. In other words, it minim-

izes the latency between an event and the response, with the goal of providing fast, actionable intelligence.

To keep streams from becoming unmanageable, input windows can be created to define how large a stream's data set can grow or how long items will be kept. While the concept of input window management is critical for CEP systems, it does not exist in traditional databases systems. CEP systems can provide very fast and efficient incremental materialized view maintenance with the added benefits of input windows over fast-streaming data.

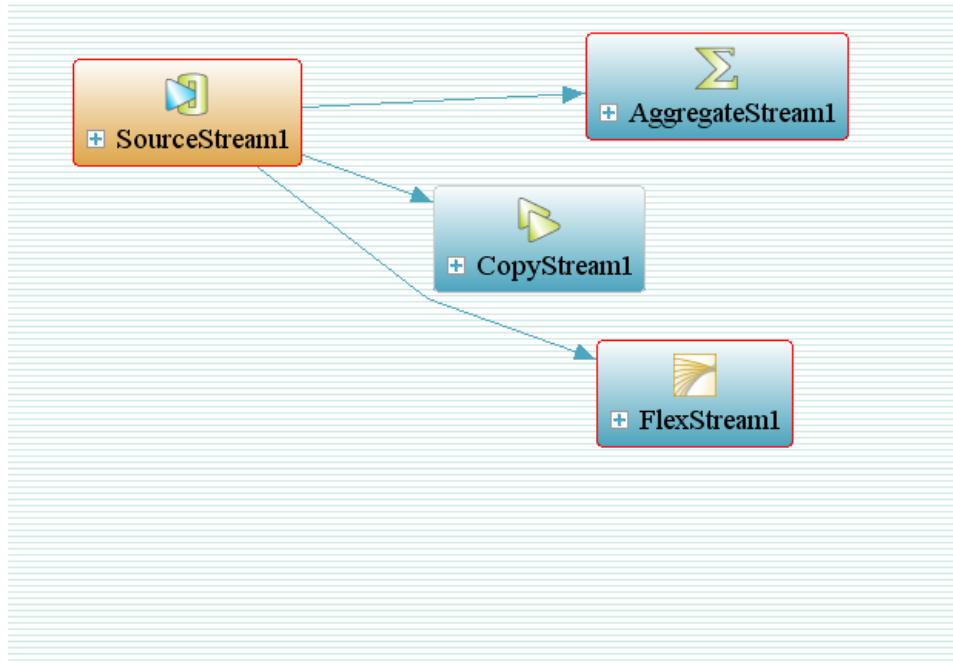
1.3. Streams - the Basic Building Blocks of a Data Model

At its most basic level, a data model is made up of a set of streams. There are two fundamental types of streams: source streams and derived streams. Source streams are the point at which data enters the model from external sources. Derived streams are the results of applying an operation to one or more input streams to produce a new event stream derived from the input stream(s). So the simplest form of a data model would consist of a single source stream receiving a stream of incoming events and a single derived stream that applies an operation to the source stream to produce a new event stream according to some business logic implemented within the operation. But models can get quite complex, with numerous source Streams and derived Streams. A model with 50 or more streams is typical.

Source streams are the entry point for all event streams into the model and sets of reference or configuration data used within the model. While a source stream may receive incoming event data on an event-driven basis, it can also receive static or semi-static sets of data that are loaded once or periodically refreshed. A single model may have any number of source streams.

Derived streams don't receive input from an outside source but act on other streams in the model. A derived stream will have one or more input streams, which may be source streams, other derived streams, or any combination thereof. Derived streams apply one or more operations to the data arriving on the input streams to produce an output stream. An *input* stream for a derived stream is the output of another stream (either source or derived).

A key aspect of this architecture is that streams are typically stateful. This can be a confusing concept, since a stream of messages delivers each message one at a time. But most streams in a data model simply don't operate on a single message and output the result but rather retain a set of records. Depending on the data model, all records can be retained or an input window can be defined. The window can specify the number or time period of records to retain. A stream could get infinitely large, but a unique aspect of the architecture is its ability to process incoming messages as updates to a stream. There are some data models where a stream has a finite set of elements and incoming events update the elements rather than just getting appended as new elements.



1.3.1. The Anatomy of a Stream

As described above, a data model consists of data flowing between *streams*. A "stream" in the Sybase Aleri Streaming Platform has several characteristics:

Input stream(s)	Every stream in the data model will have one or more input streams. In the case of a source stream, this is a stream of events from external systems. In the case of Derived Streams, this is the output of another stream(s).
Key	Every stream in a data model is indexed according to a key column or set of key columns. Every row in the stream must have a unique value in the key. For streams of data entering the Sybase Aleri Streaming Platform that do not have a natural key, the source stream can apply a key that serves to uniquely identify the row.
Store (data store)	Every stream is assigned to a data store that holds the events produced (output) by that stream. By default, streams are assigned to an in-memory store, but advanced users can assign streams to a log store enabling disk-based data persistence for full state recovery or a "stateless" store to optimize performance.
Retention rules	Streams can have retention rules set to determine how much event data is retained in the store. The rules define the number of events to be retained or a time window.
Operations	Every derived stream includes one or more operations to be applied to incoming events to produce the output of the stream. The type of operations applied in the stream depends on the type of derived stream.
Output	Every stream produces an output stream. The output stream is the stream of events resulting from the operation(s) applied by the stream. The output can serve as the input to another stream and/or it can be published or subscribed by external systems or applications.

Datatypes	Here are the seven datatypes for columns: <ul style="list-style-type: none">• date• double• int32• int64• money• string• timestamp
-----------	--

1.3.2. Stream Operations - Types of Derived Streams

Each derived stream applies one or more operations to incoming events to produce an output stream. There are nine types of derived streams:

Aggregate	Groups records according to a common value in one or more columns and produces a single summary record for each group. Takes a single input stream.
Filter	Filters records according to a defined filter expression. Only records on the input stream that cause the filter expression to be 'true' are included in the output. Takes a single input stream.
Compute	Computes output records from input records. The fields in the output record are computed according to an expression defined for each column. Takes a single input stream.
Extend	Is just like a Compute Stream, but does not require an expression for every column in the output stream. Columns in the input stream are passed through unmodified and any new columns are added according to the column expressions defined for the Extend Stream. Takes a single input stream.
Pattern	Watches for patterns of events and produces one or more output records whenever a defined pattern is detected. Takes any number of input streams. Patterns can be defined for a single stream or across multiple input streams.
Join	Joins two or more streams to produce a single output stream. Just like a database join, output records are produced by matching records from the input streams according to common values in selected columns.
Union	Merges two or more streams into a single stream. Input streams must have the same format (column definitions).
Copy	Produces a copy of the input stream. Primarily used for clustered models to provide a local copy of an input stream. Can also be used to effect a self-join.
Flex	Is a programmable stream. Takes any number of input streams. Events on an input stream invoke a method written in the SPLASH scripting language.

Then click on the Connector tool in the **Palette**, click on the stream that will provide input to the new stream, and then click on the new stream. This will add an arrow to the diagram, showing the flow of data.

1.4. Choice of Authoring Environments

The Sybase Aleri Streaming Platform supports three different authoring environments: visual authoring in the Aleri Studio, Aleri SQL and AleriML. The variety helps minimize the learning curve and maximize productivity for application developers.

Aleri Studio and Aleri SQL are the two high-level options for visual authoring.

The Aleri Studio is an Interactive Development Environment (IDE) based on the Eclipse™ framework, which offers a visual paradigm for creating and editing data models. Visual Authoring is ideal for new users since it doesn't require programming skills or knowledge of language syntax.

Aleri SQL uses the "Structured Query Language" (SQL) supported by virtually all relational databases. Because the architecture of the Sybase Aleri Streaming Platform leverages relational concepts, SQL is very well-suited for expressing continuous queries that result in derived streams. Users comfortable with SQL will find it very easy to define data models using Aleri SQL since it is consistent with standard SQL but adds some extensions to address a streaming data environment. Modeling with Aleri SQL can be done in the Aleri Studio or any text editor.

Both the Aleri Studio and Aleri SQL generate a data model in the form of AleriML - an XML document that conforms with the Aleri XSD. It's this XML file that is read by the Sybase Aleri Streaming Platform to load the data model. Users can build data models directly in AleriML using the XML or text editor of their choice. Alternatively, a data model can be created in the Aleri Studio or Aleri SQL and subsequently edited in AleriML.

Users have the flexibility to switch between creating or editing a model in AleriML and viewing and editing in the visual programming environment of the Aleri Studio. Changes to the model made in either environment are reflected in the other. For example, an initial model can be diagrammed in the Aleri Studio and then details can be edited directly in the underlying XML file. AleriML is also useful when generating or modifying a model from within another application, since off-the-shelf XML parsing/formatting tools can be used.

Currently, it's not possible to create an Aleri SQL model from an AleriML model. Any changes made using the visual editor of the Aleri Studio or to the AleriML model will not be reflected in the Aleri SQL model.

Chapter 2. Authoring in the Aleri Studio

This chapter describes how to use the Aleri Studio to create, edit and visualize a data model.

The Aleri Studio is an Interactive Development Environment (IDE) based on the Eclipse™ framework. It lets you create, edit, diagram, test, debug and optimize a data model.

The Aleri Studio gives two choices for creating a data model:

1. **Visual Programming:** the model created and edited using diagrams. Stream shapes are dropped onto a canvas and connected with arrows indicating the flow of data. Stream properties and child elements are defined to control behavior.
2. **Aleri SQL:** the model is created using a textual language - Aleri SQL - in the syntax-aware SQL editor within the Aleri Studio.

The Aleri Studio can be used to build a data model from scratch or to visualize, edit and test a model that was previously created in Aleri SQL or AleriML. If a SQL Model is edited using the diagram editor, the changes will not be reflected in the original SQL model.

Models are created, edited, and diagrammed in the authoring perspective of the Aleri Studio. This is one of three perspectives in the Studio. The other two - the Run-Test perspective and the Execution Scenario perspective - are used to test, debug and examine a model. More details on those perspectives are available in [Chapter 6, *Running and Testing a Data Model*](#) and [Chapter 8, *Aleri Studio Execution Scenarios*](#).

2.1. Data Files Used by the Aleri Studio

The Aleri Studio creates and opens four types of data files, which are described below.

<file>.xml	Every data model will ultimately be contained in a .xml file that contains the model definition in AleriML. The Aleri Studio automatically creates the .xml file for models. It can also read and load models contained in a .xml file (that conforms with the AleriML DTD) not created within the Aleri Studio. When the Streaming Processor is run, it reads the model from the xml file.
<file>.sql	Models that are written in Aleri SQL are contained in a text file named with a .sql extension. When the Aleri Studio is used to define a data model, it creates the file automatically. The Aleri Studio can also read and load an Aleri SQL model that was created using any text editor. Before a model defined in Aleri SQL can be run, the model must be translated to AleriML and saved in an xml file. This is done automatically if the SQL model is opened in the Aleri Studio and then run. There is also an option on the File menu to convert an Aleri SQL model to an AleriML model.
<file>.notation	Models created or opened in the Aleri Studio's Visual Authoring environment will have a .notation file that contains the diagram(s) and related information. The data model itself will be contained in the associated .xml file of the same name. When creating a new model, the Aleri Studio defaults to Visual Authoring, automatically creating both a .notation file and the associated .xml file.
<file>.js	This is a Java script file, specific to the Aleri Studio and completely independent of the data model(s), that can automate tasks. See Chapter 9, <i>Studio Scripting</i> for more details.

2.2. Opening and Viewing an Example

The Aleri Studio includes several example data models to help new users get started. Each example includes a notation file, the underlying xml file, and test data. These examples can be loaded into the Aleri Studio environment with the **File/Open** menu.

When you click on **Open Example** on the Welcome page, the **File/Open** dialog box opens pointed to the directory containing the examples. On a Windows® PC, this is typically the My Documents\Aleri folder. On Linux® and Solaris™ systems, this is the /home/alери/alери/platform/3.2.0/examples directory unless you specified an alternate location when you installed the software.

2.3. Running a Data Model

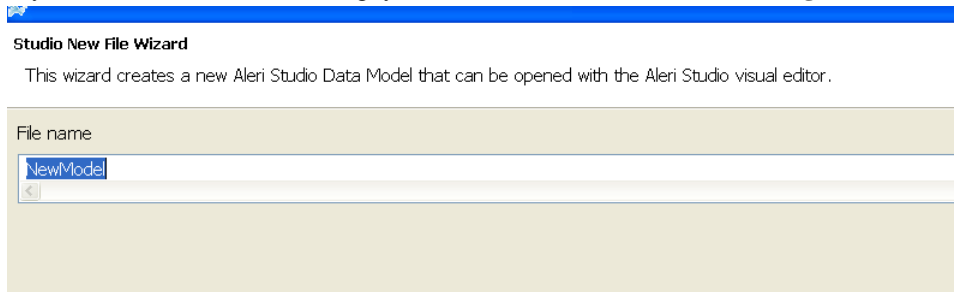
Use the **Run** button in the toolbar under **Scripting** to run an example or new data model. Start-up options can be configured from the drop-down menu available to the right of the **Run** button. The **Run** button can also be used to stop the model so make sure it is green when you want to start it. If it is red, then just click on it to turn back to green.

2.4. Creating or Opening a Data Model in the Aleri Studio

2.4.1. Create a New Data Model

The following steps explain how to create a new data model.

1. Select **New File** from the **File** menu.
2. You will have the option of selecting one of three choices:
 - **Data Model using Visual Editor**
 - **SQL Data Model**
 - **Studio script**
3. If you want to do visual authoring, you should select the **Data Model using Visual Editor**.



4. The New File Studio Wizard then appears where you should type a name for data model in the **File Name** field.

If you are creating a model using Visual Authoring, this will produce both a .notation file and .xml file. If you are creating a model using SQL, this will only produce the .sql file; an associated .xml file will be created later.

5. Click **Finish**.

The Aleri Studio performs the following:

- Creates the new data model files.

- Opens the Authoring Perspective, which displays a blank diagram.

2.4.2. Open An Existing Data Model in the Aleri Studio

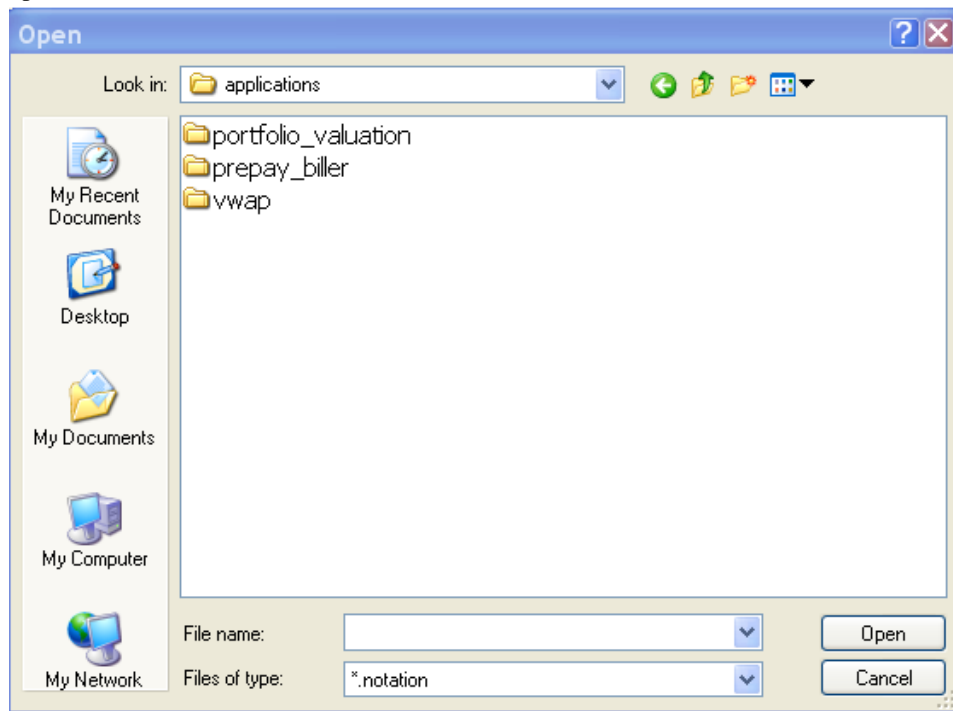
You can use the Aleri Studio to visualize and/or edit a data model, no matter which authoring environment was used to create it.

1. Select **Open File** from the **File** menu.

By default, this will list models that were created using the Visual Editor in the Aleri Studio (or previously edited and saved with the Visual Editor). These are models for which a .notation file exists. Navigate to the directory containing the model you wish to open and select the model.

Alternatively, you can open an SQL model in the SQL editor. To do this, change **Files of type** to *.sql to see which SQL models are available to open.

You can also open an AleriML model in the Visual Editor. To do this, change the **Files of type** to *.xml to see which AleriML models are available. When you open an AleriML model, it will be opened in the visual editor and a .notation file will be created for the model.



2. Select the file you wish to open in the navigator or type the name in the **File Name** field.
3. Click **Open**.

2.5. The Visual Authoring Workspace

The Visual Authoring perspective of the Aleri Studio provides the tools and views needed to create, view and edit a data model. It consists of the following views:

- The **Editor** is the canvas at the center of the workspace where the model is built and displayed.
- The **Palette** on the left shows the stream types that can be added to the model. To add a stream to the model, select the appropriate type on the **Palette** and then click in the editor to drop the stream onto the diagram and into the model. The **Palette** also contains a note object that can be used to add comments to the diagram. At the bottom of the **Palette** are collections of advanced tools that can be added to the diagram.
- The **Properties** view in the lower left corner displays the properties of the object currently selected in the diagram. Properties can be set or modified in this view.
- The **Data Location Explorer** tab (located next to the **Palette** and **Outline** tabs) allows you to define external data locations that will either produce data that gets processed by the model or receive output from the model.
- The **Outline** tab (next to **Palette**) is an indexed list of all elements defined in the current data model.
- The **Search Model** view allows the user to search a model for strings contained within elements; the search results are displayed in the view. The found elements can then be modified individually using the Properties window, or multiple items can be selected and modified using the replace action.
- The **Problem(s)** view displays any errors found when checking the model for violations.
- The **Studio Shortcuts** view contains a handy list of the Aleri Studio's keyboard shortcuts.
- The **Console** view is used to display messages generated when running Studio scripts.

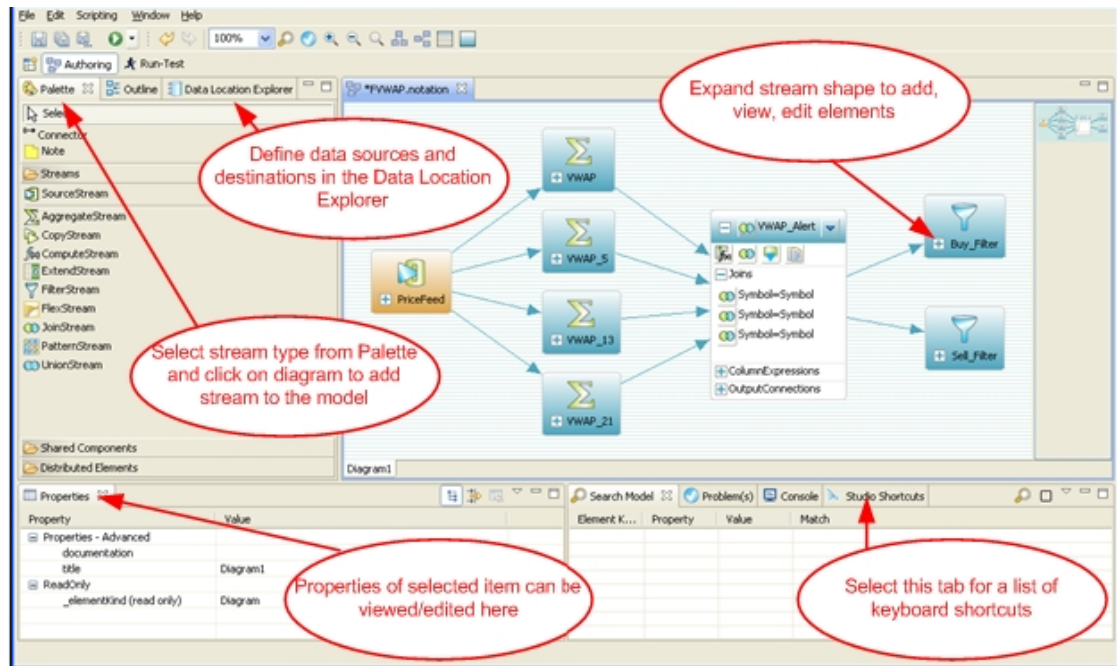
The small navigator in the upper right corner of the Editor is called the **Overview**. It provides a convenient way of navigating around a large diagram. This can be turned off by right clicking it to hide it. Once turned off, it can be restored by right clicking on the canvas in the editor and selecting Show...Overview.

The toolbar contains buttons to:

- **Zoom** in/out on the diagram.
- Check the Model for Violations.
- Toggle all stream shapes between icons and verbose views.
- Lay out the diagram.

Just below the toolbar are the tabs to select the perspective. The Visual Authoring and Run/Test perspectives are open by default. Click on the **Perspectives** button at the left end of this toolbar to open other perspectives or select Open Perspective in the Window menu.

2.5.1. Visual Authoring: Modeling Using Diagrams



Visual Authoring revolves around diagrams. A diagram is opened in the Editor, which is the main central view in the Visual Authoring perspective. The model is displayed in the diagram as a collection of stream shapes that are interconnected with arrows showing the flow of data. A model is created by selecting new stream types from the **Palette**, dropping them onto the diagram, connecting them to other streams, and configuring their behavior.

You interact with the diagram by adding shapes from the **Palette**, moving shapes around, and deleting them. Each shape in the diagram corresponds with an element in the underlying data model. You modify the data model by interacting with the shapes in the diagram. When you click on a shape in the diagram, the border changes to indicate that the shape is selected. You can then define or change the properties and behavior of that element.

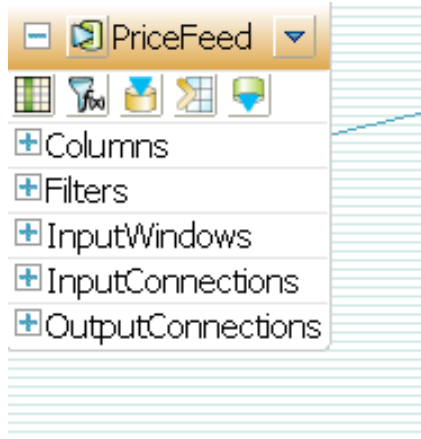
2.5.2. Working With Stream Shapes in the Visual Editor


2.5.2.1. Adding Streams to the Model

When you are creating a new model, the diagram starts off as a blank canvas. At this point, there are no elements in the model. A data model is created by defining streams, the properties and behaviors of each stream, and the flow of data between streams. To create a model, you start by adding streams to the diagram. To do this, select the type of stream you wish to add from the **Palette**, then click on the diagram to add that type of stream to the model.


2.5.2.2. Collapse/Expand Stream Shapes

Streams in a diagram can be displayed in one of two ways: in “Verbose” or “Iconic” mode. By default, new stream shapes are added to the diagram in Verbose mode. This shows the property and elements of the stream in the diagram.



As diagrams expand, you might want to collapse a stream shape into a simple icon, which hides the detailed properties of the stream in order to take up less space. A verbose stream shape can be collapsed to an icon by clicking on the collapse button. This should get the  in the upper left corner of the shape.



A stream icon can be expanded by clicking on the expand button  in the icon.

All streams can be toggled between icons and verbose shapes via the **All Verbose** and **All Iconic** buttons in the toolbar.

2.5.2.3. Connecting Streams to Control Data Flow

Sybase uses a "dataflow programming" model for event processing. A complex application is decomposed into a set of streams with data flowing between them. In Visual Authoring, the dataflow is shown by arrows (connectors) in the diagram. The connectors direct the output of one stream to one or more streams where it will become their input.

Connectors are added to the diagram as follows:

1. Select the **Connector** tool from the **Palette**.
2. Click on a stream in the diagram that will produce the output you are now directing.
3. Now click on the stream that will receive the data. The connector will appear on the diagram, with the arrow indicating the direction of flow.

By creating adding a connector to the diagram, you are also setting the input stream property of the destination stream in the underlying model.

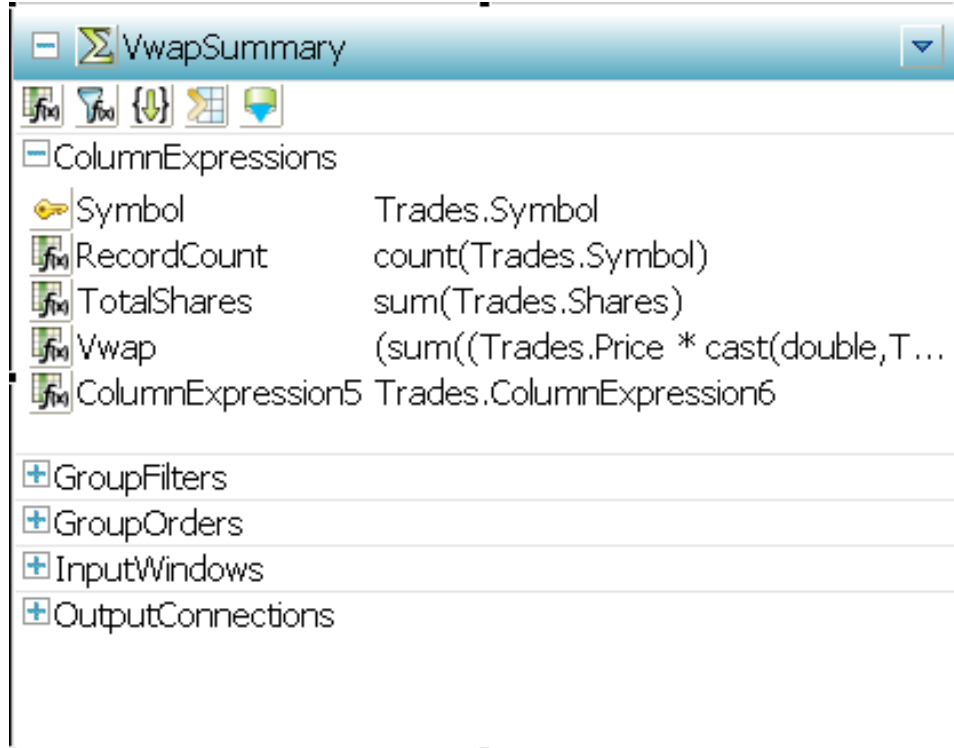
To add multiple connectors to the diagram without having to re-select the Connector tool on the **Palette** each time, hold the shift key while selecting the **Connector** tool in the **Palette**. You can then make as many connections as you like. Then press the **Esc** key or click on the **Select** tool in the **Palette** to return to normal selection mode.

2.5.2.4. Edit the Properties and Behavior of a Stream

The properties and behavior of a stream can be set or altered directly within the stream shape on the diagram. If the stream is collapsed into an icon, you would start by expanding to the verbose shape by clicking on the - in the left corner so that + appears .


You can then add elements to the stream using the toolbar or the drop-down menu in the header of the shape. The stream name can be changed by clicking on the icon to the left of the stream name.

Properties and child elements of the stream can be viewed and edited within the individual compartments in the shape. Each compartment can be expanded or collapsed using the controls in the shape as needed. If you want to move a new column after adding it, you can right-click on the name of the new column to get a drop-down menu that offers the options of moving a child element up or down.



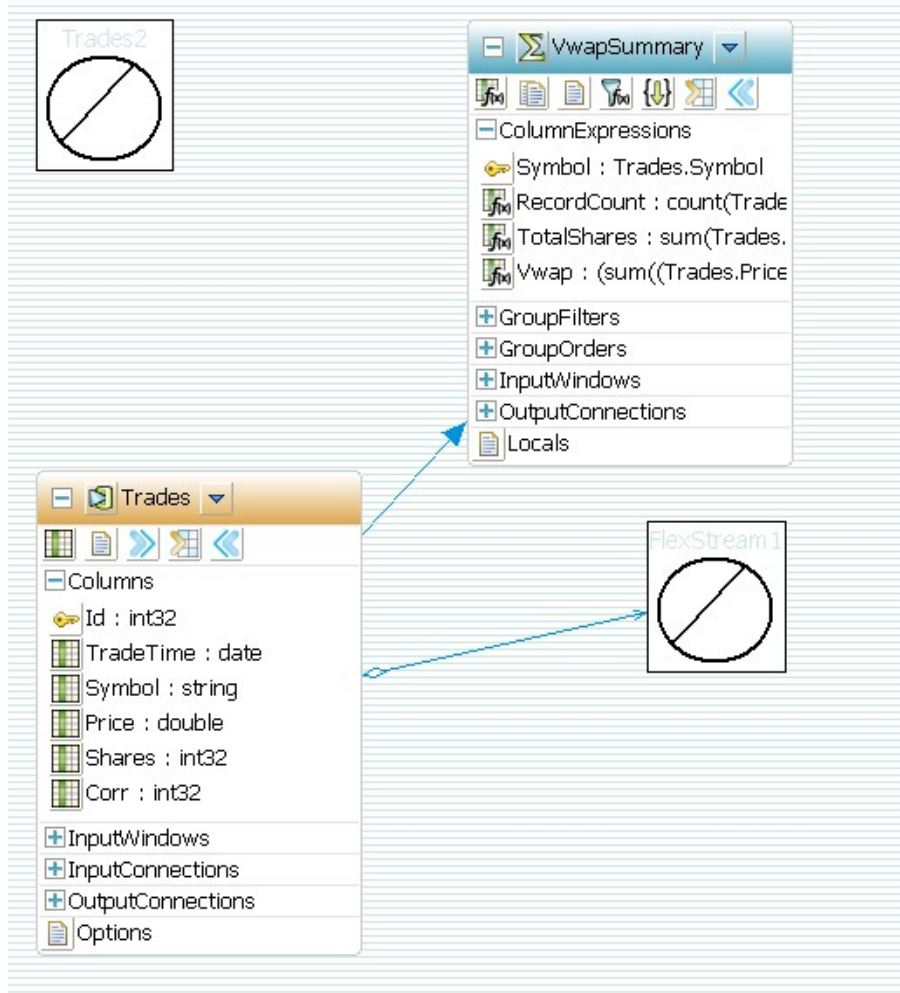
Properties and child elements of a stream can also be viewed and edited in the Properties View as described in the following sections.

2.5.2.5. Automatically Lay Out a Diagram

The Aleri Studio provides a convenient tool to automatically arrange shapes on a diagram. You can choose between a horizontal (left to right data flow) or vertical (top to bottom data flow) layout. To apply an automatic layout, select one of the two layout buttons in the toolbar, horizontal  or vertical



2.5.2.6. Placeholder Shapes



Your model might contain placeholder shapes like those pictured above. These shapes appear when a referenced element has been deleted outside of the Aleri Studio or your model contains elements that were featured in earlier versions of the Sybase Aleri Streaming Platform, but no longer exist in Version 3.0 and later, such as RowDefinitions or rules.

Placeholder shapes let you preserve the diagram's appearance. The elements are only visual references and do not exist in the model. It's easy to remove the shapes when no longer needed.

2.5.2.7. The Diagram Overview

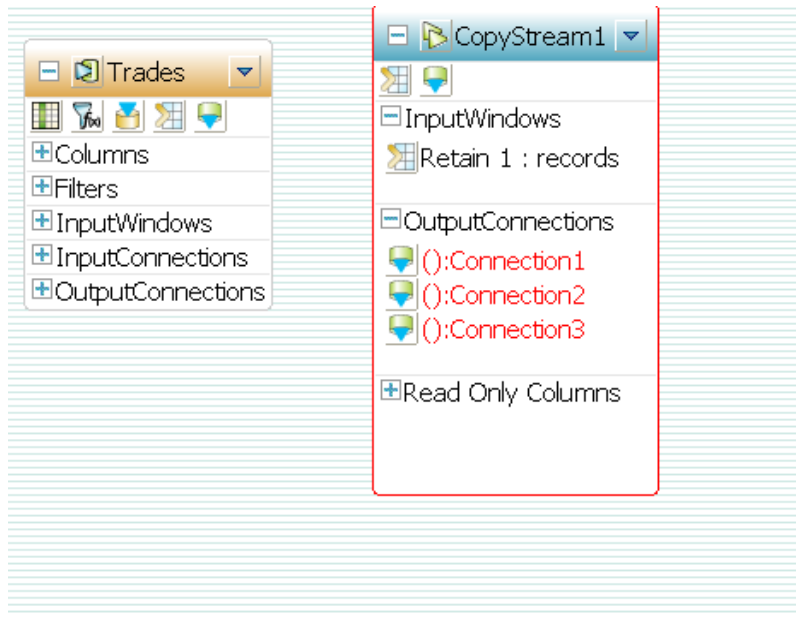
An overview of the diagram is shown in the upper right corner of the Editor is displayed by default. This is a miniature view of the full diagram and a convenient navigation tool for large diagrams. It shows which piece of the diagram is currently displayed in the editor. You can move the current view by clicking on the block and dragging it around the overview.

You can hide the overview by right-clicking on it and selecting **Hide Overview**. If the Overview is hidden and you wish to restore it, right click anywhere in the background of the diagram and select **Show > Show Overview**.

2.5.2.8. Visual Feedback: Red Borders

A stream shape on the diagram is outlined in red if its definition is incomplete or incorrect. When a stream has all required properties and elements defined, the border will turn gray. To see what items need to be set or defined, select one or more items and run **Check Model Violations** or run it with no

items selected to check the entire model and see the results in the **Problem(s)** view.



There is also a context (right-click) menu command for shapes named **Refresh Shape Visuals**. Occasionally a stream may be shown with a red border even though you believe it has been fully and correctly defined. If this happens, select **Refresh** from the right click menu to see if the border turns gray. If so, the stream definition is OK. But if it remains red, there is a problem with the stream configuration.

2.5.2.9. Delete Elements from a Data Model

There are two types of “delete”:

- **Deep Delete** removes the shape from the diagram and completely deletes the associated element from the data model. Undo last action (**Ctrl-Z**) can be used to bring it back in the current session, but once the model is saved, the shape cannot be brought back to the diagram or the model.
- **Delete from Diagram** removes a shape from the current diagram but leaves the associated element in the data model. The element still exists in the model and can be accessed from the Outline Area (elements cannot be deleted from the Outline without completely deleting them from the model - all elements in the model will appear in the Outline). The element can be added back to the diagram by dragging it from the Outline Area and dropping it onto the diagram.

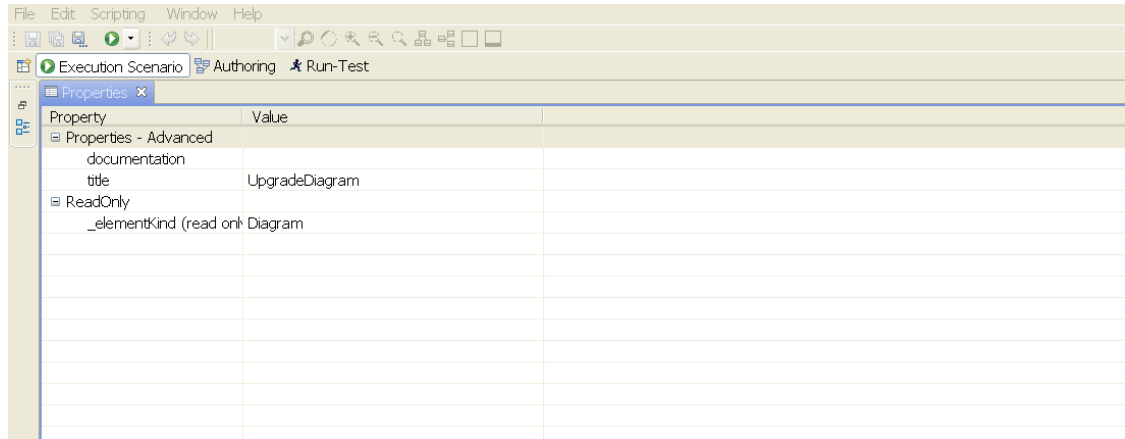
DEL on the keyboard performs a **deep delete**, completely deleting the element.

Ctrl-DEL will **delete from diagram** but leave the element in the model.

Both choices are available from the context menu. Bring up the context menu by right-clicking on an item in the diagram.

2.5.3. The Properties View

The Properties view lists all the properties for the element selected in either the Diagram or the Outline view. Properties can be modified (unless noted as read-only) by clicking on the cell and typing a value or selecting from the list (depending on the property).



Note

If there are too many rows or columns to fit in the view, the program will add scrollbars. If you don't see what you're looking for, try scrolling.

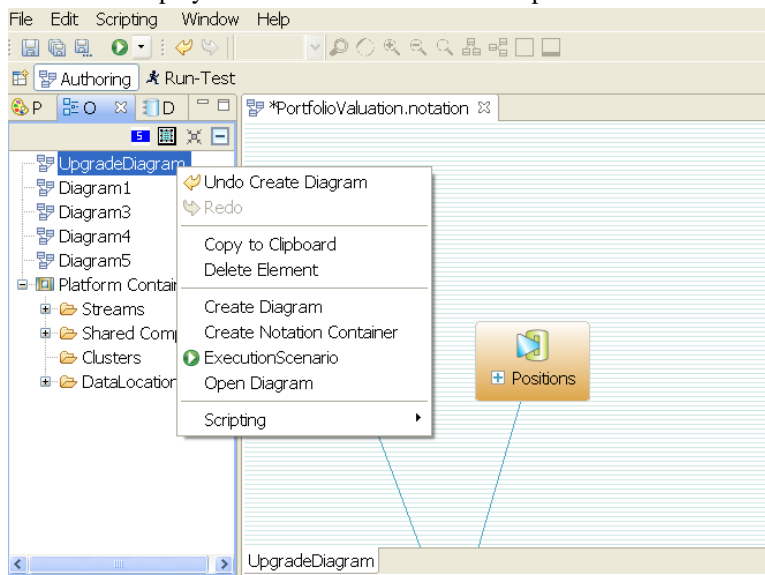
2.5.4. Aleri Studio Shortcuts

The Aleri Studio Shortcuts view in the lower right corner of the perspective provides a handy list of keyboard shortcuts.

2.5.5. The Outline View

The Outline view shows all elements in the current model in a tree-style browser. It can be used to quickly navigate a model or access elements that are not displayed on the diagram.

Users can access the Outline view by clicking on Window in the toolbar and then selecting Show View, which will display the menu with Outline as an option.



You can even add elements to the model directly from the Outline view without adding them to the diagram. Do this from the context menu (right click) and then edit the elements properties and child elements from the Properties views.

Two gestures in the Outline view make it easier to find streams in large models:

- If you right-click **Stream Object**, it will run the **context** menu choice for Show On Current Diagram.
- You can also double-click **Stream Object**.

2.5.6. Search, Problems, and Console Views

See [Section 2.13.4, “Search the Model”](#) and [Section 2.12, “Checking a Model For Errors”](#) for more information.

2.5.7. Adding Other Elements to the Diagram

- **Notes:** The **Palette** contains a Note tool that can be used to add comments to the diagram. Notes don't affect the model.
- **Shared Components, Distributed Elements:** at the bottom of the **Palette** are two additional groups of elements that can be added to the diagram if desired. These elements are described in

2.5.8. Online Help

The Aleri Studio provides integrated online help that can be used while authoring. The following choices are available from the **Help** menu:

- **Welcome** displays the Aleri Studio's introduction screen
- **Help Contents** displays online help in a separate window
- **Dynamic Help** displays the online help inside Studio. Context sensitive help is displayed by **F1** on Windows and **Ctrl+F1** on UNIX®.
- **Search** displays the online help in the search window inside the Aleri Studio
- **Show Studio Shortcuts View** displays the keyboard shortcuts for Studio in a separate window
- **Context Sensitive Help** (**F1** on Windows, **Ctrl+F1** on UNIX) displays the context sensitive topic for the element (shape, outline view tree node or a Studio window) selected in the UI.

2.5.9. Customizing the Workspace

As the Aleri Studio is implemented on the Eclipse™ framework, many users will be familiar with how to customize the workspace. Individual views can be moved by dragging them to another position, collapsed by clicking on the minimize tool in the view header or closed by clicking on the "x" in the view tab. Views can be added to the workspace from the **Window > Show View** menu. Right clicking on a view tab or header also provides a list of actions that can be performed.

2.6. Connections

A connection to an external data source is either an input or output connection. An input connection allows a source stream to absorb data, while an output connection enables the publishing of stream data. Connections are one of the two ways that streams can receive or publish data. External components, such as adapters, can serve the same role for streams.

You can use the Selection Dialog when editing connections that lists a tree of DataLocations. You must choose a predefined Data Location for a connection and fill in any required parameters. If you select a DataLocations that resides in a shared model, then it is copied to the current model. Refer to the [Sec-](#)

tion 2.7, “Data Location” for more information on on editing parameters.


After you select a data location for your connection, you will see a property sheet with default properties filled in. These default values are taken from the selected Data Location. You can change these properties, but the changes will only impact the connection you are defining at the time and not the underlying Data Location.

Among the connection features in the Sybase Aleri Streaming Platform:


- Show Sample Data lets you ask an input connection to get and display a few sample rows of data from its source.
- Show Schema lets you ask an input connection to show the schema for a table or pseudo table, which is a collection of records with the same structure.
- Field Mapping is an optional feature that lets you define a mapping between the field names of a Stream and a configured Connection. Projection and permutations are supported. If you don't choose this option, then the model will use the existing order.

2.6.1. Add Input Connections

If you want to add an input connection:

1. The source stream should be in verbose mode. If the stream isn't in verbose mode, then click on the + sign in the left corner of the stream.
2. Click the Add In Connection  icon, which is below the name of the source stream.
3. Right-click the mouse and select the **edit** menu choice to edit the connection.

2.6.2. Output Connections

Output Connections are Child elements of any stream. An Output Connection is created using the **Add Output Connection** toolbar button on the verbose shape  or the **Create New Child** context menu on the shape.

The Output Connection name and parameters can be edited in three ways:

- Using the F2 key
- Double-clicking the corresponding item in the verbose shape
- From the Edit Parameters context menu for the Connection

2.7. Data Location

Data Locations make it easy to integrate the Sybase Aleri Streaming Platform with existing data sources as well as with data consumers. The feature works by getting necessary information from the data source or data consumer to work with the Sybase Aleri Streaming Platform, saving time as you create your model. Data Locations contain a "Connection Type", with optional and required parameters. A Connection Type is a generic data connection (input or output connector), used directly in a stream to define a connection, such as with a database, message bus, and files and sockets. It can also be applied to build Data Locations to use in the Data Locations Explorer. Connection Types usually specify generic details

regarding the connection, such as:


Database Type	Sybase ASE, kdb+, DB2®, Oracle®, Teradata®, Netezza®, SQL Server®, PostgreSQL
Database Host Name	the machine running the database
Server	server host name
Directory	location of data files

The Data Location can then be used to discover datatypes or perform other Data Location operations such as assigning the Connection to a source or derived stream. Multiple connector can share the same Data Location.


Data Location Discovery lets you browse into a Data Location to discover tables (or pseudo tables) and schema. It's typically used to browse a Data Location and drag and drop a table onto the authoring diagram to create a source stream with predefined schema and connection information.

2.7.1. Adding Data Locations

This section applies to Visual Authoring only; see *The Authoring Reference* for information on data sources and connections in SQL models.

If you want to create or edit a Data Location, you must first click on the Data Locations Explorer tab  in the upper left hand corner next to the Palette and Outline tabs.

Here are the steps to add a Data Location in the Aleri Studio:

1. Double-click on the **Create Data Location** icon to open the edit parameters dialog.  below the tab. This will bring up a dialog box.
2. Select the type of Connector that you want from the available choices.
3. After you select a type, the default parameters are displayed and default values are automatically filled in. You can tailor the parameters to your specific needs. Required parameters are shown in red.
4. Click OK to finish.

You can also use the Data Location Template to create a Data Location.

1. Select the + next to the **Data Location Template** option
2. You will then see a listing of Connector types for you to select. Click on the + for a list that contains each individual connector.
3. Right-click on your connector choice and select **Create Data Location from Template**. You should check and (if necessary) edit the listed parameters.
4. Click OK when you are done.

An example of a Data Location is a Connection Type named Database Input that ships with the Sybase Aleri Streaming Platform. It has a predefined Data Location in the Data Location Templates folder, named sqlsrv_input that uses Connection Type. Many parameters for this Data Location are already filled out, including Database Type (MS SQLserver), Hostname, Port, Username, Password and other information.

If you want to delete Data Location from your model, click on the specific Data Location and then right-click. When the menu appears, choose **Delete Element**.

See the *The Authoring Reference* for more detailed information on parameters for Connectors.

2.8. Data Location Explorer

The Data Location Explorer view has Data Locations grouped into a shared set known as Data Location Templates, which are a shared folder of predefined Data Locations delivered with the Sybase Aleri Streaming Platform. The required parameters of each Data Location are filled in, but the values may not be correct for your installation. You should review the Data Locations to see if you need to customize the parameters for your needs. Data Location Explorer can discover or explore the individual locations from the Connectors as well as a list of tables and the structure for each data source. But not all Connector types support this discovery feature.

You can use the Data Location Explorer to define data sources for a model while its being built.

2.8.1. Using The Data Location Explorer

Use the following procedure to build a model from scratch using a configured Data Location and the Discovery feature.

1. Since the Data Location is configured with a proper set of parameters, you can right-click on the Data Location, select Discovery to get a set of tables and fields.
2. Select a table from Discovery and drop it on the diagram. This creates a source stream with all fields names and types filled in.
3. Verify that the information is correct in the source stream. If not, corrections can be made in the source stream created in the previous step.
4. Repeat the above steps for each data source in the model.
5. You can then create derived streams in the standard manner.

Here are instructions on how to add an input or output connector to an existing data model:

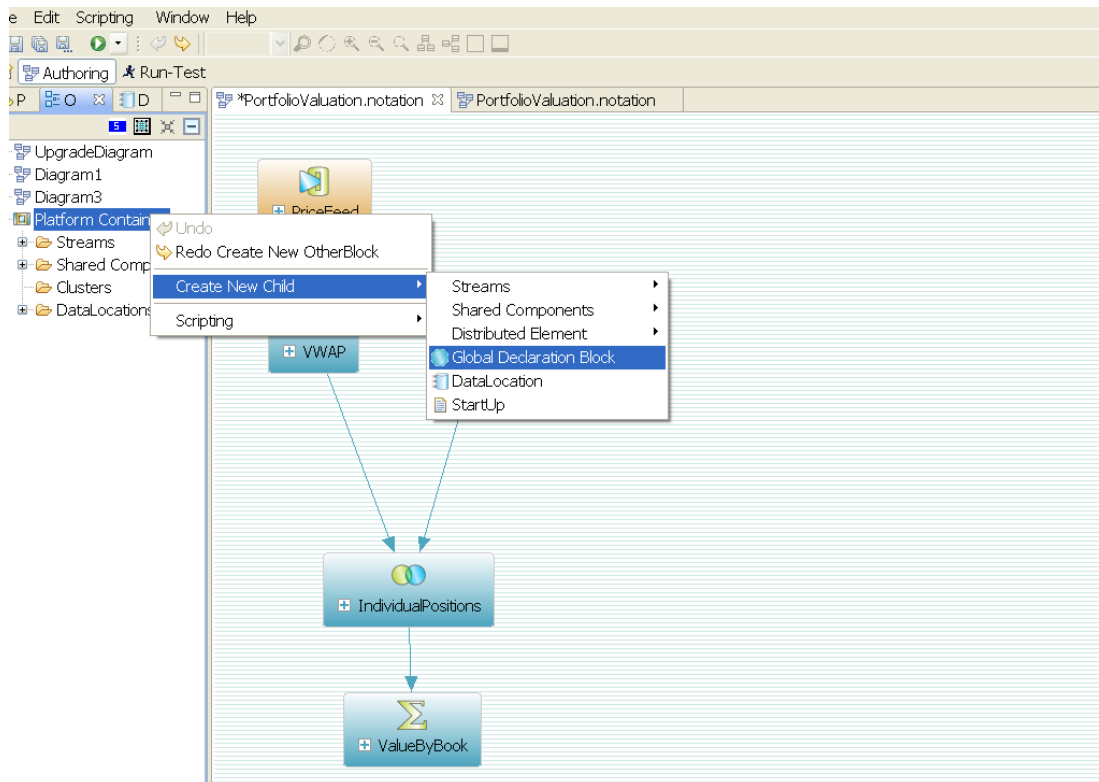
1. Make sure to have Data Locations defined in the Data Location Explorer that define the appropriate data sources that the model will be using.
2. You then go to the source stream that you want to define for either the input or output connection.
3. Click the "add input(output) connector" icon. This adds a new connection named "connection1" to the InputConnections (or OutputConnections) section of the source stream.
4. Double-click on the new connection "Connection1" to open up its definition dialog box.
5. Choose a Data Location from the drop-down menu that defines the type of data source for this connection.

6. Either use the Discovery feature, or manually enter a table name.
7. You have the option of using the field mapping button to define a mapping between the column names of the stream and the columns of the table as it is known to the connector. Otherwise the model will take the order that exists.

2.9. Adding Global Variables and Functions

Global variables and functions can be used by any stream. Global variables can also be altered at runtime via the Command and Control interface. See the *Authoring Reference* for more information about global variables and functions.

To Create and Edit a Global Declarations Block:



- Create a Global block in the Outline view by right clicking on the Platform Container to bring up the context menu. Select **Create New Child > Global Declaration Block**.
- Edit the Global block in the Outline view by right clicking on the Block to bring up the context menu. Select **Edit Global Declarations**.
- Add variable and function declarations in the **Edit Global Declarations** dialog box. Then click **Ok** to save the declarations, or click **Cancel** to abort the changes.

2.10. Defining Streams

This section applies to Visual Authoring only; see *Authoring Reference* for information on how to define streams using Aleri SQL.

This section is organized by stream type. Each stream type, what it does, and how to configure it, is

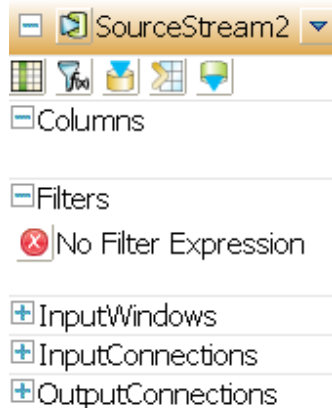
defined in the following sections. See the *Product Overview* for an overview of the concepts of a data model.

Tip

It's generally best to start building a model by defining your data sources, which will automatically create your source streams. Next, build the model following the flow of data. So working from left to right or top to bottom, add derived streams and connect them to input streams. Afterward, the streams should be configured.

2.10.1. Source Stream

A source stream is the entry point for external data to flow or be loaded into the model. All external data enters the model via a source stream. An easy way to create a source stream is to follow the process described in [Section 2.5.2.1, “Adding Streams to the Model”](#) to identify a data source and then establish a connection. This will automatically create a source stream and import the data structure from the source, defining the columns of the source stream. The columns are child elements of the source stream. Some data sources won't be available at the time of authoring, so a source stream can be added to the model manually using the steps described here.



To add a source stream to the model:

1. From the **Palette**, select **source stream** and then click on the diagram in the place where you wish to add the new stream.
2. You may edit the stream name: click the icon to the right of the stream name in the shape header and edit the name. Note that every stream must have a unique name which is the streamID.
3. Add columns using the **Add Column** button in the toolbar of the stream shape. For each column, give the column a name, indicate the data type, and indicate whether the column is a key column.

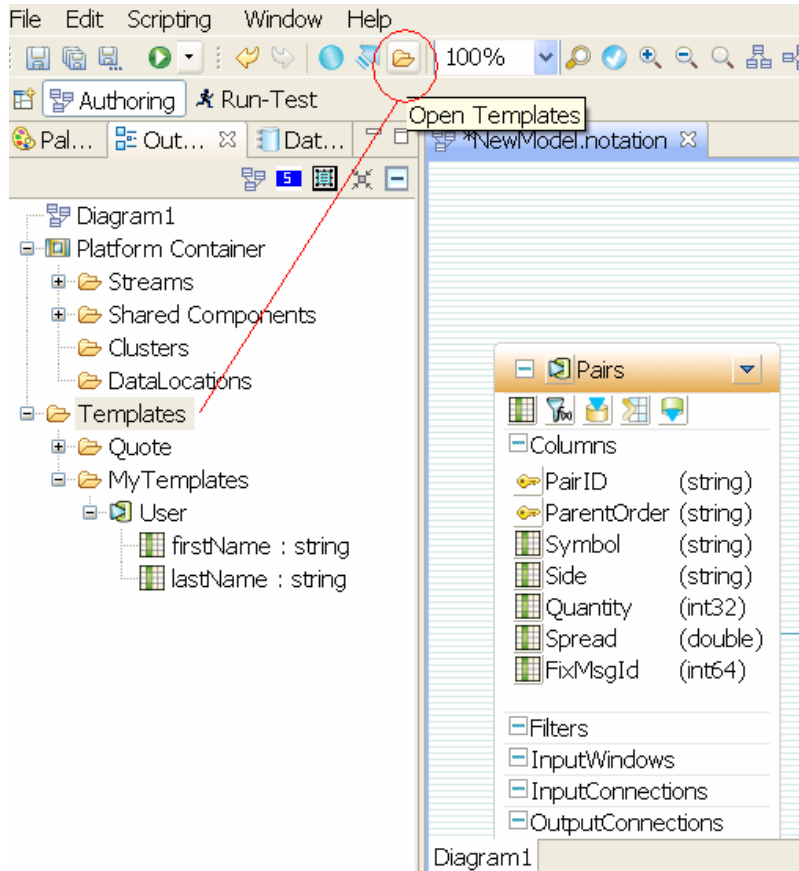
Columns can also be created from the Create New Child context menu on the shape. The column name and datatype can be edited from F2 or by double-clicking the corresponding item in the verbose shape. It can also be edited within entries in the **Properties** window. You can see a list of the different datatypes for columns in [Section 1.3.1, “The Anatomy of a Stream”](#)

The key attribute in a column can be modified by clicking the **Column/Key** button in the verbose

shape or using the **Properties** window. For more information about keys, see [Section 1.3.1, “The Anatomy of a Stream”](#)

4. (Optional): add an Input Window, Input Connection or Output Connection. See [Section 2.11, “Optional Stream Elements”](#) or [Section 2.6, “Connections”](#) for more information.
5. (Optional): set stream options to override defaults. To do this, click on the **Options** element at the bottom of the stream shape to open the Options dialog box. See [Section 2.11, “Optional Stream Elements”](#) for more information.

2.10.2. Adding Streams From A Template Model



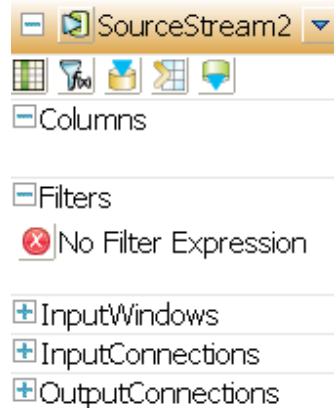
Another way to add a source stream or FlexStream to a model is utilizing the Aleri Studio's collection of template models, which contain pre-configured streams. In order to use this feature, you need to be in the Authoring mode with the Outline view to proceed with the following:

- Select one or more template streams using drag and drop from the Outline view to the diagram to make a copy of the stream.
- You can then use **create/update/delete** template models from the Outline view. A button located on the main toolbar will let you navigate the templates folder. In the Preference dialog box, which you can get by clicking on **Windows**, there is a **Template Directory** where you can read and store template models.
- Note that the display of a template model in the outline is controlled by a new preference option, **Show template models in outline**, which shows a subset of a template model, namely the source

streams and FlexStreams. This option is off by default, so you need to specifically check it in the preference menu. See [Section 2.13.6, “Modify Studio Preferences”](#) for more information.

2.10.3. Aggregate Stream

An Aggregate Stream is a derived stream that takes a single input stream, groups the records according to common key values, and produces a single summary record for each group.

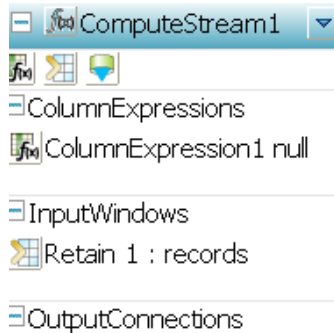


To add an Aggregate Stream to the model:

1. From the **Palette**, select **Aggregate Stream** and then click on the diagram to add the new stream.
2. You can edit the stream name by clicking the icon to the right of the stream name in the shape header and editing the name. Note that every stream must have a unique name which is the streamID.
3. Use the Connector tool to create a connection between the new Aggregate Stream and the stream that will provide the input to this stream by selecting the connector tool from the **Palette**, clicking on the stream in the diagram that will provide the input, and then clicking on the new stream. This adds a connector that shows the data flow and sets the new stream's istream property.
4. ColumnExpressions are child elements of an Aggregate Stream. The ColumnExpressions toolbar can create one in the verbose shape mode or Create New Child context menu on the shape. Its name and value fields can be edited by either F2 or double-clicking the corresponding item in the verbose shape. ColumnExpressions can also be edited in the corresponding entries in the **Properties** window. The key attribute of a ColumnExpression can be modified by clicking the **Column/Key** button in the verbose shape or using the **Properties** window.
5. (Optional) add Group Filter and Group Order elements and edit their associated expressions. See the *Authoring Reference* for more information.
6. (Optional): add an Input Window or Output Connection. See [Section 2.11, “Optional Stream Elements”](#) or [Section 2.6, “Connections”](#) for more information.
7. (Optional): add local variables. See [Section 2.11.2, “Local Variables and Functions”](#) for more information.

2.10.4. Compute Stream

A Compute Stream is a derived stream that takes a single input stream and for every record on the input stream, produces a single output record. Each field in the output record is computed according to a column expression. The Compute Stream can be used simply to normalize data schemas by converting one set of columns to another set of columns. It can also be used with variables, to compute new values (such as net change and moving average) based on current and previous values.

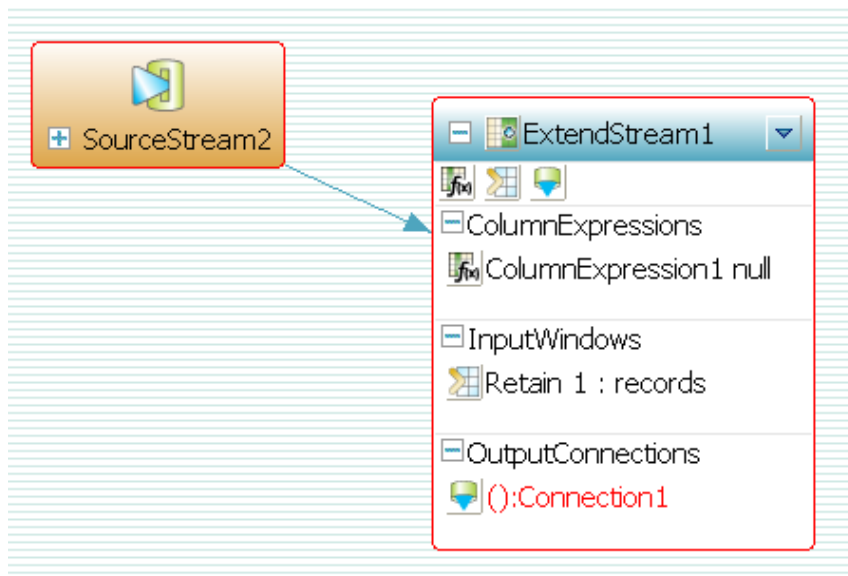


To add a Compute Stream to the model:

1. From the **Palette**, select **Compute Stream** and then click on the diagram to add the new stream.
2. You can edit the stream name by clicking the icon to the right of the stream name in the shape header and editing the name. Note that every stream must have a unique name which is the streamID.
3. Use the Connector tool to create a connection between the new Compute Stream and the stream that will provide the input to this stream: select the connector tool from the **Palette**, click first on the stream in the diagram that will provide the input, and then click on the new stream. This will add a connector that shows the data flow and will set the new stream's istream property.
4. ColumnExpressions are child elements of a Compute Stream. The ColumnExpressions toolbar can create one in the verbose shape mode or Create New Child context menu on the shape. Its name and value fields can be edited by either F2 or double-clicking the corresponding item in the verbose shape. ColumnExpressions can also be edited in the corresponding entries in the **Properties** window. The key attribute of a ColumnExpression can be modified by clicking the **Column/Key** button in the verbose shape or using the **Properties** window.
5. (Optional): add an Input Window or Output Connection. See [Section 2.11, "Optional Stream Elements"](#) or [Section 2.6, "Connections"](#) for more information.
6. (Optional): add local variables. See [Section 2.11.2, "Local Variables and Functions"](#) for more information.

2.10.5. Extend Stream

An Extend Stream is a derived stream that takes a single input stream and for every record on the input stream, produces a single output record. It is like a Compute Stream, except that the columns of the input stream are automatically carried over. You can compute a different value for a column, or add more columns using column expressions. The columns of an Extend Stream are therefore the columns of the input stream and those defined by the column expressions.

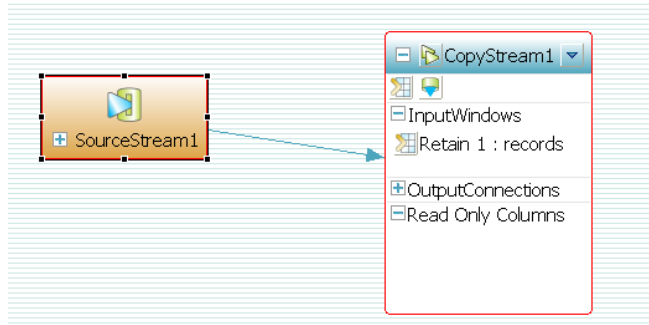


To add an Extend Stream to the model:

1. From the **Palette**, select **Extend Stream** and then click on the diagram to add the new stream.
2. You can edit the stream name by clicking the icon to the right of the stream name in the shape header and editing the name. Note that every stream must have a unique name which is the streamID.
3. Use the Connector tool to create a connection between the new Extend Stream and the stream that will provide the input to this stream: select the connector tool from the **Palette**, click first on the stream in the diagram that will provide the input, and then click on the new stream. This will add a connector that shows the data flow and will set the new stream's istream property.
4. ColumnExpressions are child elements of an Extend Stream. The ColumnExpressions toolbar can create one in the verbose shape mode or Create New Child context menu on the shape. Its name and value fields can be edited by either F2 or double-clicking the corresponding item in the verbose shape. ColumnExpressions can also be edited in the corresponding entries in the **Properties** window. The key attribute of a ColumnExpression can be modified by clicking the **Column/Key** button in the verbose shape or using the **Properties** window.
5. (Optional): add an Input Window or Output Connection. See [Section 2.11, "Optional Stream Elements"](#) or [Section 2.6, "Connections"](#) for more information.
6. (Optional): add local variables. See [Section 2.11.2, "Local Variables and Functions"](#) for more information.

2.10.6. Copy Stream

A Copy Stream takes a single input stream and simply duplicates it. It can optionally include an Input Window setting retention rules that cause it to only include a subset of the input stream. A Copy Stream can be used to enact a self-join since the Join Stream does not allow a stream to be joined to itself, but does allow a stream to be joined to a copy of itself. Copy streams are also used in clustered models.

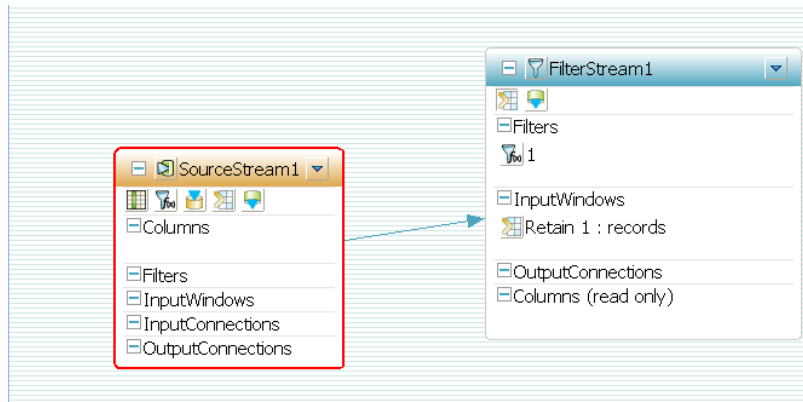


To add a Copy Stream to the model:

1. From the **Palette**, select **Copy Stream** and then click on the diagram to add the new stream.
2. You can edit the stream name by clicking the icon to the right of the stream name in the shape header and editing the name. Note that every stream must have a unique name which is the streamID.
3. Use the Connector tool to create a connection between the new Copy Stream and the stream that will provide the input to this stream: select the connector tool from the **Palette**, click on the stream in the diagram that will provide the input and then click on the new stream. This adds a connector to show the data flow and set the new stream's istream property.
4. (Optional): add an Input Window or Output Connection. See [Section 2.11, "Optional Stream Elements"](#) or [Section 2.6, "Connections"](#) for more information.

2.10.7. Filter Stream

A Filter Stream takes a single input stream, applies a filter, and produces an output stream of all input records that match the filter criteria. At least one filter expression must be defined. Filter expressions must evaluate to an integer (int32) and are "false" if the expression results in a value of zero. Each incoming record will be evaluated against the expression(s) and only records for which all filter expressions evaluate to "true" (non-zero) will be included in the output.



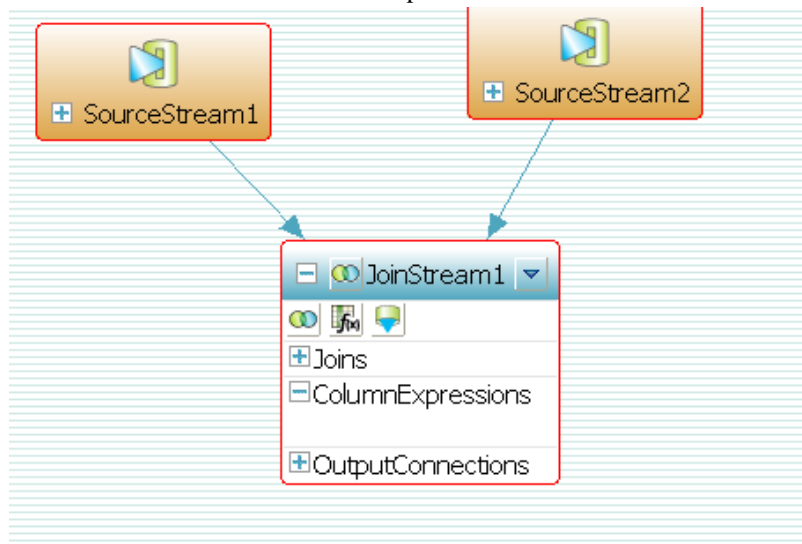
To add a Filter Stream to the model:

1. From the **Palette**, select **Filter Stream** and then click on the diagram to add the new stream.
2. You can edit the stream name by clicking the icon to the right of the stream name in the shape header and editing the name. Note that every stream must have a unique name which is the streamID.

3. Use the Connector tool to create a connection between the new Filter Stream and the stream that will provide the input to this stream: select the connector tool from the **Palette**, click on the stream in the diagram that will provide the input and then click on the new stream. This adds a connector that shows the data flow and will set the new stream's istream property.
4. Define a filter expression: click on the filter expression icon to the right of the "null" filter expression and type in your expression. See [Chapter 3, Expressions](#) for more on expressions. Additional filter expressions can be added using the "Add Filter" icon in the toolbar of the stream shape.
5. (Optional): add an Input Window or Output Connection. See [Section 2.11, "Optional Stream Elements"](#) or [Section 2.6, "Connections"](#) for more information.
6. (Optional): add local variables. See [Section 2.11.2, "Local Variables and Functions"](#) for more information.

2.10.8. Join Stream

A Join stream matches records from two or more input streams to produce a single output stream. Records are "matched" when they have matching values in a column. The output records in a Join Stream contain a set of columns that are computed from column values in the matching input records.



Joins in the data model are like joins in a relational database. Joins have a type property that specifies the type of join (for example, inner, left outer, full outer). When a new record arrives on an input stream, it's compared to all retained data on the other input streams to see if there is a match. If there is a matching record, an output record is produced. If there is no match, an output record will still be produced if it is an outer join. See [Section 5.4, "Joins"](#) for a full explanation of joins.

Joins have the following requirements and constraints:

- There must be at least one Join element. If there are more than two input streams there will be additional join elements.
- Each input stream must be mentioned in at least one of the Join clauses.
- Inner Joins are allowed only when all the input streams are "insert-only". This ensures the correctness of Joins under updates and deletes while maintaining efficiency. The following are examples of "insert-only" streams:

- a source stream with the "insert only" option selected
- a Union Stream, Compute Stream or Filter Stream whose inputs are insert-only
- a Join Stream whose “many” inputs in the many-to-one Joins are insert-only, and whose “one” inputs are “static” streams

To add a Join Stream to the model:

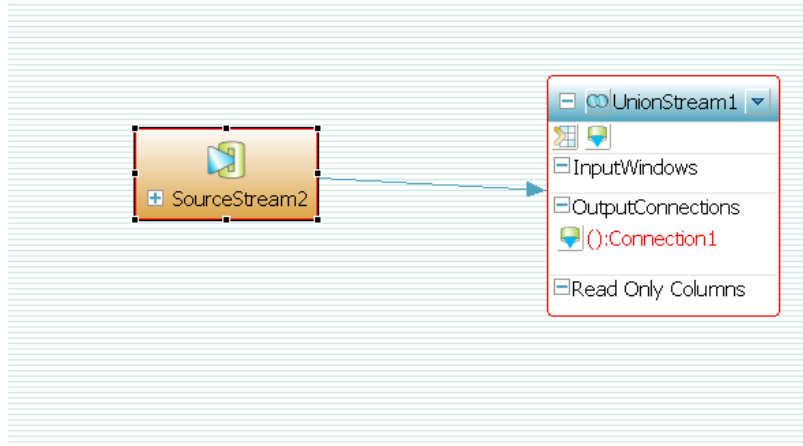
1. From the **Palette**, select **Join Stream** and then click on the diagram to add the new stream.
2. You can edit the stream name by clicking the icon to the right of the stream name in the shape header and editing the name. Note that every stream must have a unique name which is the streamID.
3. Use the Connector tool to create connections between the new Join Stream and streams that will provide the input. First, select the connector tool from the **Palette**, click on the stream in the diagram that will provide an input and then click on the new stream; repeat this for each input stream. This will add connectors that show the data flow and set the new stream's istream property.
4. ColumnExpressions are child elements of a Join Stream. The ColumnExpressions toolbar can create one in the verbose shape mode or Create New Child context menu on the shape. Its name and value fields can be edited by either F2 or double-clicking the corresponding item in the verbose shape. ColumnExpressions can also be edited in the corresponding entries in the **Properties** window. The key attribute of a ColumnExpression can be modified by clicking the **Column/Key** button in the verbose shape or using the **Properties** window.
5. (Optional): add an Input Window or Output Connection. See [Section 2.11, “Optional Stream Elements”](#) or [Section 2.6, “Connections”](#) for more information
6. (Optional): add local variables. See [Section 2.11.2, “Local Variables and Functions”](#) for more information.

2.10.9. Union Stream

A Union Stream merges multiple streams that share a common row structure into a single stream. A Union Stream can have two or more input streams.

For a Union Stream to function properly, all of its input streams must have: the same number of columns, the same datatype for each column, and the same key columns.

The Union Stream has a mergeKeys attribute, which allows it to handle inserts or deletes for the same keys from different inputs. For instance, say a stream receives an insert for a key from input stream Input1, and another insert for the same key from input stream Input2. When the attribute is set to false (the default), the second insert is rejected. If the attribute is set to true, the second insert is turned into an update. Similarly, deletes on the same key from different streams do not cause errors when the attribute is set to true.

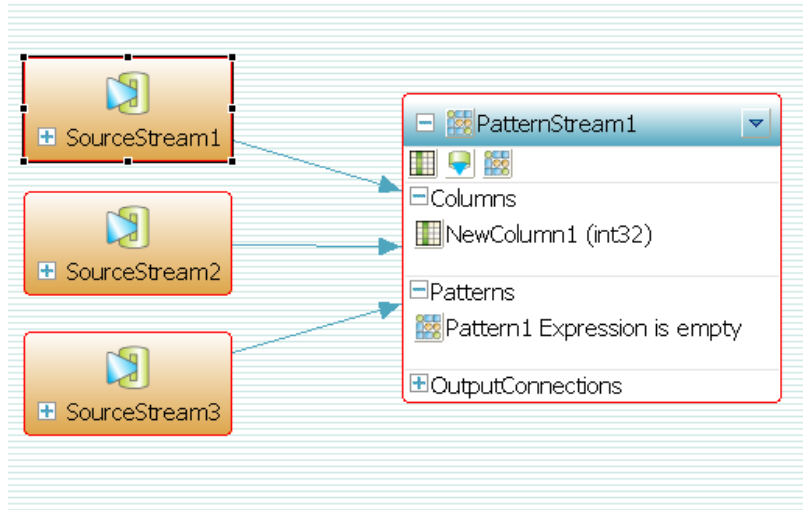


To add a Union Stream to the model:

1. From the **Palette**, select **Union Stream** and then click on the diagram to add the new stream.
2. You can edit the stream name by clicking the icon to the right of the stream name in the shape header and editing the name. Note that every stream must have a unique name which is the streamID.
3. Use the Connector tool to create a connection between the new Union Stream and stream that will provide the input to this stream. First, select the connector tool from the **Palette**, click on the stream in the diagram that will provide the input, and then click on the new stream. This adds a connector that shows the data flow and will set the new stream's istream property.
4. (Optional): add an Input Window or Output Connection. See [Section 2.11, “Optional Stream Elements”](#) or [Section 2.6, “Connections”](#) for more information.

2.10.10. Pattern Stream

A Pattern Stream is used to detect patterns of events across one or more streams. A Pattern Stream takes one or more input streams, and every incoming event is matched against all the patterns defined in the Pattern Stream. When an event pattern is matched and output record is generated.

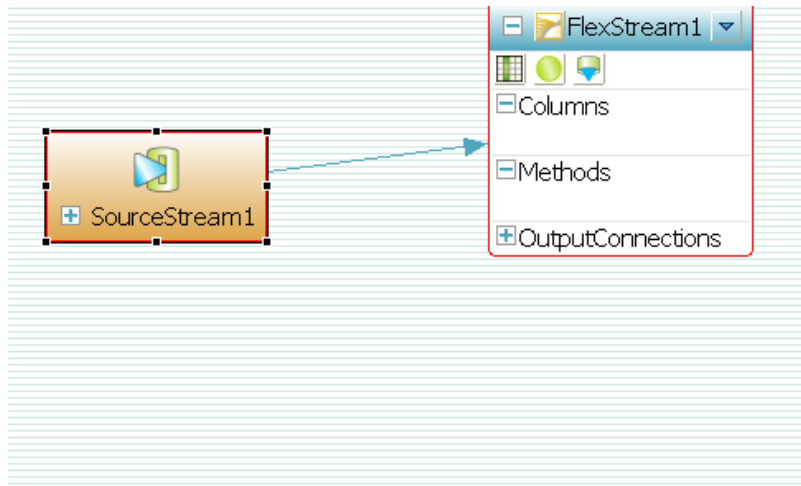


To add a Pattern Stream to the model:

1. From the **Palette**, select **Pattern Stream** and then click on the diagram to add the new stream.
2. You can edit the stream name by clicking the icon to the right of the stream name in the shape header and editing the name. Note that every stream must have a unique name which is the streamID.
3. Use the Connector tool to create one or more connection(s) between the new Pattern Stream and the stream(s) that will provide input to this stream: select the connector tool from the **Palette**, click first on the stream in the diagram that will provide an input, and then click on the new stream; repeat this for each input stream. This will add connectors showing the data flow and setting the new stream's istream property.
4. Define one or more patterns. Click the **Add Pattern** button on the toolbar in the stream shape. Then double click on the new pattern in the "Patterns" compartment of the stream shape to open it in the expression editor. Now define your pattern. Remember that **Ctrl-Space** in the editor brings up a pick list. See [Chapter 4, Pattern Matching](#) for more information.
5. Columns can also be created from a Pattern Stream by using the **Add Column** toolbar on the verbose shape or Create New Child context menu on the shape. The column name and datatype can be edited from F2 or double-clicking the corresponding item in the verbose shape. It can also be edited within entries in the **Properties** window. The key attribute can be modified by clicking the **Column/Key** button in the verbose shape or using the **Properties** window.
6. (Optional): add an Output Connection. See [Section 2.6, "Connections"](#) for more information.
7. (Optional): add local variables. See [Section 2.11.2, "Local Variables and Functions"](#) for more information.

2.10.11. FlexStream

A FlexStream is a programmable stream used to implement custom event processing logic that does not follow the pattern or constraints of the standard relational stream operations. It can take any number of input streams and whenever an input record is received on an input stream, the FlexStream processes it using a method assigned to that input stream. Each method contains a small program written in a special scripting language called SPLASH (Streaming Platform LAnguage SHell). Refer to the *Authoring Reference* for information about SPLASH.

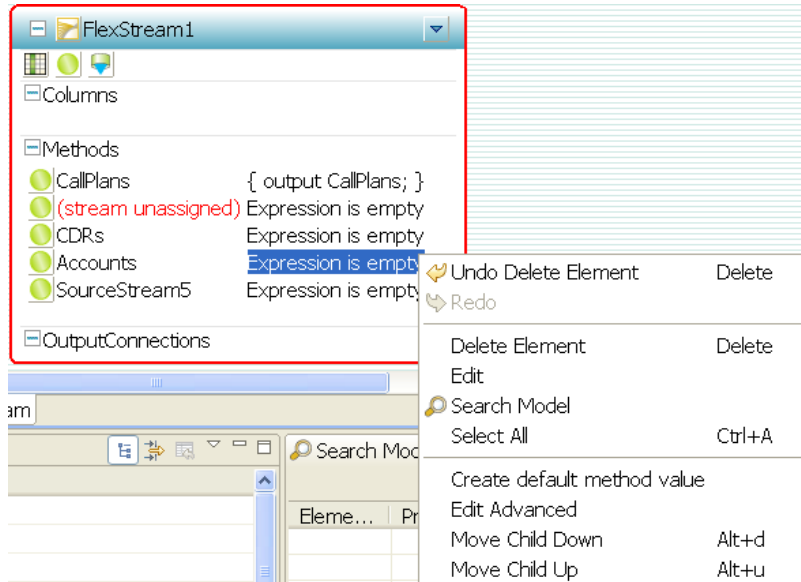


Note

A FlexStream is an advanced tool that requires some programming skills. Many common operations can be performed with the standard relational stream types (for example, Filter, Com-

pute, Aggregate, Join).

To add a FlexStream to the model:



1. From the **Palette**, select **Flex Stream** and then click on the diagram to add the new stream.
2. You can edit the stream name by clicking the icon to the right of the stream name in the shape header and editing the name. Note that every stream must have a unique name which is the streamID.
3. Use the Connector tool to create one or more connection(s) between the new FlexStream and the stream(s) that will provide the input to this stream: select the connector tool from the **Palette**, click first on the stream in the diagram that will provide an input, and then click on the new stream; repeat this for each input stream. This will add connectors that show the data flow and will set the new stream's istream property.
4. The Connector tool creates a method with an empty expression or a default method value by right-clicking on the mouse. The default method value is SPLASH code that copies an input event to an output event. If you are writing script, **Ctrl-Space** in the editor brings up a list of options. See *Authoring Reference* for the SPLASH reference guide.
5. Columns can also be created from a FlexStream by using the **Add Column** toolbar on the verbose shape or Create New Child context menu on the shape. The column name and datatype can be edited from F2 or double-clicking the corresponding item in the verbose shape. It can also be edited within entries in the **Properties** window. The key attribute can be modified by clicking the **Column/Key** button in the verbose shape or using the **Properties** window.
6. (Optional): add an Output Connection. See [Section 2.6, "Connections"](#) for more information.
7. (Optional): add local variables. See [Section 2.11.2, "Local Variables and Functions"](#) for more information.
8. (Optional): add a Timer using the SPLASH scripting language. A timer is a block of code that runs periodically. Start by clicking on the arrow at the top of the stream. Select **Create Timer**. Then double click on the new timer in the "Methods" compartment of the stream shape to open it in the expression editor. You can then write the script for this timer.

Remember that pressing **Ctrl-Space** in the editor brings up a list of options for you to pick. The in-

terval attribute can be modified by using the **Properties** window. The default value is 1, meaning the timer runs every second.

You can also add a FlexStream to a model from a collection of pre-configured template models included in the Aleri Studio. See [Section 2.10.2, “Adding Streams From A Template Model”](#) for directions on how to utilize those templates.

2.11. Optional Stream Elements

See [Section 2.6, “Connections”](#) for information on adding input or output connections.

2.11.1. Input Windows

An Input Window sets limits to the amount of data that a stream receives from an input stream. The limitations are set by size with a fixed number of input records or a time restriction with records only being kept for a specified period.

The method, which restricts size, is defined by setting the type attribute to "records". The target number of records for the window size is specified by the value attribute, and the amount that the window can grow beyond the target number of records is specified using the attribute slack.

The following is an example:

```
<UnionStream id="unionStream" istream="filter1 filter2" store="myStore">
<InputWindow stream="filter1" type="records" value="128" slack="16"/>
  <UnionStream>
```

The above instructions dictate that a window be visible to the Union Stream, a window of the input data stream filter1, that has at most 128+16 = 144 records in it. The current implementation allows the window to grow to 128+16 records, and then truncates it back down to 128 records. This continues for as long as data flows into the stream filter1.

The second method based on a specific time limit is defined by setting the <type>attribute to "time", and by setting the maximum allowed age of a record in seconds to the <value>attribute.

Here is an example:

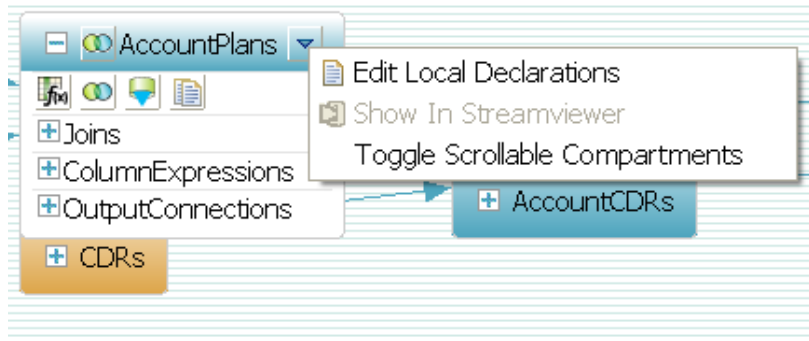
```
<UnionStream id="unionStream" istream="filter1 filter2" store="myStore">
<InputWindow stream="filter2" type="time" value="300"/>
  <UnionStream>
```

The above instructions make a rolling window of the last 300 seconds worth of data from the input stream filter2 to the Union Stream.

2.11.2. Local Variables and Functions

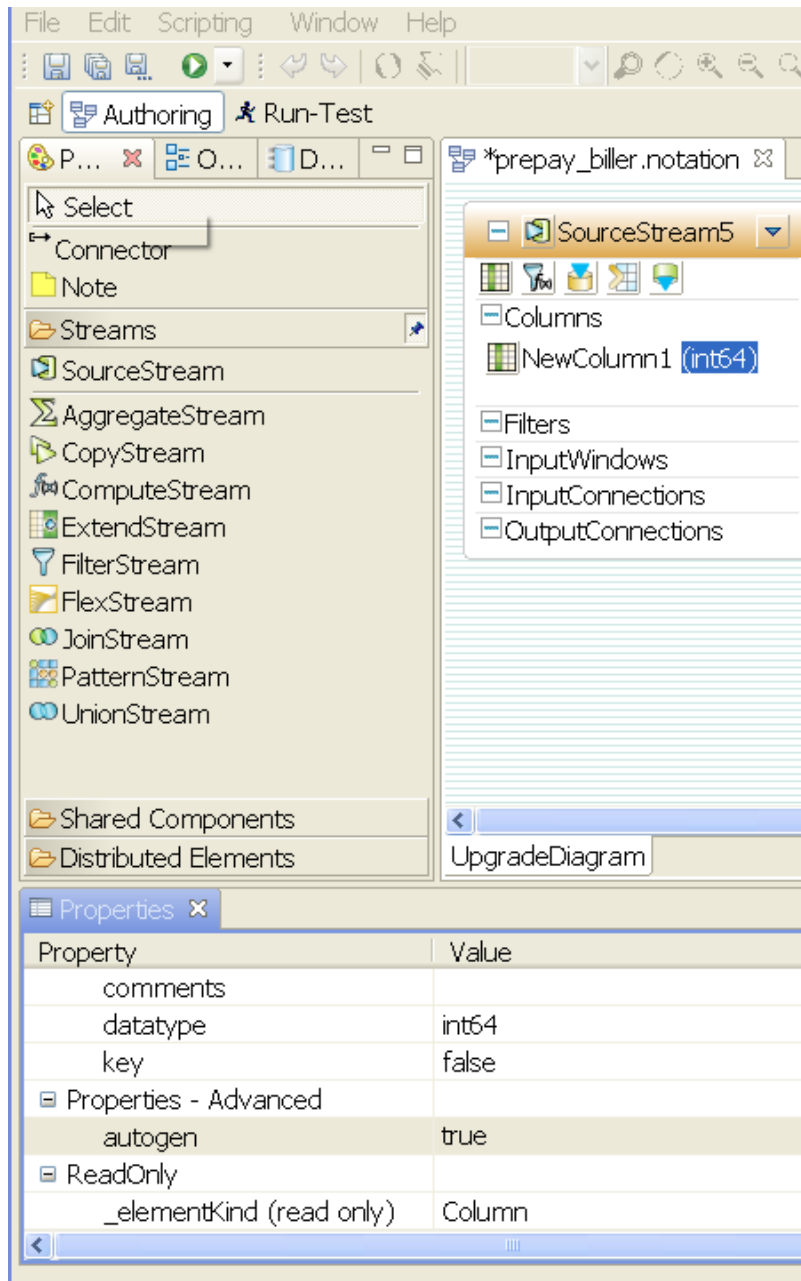
Local variables and functions can be defined for certain stream types. See *Authoring Reference* for more information about the syntax of local variables and functions.

To Create and Edit a Local Declarations Block:



- Create a Local block by clicking on the arrow at the top of the stream. Select **Edit Local Declarations**.
- Add variable and function declarations in the **Edit Local Declarations** dialog box. Then click **Ok** to save the declarations, or click **Cancel** to abort the changes.

2.11.3. Automatically Generated Columns



You have the option with source streams to specify that an automatically generated sequence number be added to each record. This is often used when the incoming data does not have an obvious key, and you want to automatically generate one.

The automatically generated column must be an int64 type. Each source stream can only have one automatically generated column.

To create an automatically generated column, click on the column in source stream. In the **Properties** sheet for the column, you would choose **Properties - Advanced**. Next, set the **autogen** flag to **true**.

For more information, see the *Authoring Reference Manual* .

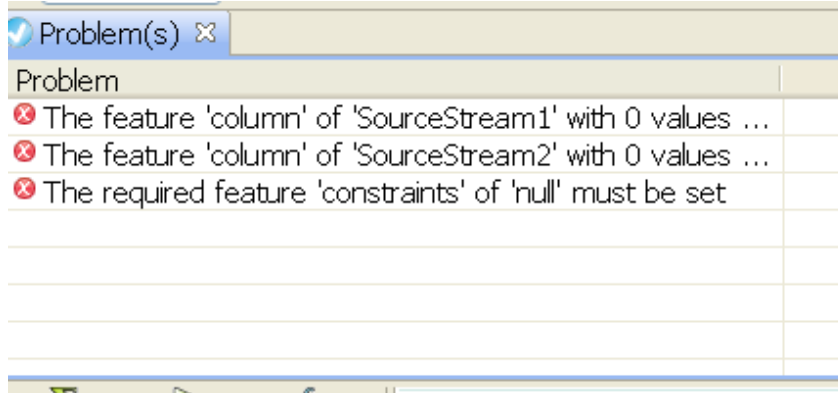
2.12. Checking a Model For Errors

The Aleri Studio can validate a data model: checking for syntax errors, required properties that have not been set, and other problems. A red border on a stream shape in the diagram indicates that the stream has not been properly defined; running a model check will produce error messages indicating the nature of the problem. Problems might exist even if there are no red borders on the diagram so it's recommended to run a check before trying to run the model.

To run a validation report on the full model:

Click on the **Check Model for Violations**  button on the Toolbar.

The Aleri Studio runs a Validation Report and reports any errors in the **Problem(s)** View (tab in lower right corner of workspace). The Problem View will list any problems found. Double-clicking on an error message in the Problem(s) View will highlight the problem element in the Outline View.



Check for model violations on individual elements

To check a single element for errors rather than the entire model:

1. Select one or more elements on the diagram.
2. Click the **Check Model for Violations** button on the tool bar.

Constraint Checks

The Aleri Studio makes every attempt to ensure that data models are built correctly by adding runtime violation checks and through the violations report feature. In addition to checking for required elements and properties, unique ID and Connector constraints are checked.

- Checking is done when renaming elements on the diagram (inline editing of the **ID** field for the shape) or modifying the *ID* field using the Properties area. The *ID* field of the following elements must be globally unique:
 - Stream
 - Rule
 - Store

The Aleri Studio will not allow you to set the **ID** field of two elements defined above to the same value. When the Aleri Studio detects a case of non-unique IDs, it reverts the second **ID** field to the previous value and logs an error message to the Console area. This constraint violation is also caught during reports violations. If for some reason the ID constraint violation is not caught preemptively,

the ID will be displayed in red when the shape is placed on a diagram.

- This constraint is enforced when using the Connector tool to create a connector between two shapes. The Connector Check is done to ensure that:
 - A single store is connected to a stream.
 - Any stream with only one input stream (istream) slot has only one input stream.
 - Invalid creation of a connector between two stream types does not occur.

While the Aleri Studio is able to detect most errors, there are some errors that cannot be detected until run-time. There will be cases where the Aleri Studio error check found no errors, but the Streaming Processor will produce one or more error messages when attempting to load the model.

2.13. Advanced Visual Authoring

This section describes how to use advanced modeling features within the Visual Authoring environment. It is not necessary to use these features to build a model. Full descriptions of these features are found in [Chapter 5, *Advanced Authoring Concepts*](#).

2.13.1. Data Storage Managers

See [Section 5.3, “Stores”](#). for a full explanation of Storage Managers.

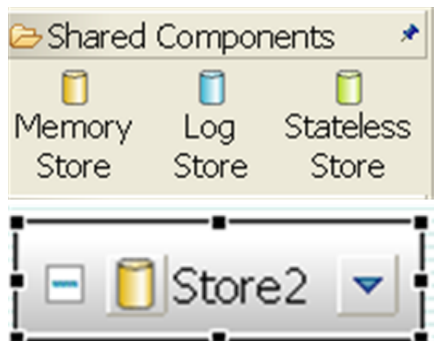
Every stream is assigned to a storage manager (or "store"). The store provides a write-optimized storage mechanism for all the event data retained in the stream. There are three types of storage managers: Memory, Stateless, and Log; the type of storage manager a particular stream is assigned will affect the behavior of the stream. Streams are assigned to a single shared Memory Store by default. Advanced users can create additional stores in order to fine-tune the performance of a model or enable full state recovery for one or more streams using a *Log Store* for disk-based data persistence.

Memory Store	The default store type which holds all records in memory; the records will be lost when the Sybase Aleri Streaming Platform shuts down.
Log Store	Same functionality as a memory store, but all data is persisted to disk using a highly efficient structure that has been write-optimized for high throughput. Enables full state recovery after failure.
Stateless Store	A highly efficient store that only retains sufficient data to process new records. Streams assigned to a Stateless Store cannot be queried and cannot be assigned to a stream that acts as an input to a Join Stream.

To Create a Store:

A store can be created in either of two ways:

- Create a store in the Outline view: right click in the Outline view to bring up the context menu. **Select Create New Child > Shared Components > Store.**
- Add a Store to the diagram: select the desired type of Store from the Shared Components section of the **Palette** and drop it onto the diagram.



Setting the Properties of a Store:

First select the store in either the Outline view or in the diagram. Set the properties of a Store in the Properties view.

Log Stores require the following parameters to be defined:

<i>file</i>	full pathname of a file in which to persist the data
<i>fullsize</i>	the maximum size of the store (in megabytes)
<i>sweepamount</i>	the amount of the log to clean in a single pass (specified in megabytes). If omitted, it defaults to 20% of the fullsize parameter.
<i>sync</i>	whether to commit each record/envelope when it is received by source streams assigned to this Log Store. Setting this to <code>true</code> will negatively impact performance so it's better to issue commits periodically from the data source.
<i>reservePct</i>	the fraction of the log to keep as the free space reserve (specified in percent). If omitted, it defaults to 20 percent.
<i>ckcount</i>	controls the maximum number of records written before writing the intermediate metadata. If omitted, it defaults to 10000.

See [Section 5.3, “Stores”](#) for detailed information on store properties.

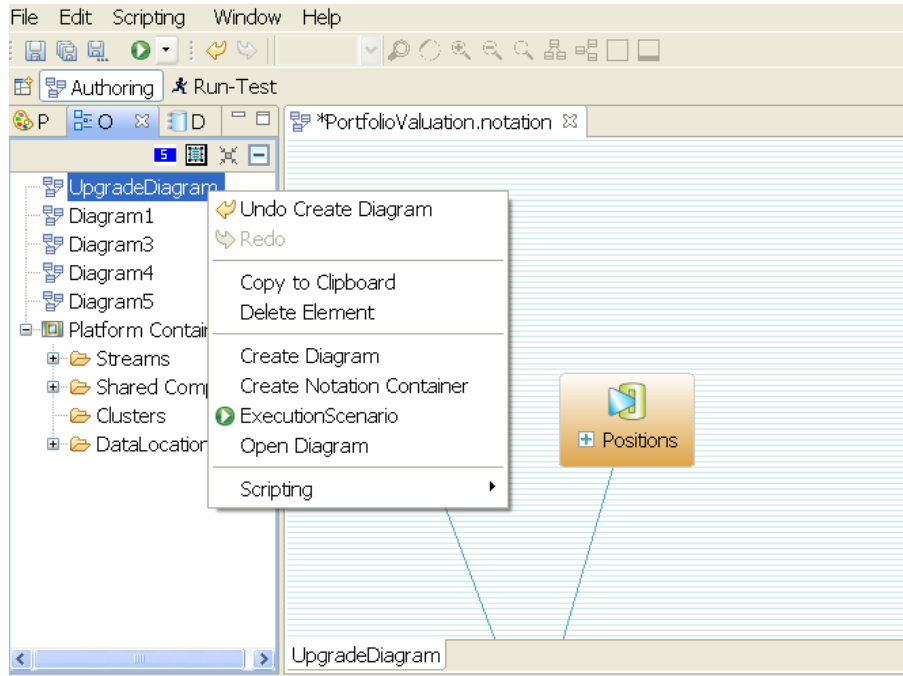
Performance, Indexing

You can choose the type of index used on a Memory Store for performance. Testing is required to determine which indexing is best for the data model. You can also tune for performance by keeping static data in a separate store from those used for dynamic streams.

2.13.2. Working With Diagrams

A single model can have more than one diagram. This is particularly useful if you have a very large model that you want to break into a series of diagrams where each diagram only shows a sub-set of the streams in the model. This section describes how to create new diagrams as well as other advanced techniques for working with diagrams.

2.13.2.1. Create a New Diagram





There are three ways to create a diagram from the Outline view:

- Select an existing diagram in the outline view and right-click to display the context menu. Then select the **Create Diagram** menu choice.
- Select an existing NotationContainer in the Outline view and right click to display the context menu. Then select the **Create Diagram** menu choice.
- You select the **Create a Diagram** in the Toolbar for the Outline view.

If you want to create a copy of an existing diagram in the Outline view, first create a Notation Container using the context menu. Then select a diagram and context menu copy. Finally, select a new Notation Container and select paste.

To select which diagram is displayed in the editor, double-click on a diagram name in the Outline View.

2.13.2.2. Populate a Diagram with All Streams

This convenience function is to be used after a new diagram has been created but is still blank. Right-click in the diagram and select **Populate diagram with all Streams**. This adds all streams that are defined in the model to the diagram and show connectors between them. After the diagram has been populated with shapes, you can lay them out manually or let the Aleri Studio apply an automatic layout by selecting either the horizontal or  or vertical  layout button in the toolbar.

2.13.2.3. Add Existing Elements to a Diagram

Do this when elements of the data model are not currently shown in the diagram.

1. In the Outline area, select one or more data model elements.
2. Drag the elements from the Outline to the diagram. The elements will be added to the diagram.

2.13.2.4. Show Related Streams

The **Show Related Streams** feature can be used to add relatives of a selected stream to the diagram. To access this option, right-click on a shape in the diagram and select **Show Related Streams** from the context menu. In the submenu that appears, you can choose one of the following options:

1. **All Sources and Targets 1-level** adds the immediate source and target streams of the selected stream to the diagram.
2. **All Sources and Targets n-levels** adds all source and target streams of the selected stream to the diagram, all the way up and down the dataflow.
3. **All Sources 1 Level** adds all immediate source streams of the selected stream to the diagram.
4. **All Sources n-levels** adds the full-source flow for this stream to the diagram.
5. **All Targets 1-level** adds the immediate target streams for the selected stream to the diagram.
6. **All Target n-levels** adds all downstream nodes of the selected stream to the diagram.

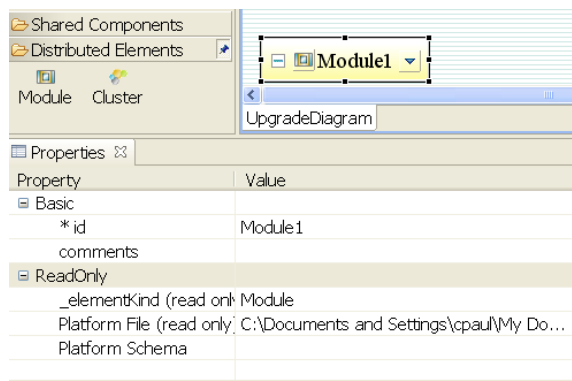
2.13.2.5. Show Shared Components of a Stream

The **Show Related Shared Components** feature can be used for the store used for a selected stream to the diagram. To do this:

1. Select one shape in the diagram.
2. Right-click the stream shape, select **Show**, and then select **Related Shared Components**.

2.13.3. Create Modules

Modules are used to break a large model into pieces. Each module consists of one or more streams. This is typically done to enable a model to run in a distributed configuration (on a cluster for example). See [Section 5.5, “Distributed Models and Clustering”](#) for a full explanation of distributed models before you use the Aleri Studio to create modules.



To Create a Module in the Diagram:

1. From the **Palette**, click **Module** and then click on the diagram to drop the new shape onto the desired area of the diagram.

2. In the Properties area, edit the ID of the Module in the Id property.
3. If desired, type a comment of the Module in the comments property.
4. Update any of the remaining properties as necessary.

To Create a Module in the Outline view:

1. From the Outline view, right-click **Platform Container**, move the mouse cursor over **Create Distributed Element**, and click **Module** in the submenu.
2. In the Properties area, edit the ID of the Module in the Id property.
3. Update any of the remaining properties as necessary.

2.13.4. Search the Model

The Search Model view allows you to search a model for string properties contained within elements and displays the results in the view. The found elements can then be modified individually in the properties window or by selecting multiple found items and performing the replace action.

To Search the model:

- Use the **Search** button on the main toolbar or Use the Search Model view and activate the context menu to search.

The results of the search sorted by type of element are displayed in the view.

- Select an element in the Search view to make its properties available for editing in the Properties view.
- You can also select one or more items in the Search view and choose to replace the found string occurrences with a new string (using the context menu item to replace).

Note:

This will replace all matching occurrences of the string in each property containing the match, that is, if the found text is FOO and the replace text is BAR. The string FOO_MY_FOO will become BAR_MY_BAR after a replacement action.

- You can also drag an item from the Search view and drop it on the diagram to create a shape (if it is a top level shape).

Here is a list of the primitive properties that are searched for each element kind:

Primitive	Properties
Stream, Store, Module, Cluster	id
Column	name
ColumnExpression, Group, GroupOrder, FilterExpression	value
Method, Pattern	name, value
Join	constraint

Primitive	Properties
Node	machine

2.13.5. View the Properties of the Model

All top-level data model elements, which are all represented on the Palette, are child elements of a “Platform Container”. Selecting the Platform Container in the Outline View allows you to see and modify the top-level properties of the data model. But there is a read-only property called Platform File, which links the .notation file to the underlying .xml file that contains the model in AleriML. There is also a property called Platform Schema, which is used to set the location of the platform.xsd file. The value in the Platform Schema property must be a correct and valid location in order to open an AleriML file in the Aleri Studio.

There is a button in the Properties Window on the tool bar **Restore Default Value** that you can press to restore the property's defined default value in platform.xsd. If there isn't a defined default value, then the property is removed for the item, which means it won't be visible in the saved XML file.

2.13.6. Modify Studio Preferences

You can set your user preferences for the Aleri Studio with the Preference dialog. It can be accessed from the Window menu in the menu bar. Note that a model must be reopened after setting preferences for changes to take effect.

When it first appears, the Preferences window shows **Authoring Preferences**. To choose any set of preferences for viewing/editing, click on one of the categories listed in the tree display on the left.



Authoring Preferences

In this dialog box, the following preference settings can be viewed and edited:

- **Allow duplicate elements on diagram** : check this box if you want to allow dropping multiple instances of the same element on the diagram from the Outline view.
- **Display diagram overview when opening a diagram**: check this box if you want to display the overview rectangle on the right hand side of the diagram canvas, which is useful for navigating around large diagrams.
- **Enable full qualified names**: check this box if you want to display the entire qualified path of the shape element in the **Name** compartment. On a diagram, the id field displays a fully qualified path. For example, you can use Module:id or just id if it is contained within the top-level Platform Container.

Compartment items that reference these types of elements are also displayed with their fully qualified paths. This is typically done for distributed modeling when placing various top-level elements (Streams, Stores, and Rules) into different modules.

- **Create new shapes in Iconic Mode**: if you check this box, new shapes on the diagram will initially be displayed with just its name type. If you deselect this check box, it will be displayed in Compartment mode which is more verbose, showing child elements within the shape.
- **Create new shapes with scrollable compartments**: check this box if you want to have a scroll bar next to the columns in a stream.
- **Show Diagram Background Pattern**: check this box if you want the background of the model to be plain without lines. If you deselect this check box, the background pattern will go to the default which has lines.

- **Show Welcome page on startup:** check this box if you want the Aleri Studio Welcome page to be displayed on startup.
- **Show Comments in tooltip:** check this box if you want icons in your model to display the collapse button  in the upper left corner of the shape or the expand button .
- **Advance expression editor as a Dialog:** check this box if you want to have dialog boxes to edit text for the child elements of a shape. You must double-click on the left side of the mouse on the element to bring up this box.
- **Restore Previous Window Size on Startup:** if you check this box, your previous Window size will be restored when you startup your Aleri Studio model.
- **Enable dynamic error marking in editors:** if you check this box, the text in the editor will be continuously validated, and it will display a red-X in the left hand margin at the point the text in the source code editor does not validate correctly.
- **Show line numbers in editors:** if you check this box, Aleri Studio text editors can show the line numbers in the left-hand margin of the editor.
- **Show template models in outline:** if you check this box, you can see the template model in an outline that shows a subset of the template, namely the source stream and FlexStream.
- **Maximum number of children to display per outline node:** which lets you control the number of elements that are shown for this associated node. If the number exceeds the maximum, it only displays the amount set in the preference, and the last element shown has an error indicator icon noting the display has exceeded the maximum. While these settings affect all parent-to-child relationships in the outline, it is mostly intended to control the number of displayed columns for very wide streams (for example, 20,000 columns per stream).
- **Maximum number of shape compartment items to display per diagram shape:** which lets you control the number of elements that are shown for this associated node. If the number exceeds the maximum, it only displays the amount set in the preference, and the last element shown has an error indicator icon noting the display has exceeded the maximum. While these settings affect all parent-to-child relationships in diagrams, it is mostly intended to control the number of displayed columns for very wide streams (for example, 20,000 columns per stream).
- **Enable dynamic error making on diagram editor:** This preference is a way to tune the redrawing of the diagram after a semantic change occurs. However, if you select this preference, the error-highlighting on the shape might be incorrect due to a semantic change that causes the model to incorrectly verify. As the diagram is not immediately refreshed, it may get a false positive error condition or vice versa. The remedy is to perform on the diagram context menu choice (**Show>Refresh Shape Visuals**).

The bottom of the Preference dialog has five other categories where you can find information about your model.

- **Examples Directory**
- **Size of Recently Used File List**
- **Data Location Home**
- **Templates Directory**

Execution Defaults

Execution Defaults are used from the Run-Test Perspective.

This dialog box allows you to view and edit the following preference settings:

- **Watch Fetch Limit:** Watch values can have many rows of data associated with them. The Watch Fetch Limit allows control of how many rows of data will be stored in the Watch value by the Aleri Studio user interface. The range for this value is 1-100 (for performance purposes). Default value: 10.
- **Studio precision:** Set the value in this field to specify the precision of numbers displayed in the numeric columns of the Streamviewer. Default: 2.
- **Use SSH for Studio Execution:** Check this box to execute the studio binaries through SSH. This option provides the ability to run the Aleri Studio on one machine and execute the Streaming Processor binaries on a separate server machine. If you do not select this option, the Aleri Studio must be installed and run on the same machine on which the Streaming Processor binaries or the server exist.
- **Debugger Initialize with All Base Data:** Setting this preference to `true` will create the Debugger SpSubscription with the SpSubscription.BASE flag. This means that all event data from the Sybase Aleri Streaming Platform will be sent through the debugger graph. Setting this preference to the default `false` will create the Debugger SpSubscription with the SpSubscription.NOBASE flag. This means that only events from this point in time forward will be sent through the debugger graph.
- **Display Query Output as Comma Separated Values:** Setting this preference to `true` will format the output as comma separated values. Setting this preference to `false` (which is the default) will display the output in tabular form.
- **Query Page Default Column Width:** This preference allows you to set the number of characters for SQL Query Output. The default is 25 characters.
- **Use Pulsed Subscribe on Streamviewer Subscription:** You can use this preference to start a Streamviewer subscription with pulsed subscribe.

Defaults

Platform Defaults are used from the Run-Test Perspective.

- **Platform file source control header:** This entry box is used to hold a standard source control header block that will be inserted as an XML comment when new `platform.xml` files are created. The block is empty by default.

Data Input Defaults

Data Input Defaults are used from the Manual Data Input view in the Run-Test Perspective or Execution Scenario Perspective. The default values on this preference page are used to populate the Manual Data Input View for the selected stream.

- The various color preferences indicate the background color of the associated shape in the diagram.
- **Text Editor Defaults** are used for the various text editors used within Aleri Studio (SQL, Javascript, and SPLASH).

Text Editor Defaults

The settings in this dialog box control how elements of expression scripts, SPLASH scripts, Javascript scripts, and SQL scripts are displayed in the Aleri Studio text editing boxes.

- **Comment Color** is the syntax highlighting color for comments in the language text editor (SPLASH, SQL, or Javascript).
- **Keyword Color** is the syntax highlighting color for keywords in the language text editors.
- **String Color** is the syntax highlighting color for strings in the language text editors.
- **Other Color** is the syntax highlighting color for all remaining words in the language text editors.

2.14. SQL Authoring

Models can be created in Aleri SQL in the Aleri Studio using the SQL Editor. Existing models can be opened and edited. It is not required that you use the SQL editor in the Aleri Studio; any text editor can be used to create and edit models in Aleri SQL.

Aleri SQL provides a familiar environment for those with experience writing SQL queries since it is based on the ANSI SQL99 standard, with extensions for the event stream environment. Users that are comfortable with SQL may find this a more efficient way of creating a model than using Visual Authoring.

Models created in Aleri SQL can be run and tested in the Aleri Studio and they can also be imported into the Visual Authoring environment. However, if changes to a model are made in the Visual Authoring environment (or to the underlying AleriML model), those changes will NOT be reflected in the original SQL model.

2.14.1. The Studio SQL Editor

Start by either creating a new SQL model or opening an existing SQL model. See the *Authoring Reference Manual* for details. If you are new to Aleri SQL, the best place to start is by opening an example. The `portfolio_valuation` example would be a good starting point. You can find it in `My Documents/Aleri/examples/models/applications` on a Windows machine or `<platform_home>/examples/models/applications` on a Linux® or Solaris™ system.

The SQL editor is a basic text editor with syntax highlighting of keywords and code completion for the Aleri SQL syntax.

```

File      : prepay_biller.sql
Description: A model to bill customers for minutes used beyond the calling
            plan minutes.
Date      : May 11th, 2005
*****/

--Define The Stores
CREATE STORE StaticStore MEMSTORE
CREATE STORE CDRsStore MEMSTORE
CREATE STORE AccountCDRsStore MEMSTORE
CREATE STORE AccountSummariesStore MEMSTORE
CREATE STORE AuthsStore MEMSTORE
CREATE STORE AccountAuthsStore MEMSTORE
CREATE STORE AccountAuthsMinsStore MEMSTORE

-- Accounts Base Table Definition
CREATE TABLE Accounts (
    AccountID      int,
    FirstName      varchar,
    LastName       varchar,
    Street         varchar,
    City          varchar,

```

See *Authoring in SQL* for the Aleri SQL syntax.

You can use the following resources for editing:

- At any point in the script, press **Ctrl-Space** from the editor to display the code completion choices available. The code completion list displays global functions and coding templates for Aleri SQL. The entry you select from the list is pasted into the editor at the current cursor location.
- Click **Check Syntax** from the context menu. The SQL script is examined for syntax errors. Any errors found are displayed in the Problem View.
- Click **Toggle Comment** from the context menu. This function “comments out” the currently selected lines.

To save the SQL test file, click **Save** from the toolbar or menu.

To cancel the changes made to the SQL file, close the editor without saving.

2.14.2. Running and Testing an SQL Model

When you are ready to run your model, click the **Build and Run SQL** button on the Toolbar. This will convert the model to AleriML and start the Sybase Aleri Streaming Platform, loading the model from the xml file. At this point, all of the facilities of the Studio Execution Perspective are available.

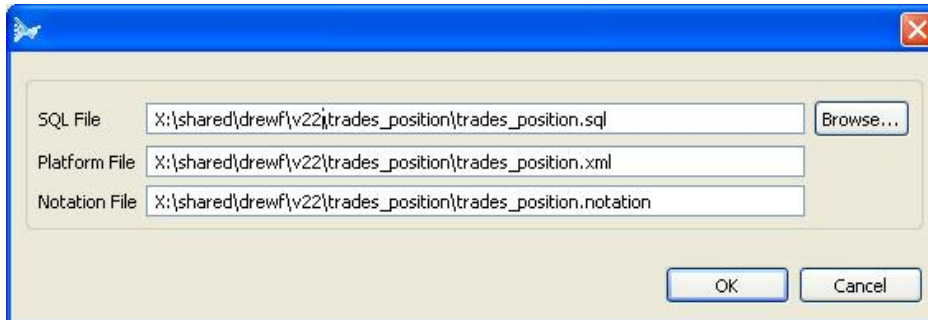
2.14.3. Convert an Aleri SQL model for Visual Authoring

The Aleri Studio can convert a data model created in Aleri SQL into a visual model that can then be

viewed and edited in its Visual Authoring environment. Refer to the *Authoring Reference* more information on creating data models in Aleri SQL.

Note

If you modify the model using the Visual Editor, changes will NOT be applied to the original SQL model.



If the SQL model is already open in the Aleri Studio SQL editor, click on the **Convert SQL** button in the toolbar.

1. Change the file names in the fields in the dialog box if necessary as follows:
 - **SQL File** is the name of SQL file to be imported. You can use the **Browse** button to select this filename.
 - **Platform File** is the name of the new AleriML xml file that will be created containing the data model.
 - **Notation File** is the name of the new Visual Authoring `.notation` file that will be created and associated with the model in the `.xml` file.
2. Click **OK**.

Alternatively, if you have not opened the SQL model in the Aleri Studio, you can convert it prior to opening it by bringing up the Convert SQL dialog box from the File menu on the menu bar.

1. From the **File** menu, click **Convert SQL**. The Aleri Studio displays the Convert SQL dialog box.
2. Update any of the following properties as necessary:
 - **SQL File** is the name of the SQL file to be imported. You can use the **Browse** button to select this filename.
 - **Platform File** is the name of the new AleriML xml file that will be created containing the data model.
 - **Notation File** is the name of the new Visual Authoring `.notation` file that will be created and associated with the model in the `.xml` file.
3. Click **OK**.

The Aleri Studio invokes the `sql2xml` utility, which converts the Aleri SQL file to AleriML. If this conversion is successful, a new Aleri Studio `.notation` file will be created and linked to the new XML

file and the .notation file. The .notation file will then be opened for editing.

Chapter 3. Expressions

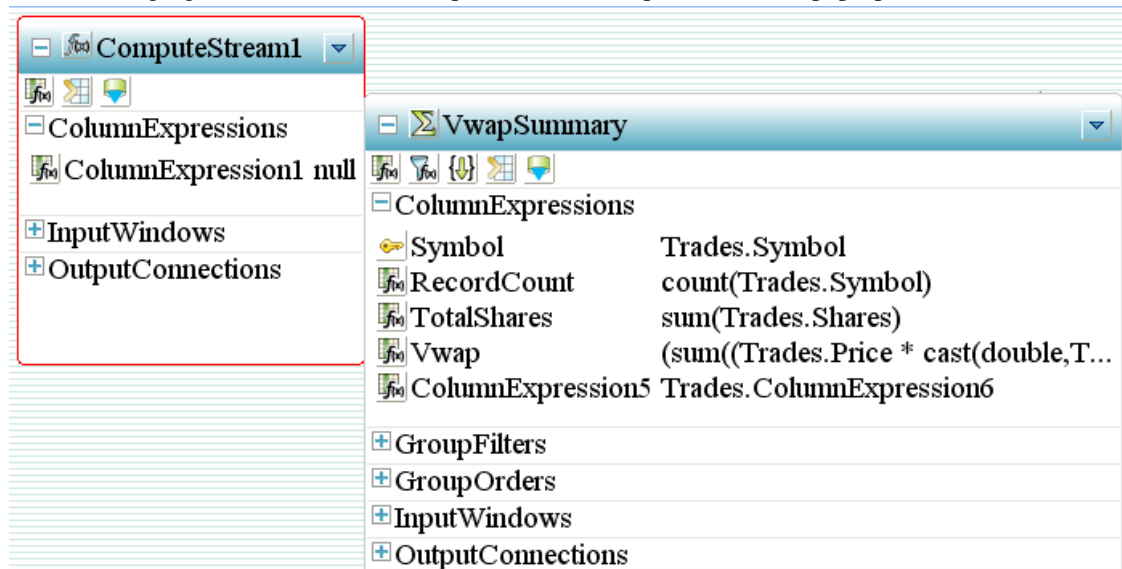
The Aleri Expression Editor provides a convenient way to author complex expressions that are associated with streams.

3.1. Expression Types

The following are the types of expressions that a stream may contain:

- Filter Streams must have a filterExpression.
- Compute Streams and Join Streams have a columnExpression for each column.
- An Aggregate Stream must have a groupExpression and a columnExpression for each column and optionally can have groupFilter and groupOrder expressions.

The following figure illustrates a valid expression in the Expression Editor pop-up:



3.2. Editing Expressions

To Edit an expression:

1. Select either a columnExpression, filterExpression or one of the other other child expression elements of a stream. There are several possible ways of doing this:
2. Right-click on the expression element in the diagram (displayed in Compartment mode) and choose Edit Expression.
3. Double-click on the expression element in the diagram (displayed in Compartment mode).

The following is another method:

1. In the **Expression** dialog area, type in the expression.
2. Click **Check Syntax**. The program examines expression's syntax for errors and displays a message box showing any syntax errors.
3. If necessary, correct the expression's syntax.
4. Click **Ok** to save the expression, or click **Cancel** to abort the changes made to the expression.

3.3. More Editing Tips

Other tips include:

- **F2** will invoke inline editing of ColumnExpression value, FlexStream method or Pattern Stream pattern.
- Double-click will open advanced editor for the ColumnExpression, FlexStream method or Pattern Stream pattern. The Advanced Editor is a dialog or a full screen text editor depending on how you set user preference for "Advanced Expression Editor as Dialog.". **F3** will open other editor. If advanced expression editor as dialog preference is set to true, double-click will open a dialog for editing the ColumnExpression. **F3** will open the ColumnExpression in the full-screen editor.
- When an expression editor is active, pressing **Ctrl-Space** will open context assistance for the editor. The context assistance will display a list of functions, datatypes, and input streams.

Chapter 4. Pattern Matching

The Sybase Aleri Streaming Platform's pattern-matching feature is a powerful tool that identifies trends and threats in real time.

Pattern matching is implemented through Pattern Streams created with rules based on Microsoft® Inc.'s Language Integrated Query (LINQ) syntax, matching syntax from functional languages and linear-time temporal logic. See [Section 2.10.10, “Pattern Stream”](#) for the steps to create a Pattern Stream.

4.1. Creating Pattern-Matching Syntax

Sybase's pattern-matching syntax allows multiple rules to be used with one Pattern Stream. That means multiple rules can be used to look for the same type of event, boosting the power of pattern-matching capabilities.

In writing syntax for the Sybase Aleri Streaming Platform's pattern matching feature, all patterns have the form:

- within <time expression>
- from <patterns>
- on <temporal statement>
- on <SPLASH statement or block>

Time expressions can be expressed in seconds, minutes, or hours, as in:

- 5 seconds;
- 10 minutes;
- 20,000 hours;

The temporal expressions have the form:

- event1 or event2
- event1 and event2
- not(event1)
- event1 fby event2

or any combination, such as :

event1 and (event2 or not(event3)) fby event4 fby not(event5 and event6)

The patterns look like:

```
<input stream name>[ fieldname1 = <variable or constant; ... ] as event_name>
```

If you have two input streams named Buy and Sell, with fields "symbol" and "price" (and possibly others since not all fields have to be specified in the pattern) then the following are legal patterns:

```
Buy[price=1.56; symbol='IBM'] as event1
Sell[price=p1; symbol='GOOG'] as event2
```

4.2. Examples of Pattern Matching

Here are three real-life business situations where pattern matching can be utilized.

A user wants to see if a broker buys stock before purchasing the same shares for a customer and subsequently, the broker sells his or her own shares, creating a "buy ahead" event. The following example illustrates how to specify that pattern.

```
within 5 minutes
from
  BuyStock[Symbol=sym; Shares=n1; Broker=b; Customer=c] as Buy1,
  BuyStock[Symbol=sym; Shares=n2; Broker=b; Customer=c] as Buy2,
  SellStock[Symbol=sym; Shares=n1; Broker=b; Customer=b] as Sell on Buy1 fby Buy2 fby Sell
output [Symbol=sym; Shares=n1; Broker=b];
```

The on clause specifies the temporal relationships between events. The fby construct is an abbreviation of "followed by." meaning that one event is followed by another. That event might not be the next event: it must just occur at some point after the preceding event.

Note the use of the variables sym, n1, n2, b, and c. These are bound to the values of the matching fields. Because the same variable sym is used in three patterns, the values in the three events must be the same (this is what is meant by "unification"). Different variables might have the same value, though (for example, n1 and n2).

Another example shows Boolean operations on events. The rule describes a possible theft as a product barcode was read on a shelf, and the same item was also identified by scanner located near the door. However, the barcode for the item was not scanned at any of the store's cash registers.

```
within 12 hours
from
  ShelfReading[TagId=tag; ProductName=pname] as onShelf,
  CounterReading[TagId=tag] as checkout,
  ExitReading[TagId=tag; AreaId=area] as exit
on onShelf fby not(checkout) fby exit
output [TagId=t; ProductName=pname; AreaId=area];
```

You can also use the Boolean "and" and "or" operations inside the "on" clause.

The following example shows how to raise an alert if a user tries to login to an account unsuccessfully three times.

```
within 5 minutes
from
  LoginAttempt[IpAddress=ip; Account=acct; Result=false] as login1,
  LoginAttempt[IpAddress=ip; Account=acct; Result=false] as login2,
  LoginAttempt[IpAddress=ip; Account=acct; Result=false] as login3,
  LoginAttempt[IpAddress=ip; Account=acct; Result=true] as login4 on
  (login1 fby login2 fby login3) and not(login4) output [Account=acct];
```

This example uses constants true and false instead of variables in part of the pattern. This means that the Result fields must hold exactly these values.

People who break into computer systems often scan a number of TCP/IP ports for an open one, and attempt to exploit vulnerabilities in the programs listening on those ports. Here's a rule that checks whether a single IP address has attempted connections on three ports, and whether those have been followed by the use of the "sendmail" program.

```
within 30 minutes
from
Connect[Source=ip; Port=22] as c1,
      Connect[Source=ip; Port=23] as c2,
      Connect[Source=ip; Port=25] as c3
      SendMail[Source=ip] as send
on (c1 and c2 and c3) fby s1
output [Source=ip];
```

Again, notice the use of constants in the pattern.

Chapter 5. Advanced Authoring Concepts

This chapter provides detailed information on the more advanced features of the Sybase Aleri Streaming Platform.

5.1. Retention

Input Window Any stream may have an input window defined for each of its input streams. An input window limits the Stream's "view" of incoming data — by curtailing the number of records coming in from an input stream. The windows on an input stream may be based on the age of incoming records (time based retention) or total number of records (count based retention).

For example:

- For a time-based retention policy, a time window (specified number of seconds) is defined on the input stream. Records older than the specified time are removed from the stream's input so that the computed value of the stream reflects only data based on the defined window.
- For a size-based retention policy, a numeric limit and a “slack” value are specified. Once the number of retained records reaches the value (`limit + slack`), the oldest records are purged to return the table to the size defined by `limit`. (The number of records actually purged at this time is equal to the `slack` value.) This increases efficiency by allowing purging in batches rather than purging one record each time a new record arrives.

Currently all streams with Memory and Stateless Stores support input windows, except for Join Streams and FlexStreams. Until this restriction is removed in a future version of the Sybase Aleri Streaming Platform, an available workaround is front-end the Join Stream or FlexStream with a Copy Stream that has an input window defined on it.

If any stream, except a Copy Stream or source stream, is defined having a LogStore, it cannot support an input window. This restriction is a result of the fine-grained recovery capabilities inherent in Sybase's Log Store technology. This last restriction on streams defined with Log Stores also may be easily overcome by front-ending the stream with a Copy Stream that has an input window defined on it. If a stream has multiple inputs, an optional input window may be defined on each of the streams inputs.

A source stream may also have an input window specified, even though a source stream has no input stream. The input window simply defines a limiting window on the data in the source stream.

If an Aggregate Stream has an input window defined, the optimizations available in the case of group additive functions are no longer possible. A copy of the group aggregation index must be held and maintained within the stream, to determine if valid updates are received. In this case, it is almost always better to front end the Aggregate Stream with a Copy Stream, and put the input window on the Copy Stream.

A time-based input window uses the record creation timestamp, rather than the timestamp of the last record update.

5.2. Expiry

A stream can have an optional expiry policy which adds one to an expiration flag on records in the stream after a defined time period has elapsed or until the expiration flag reaches a predefined maximum value. The expiry time period is relational to when the record was last updated or a time value specified in a user-defined date/time column in the record.

An expiry policy may be defined in the Properties tab of the model. The following are parameters for an expiry.

- `expiryField` names the field that is incremented each time the record is expired. If this is set to "expired", the "expired" field would take on the value 0 when the record is inserted, the value 1 after the first expiry period elapses, and the value 2 after the second.
- `expiryTime` is the amount of time in seconds of the expiry period; a "5" would mean 5 seconds.
- `expiryMaxValue` is the maximum number of times a record may expire. If this is "2", and `expiryTime` is "5", the record would expire at 5 and 10 seconds.
- `expiryTimeField` names a field in the record from which the age of the record is derived. By default this is "rowtime" a pseudo field that contains the time of the rows last update. The age of the record is defined as: current time - value of (`expiryTimeField`).

The default value for Expiry time is set to 0.

An expiry policy can be useful for a variety of tasks, including flagging records that are potentially stale (that is, having not been updated in the defined expiry interval) or identifying all records that are older than a given value. The expiry policy operates even if no events are coming in to the stream, meaning the expiry policy can be used in rules triggered by the absence of events.

A simple example would be to trigger events when a record is less than five seconds old, between 5-to-10 seconds old, or more than ten seconds old. You do it by setting an expiry policy with the `expiryTime` parameter set to "5", and the `expiryMaxValue` parameter set to "2".

A record would be generated with the expiry field set to 0 (when the record is first introduced), then if the record has not been updated after five seconds a new record with the same field values (except for an expiry field with a new value of 1) would be generated. If the record is not updated in the next five seconds, another new record with the same fields values (except for an expiry field with a new value of 2) would be generated. No matter how long the record sits in the stream without an update, no more records will be generated as the `expiryMaxValue` of "2" has been reached.

5.3. Stores

Every stream is assigned to a data store that holds the values of all retained events that are generated by the stream. The type of store used for a particular stream will partially determine the characteristics of the stream.

Store Types

- The *Memory Store* allows for data retention and all data is held in memory. There are no restrictions on the use of a Memory Store, but since all data is only held in memory, any data in the store is lost in a system failure event.
- The *log store* provides full state recovery in the event of a failure: for any stream assigned to a log store, retained data will not be lost in the event of a system failure. The log store uses a highly efficient storage mechanism to provide disk-based data persistence for all data in the store. It has been optimized for high throughput and minimal latency, but it will necessarily deliver lower performance than the Memory Store or Stateless Store.
- The *Stateless Store* does not retain or persist any of its data. It operates on a single record at a time, discarding the record once the operations on the record have been completed and the output produced. This allows a stream assigned to a Stateless Store to be extremely efficient, but it places significant restrictions on the use of

the stream:

- Data in a stateless store is not recoverable.
- A stream assigned to a Stateless Store cannot be queried.
- A stream assigned to a Stateless Store cannot be an input stream to a Join Stream.
- Only streams that are “insert-only” may be assigned to Stateless Stores. A stream is “insert-only” if it is one of the following:
 - A Union Stream, Compute Stream, or Filter Stream whose inputs are insert-only
 - A Join Stream whose “many” inputs in the many-to-one Joins are insert-only, and whose “one” inputs are “static” streams

Some notes on assigning streams to stores:

- Stateless Stores are the most efficient — they increase performance and reduce memory requirements. They should be used whenever the restrictions can be met. They are ideal for intermediate calculations.
- Multiple stores and store types can be used within a single data model.
- Multiple streams can be assigned to a single store.
- You can assign every stream that needs full-state recovery to a log store, but it's not necessary since the Sybase Aleri Streaming Platform will load re-load retained data at start-up for any stream assigned to a log store. From there, downstream nodes will be rebuilt as the retained data flows through the model. Since log stores are slower than memory stores, performance will be enhanced if you use as few log stores as possible for required state recovery. The trade-off is that recovery will take longer.
- Where multiple streams share a log store and multiple log stores are in use, there can be no cyclic conditions.

If you examine the collection of log stores in a given authored model, there is an inherent graph structure induced on the collection through the data stream flow. This induced graph must be acyclic to ensure the accuracy of the data during the recovery process.

A simple example of an illegal configuration: two log stores, LS1 and LS2, with three streams: a source stream, BS, and two Filter Streams, FS1 and FS2. If BS and FS2 are assigned to LS1, FS1 is assigned to LS2, and the data flow between streams is BS -> FS1 -> FS2, the induced graph on the log store is LS1 <-> LS2, which is a cycle. This is not allowed.

The Memory Store allows you to choose either a tree index (default) or a hash index structure. In general, the tree index provides the best performance, as it is more suited for sorted access and range queries. The hash indexes may provide better performance for equality selections. The log store only supports the tree index structure.

5.4. Joins

The Join Stream operates much the same way as a joins in relational database. Each record in a Join Stream is computed from records that “match” in one or more input streams. The type of join will determine the behavior of the join.

Before defining how to configure a Join Stream, it is helpful to recall some definitions from the terminology of relational databases. Suppose stream A and stream B are being joined. The join can be classified in two ways. The first kind of classification describes the number of possible matching rows:

- A **one-to-one** join is when a row from stream A matches at most one row from stream B, and vice versa.
- A **one-to-many** join is when a row from stream A matches possibly many rows of stream B, but each row of stream B matches at most one from stream A.
- A **many-to-one** join is the reverse of the one-to-many case.
- A **many-to-many** join is when a row from stream A may match many rows of stream B, and a row of stream B may match many rows of stream A. Many-to-many joins are *not* supported by the Sybase Aleri Streaming Platform.

The second kind of classification describes the action taken when a row from stream A has no corresponding match in stream B, or vice versa. The join may be one of the following:

- **Full-outer join** is when a row from stream A does not match any row in stream B, an output record is still generated with any values from stream B treated as “null”, and when a row from stream B does not match with any value in stream A, an output record is still generated and the values in stream A are assumed to be “null”.
- **Left-outer join** is when a row from stream A does not match with any row in stream B, the values in stream B are assumed to be “null”, but when a row in stream B does not match any value in stream A, no output row is generated.
- **Inner join** is when there is not a match and no row appears in the output.

The Sybase Aleri Streaming Platform provides all the combinations of one-to-one/many-to-one and left-outer/full-outer joins. For efficiency reasons, the Sybase Aleri Streaming Platform supports limited inner joins and does not support many-to-many joins. In most cases, inner joins are easy to simulate with a combination of full-outer and left-outer Join Streams followed by a Filter Stream that filters out null values.

Inner joins are allowed only when all the input streams are “insert-only”. A stream is “insert-only” if it is one of the following:

- a Source Stream that is “insert-only”.
- a Union Stream, Compute Stream, or Filter Stream whose inputs are insert-only.
- A Join Stream whose “many” inputs in the many-to-one joins are insert-only, and whose “one” inputs are “static” streams.

These constraints maintain the correctness of joins under updates and deletes while also preserving efficiency.

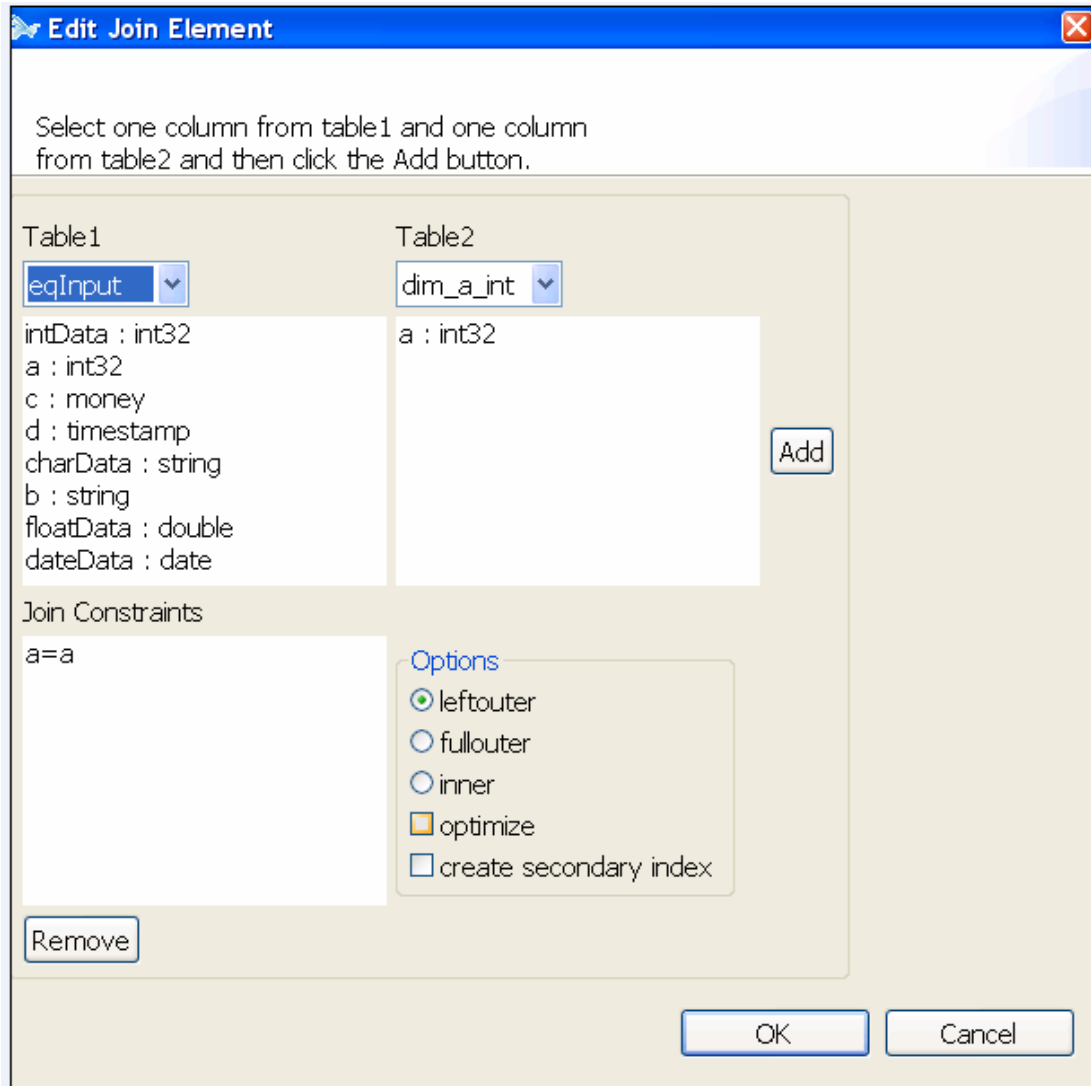
The Properties and Elements of a Join Stream

A Join Stream in a data model will have one or more Join elements. The join element will specify which fields to use to match records across the input streams. There will be a join element for each pair of input streams. Therefore, for a simple Join with only two input streams, there will be a single join element that relates records from input stream 1 to records from input stream 2. For a join with 3 inputs, two join elements will be required: the first relates input stream 1 to input stream 2, the second relates input stream 3 to either input stream 1 or 2. An additional join element is needed for each additional input stream.

A join element has the following properties:

- table1 stream ID (this is the "left" table in the join)
- table2 stream ID (this is the "right" table in the join)
- join type: left-outer, full-outer, inner (see above)
- optimize: reserved for future use, and currently ignored
- secondary: `true` if the Sybase Aleri Streaming Platform should construct an index for table1 to speed up the join, and `false` otherwise; setting this option to `true` uses more memory

If you want to edit a join element of a Join Stream, double-click the Join element to bring up the following **Edit Join Element** dialog box so you can make the appropriate changes.



5.5. Distributed Models and Clustering

A data model can grow so large or complex that it becomes necessary to break it into modules so that each module can be run on a separate machine. The fragmentation of a complex model into modules lets it utilize more processing power and memory.

Modules are added to the model by selecting the Module tool from the Distributed Elements section of the **Palette** and then placing the module on the diagram. A module can also be created by using the outline view Create New Child->Distributed Element->Module.

Streams can be assigned to module in two different ways:

- Select the stream and use the context menu item Modify->Change Parent. The Change Parent dialog will allow you to choose an existing module or create a new module to assign the stream.
- In the outline view, select the stream to move and drag the stream into the parent module.

5.5.1. Modify a Model to Run in a Cluster Configuration

Using Visual Authoring

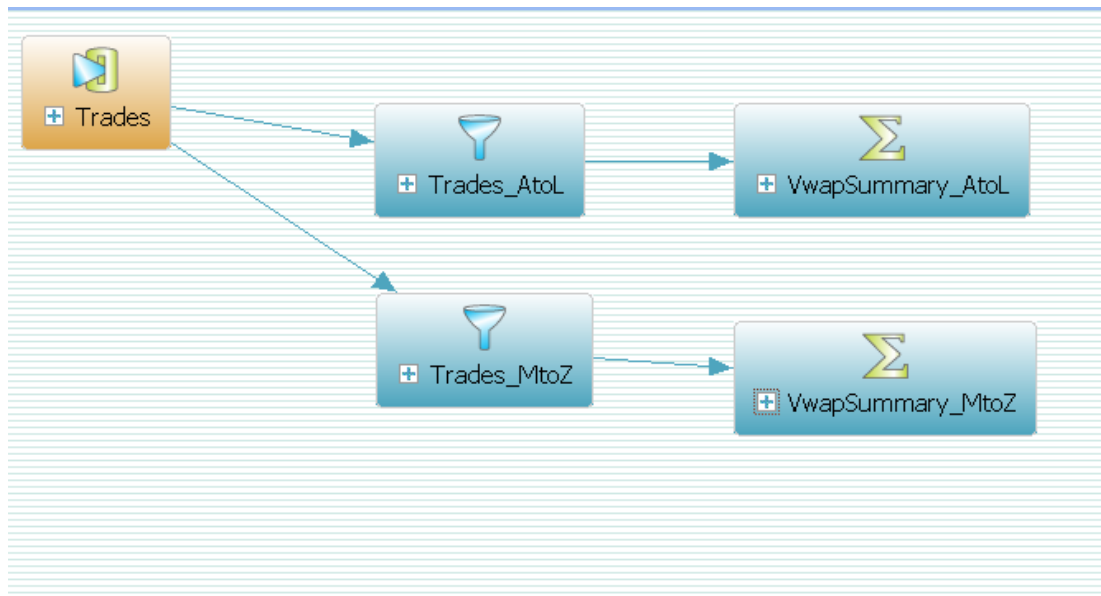
The first step in building a model for a clustered hardware configuration is creating a data model and displaying it in a diagram in the Aleri Studio. Next, decide which parts of the model should be grouped into modules.

Modules are added to the model by selecting the Module tool from the Distributed Elements section of the **Palette** and then placing the module on the diagram. A module can also be created by using the outline view Create New Child->Distributed Element->Module.

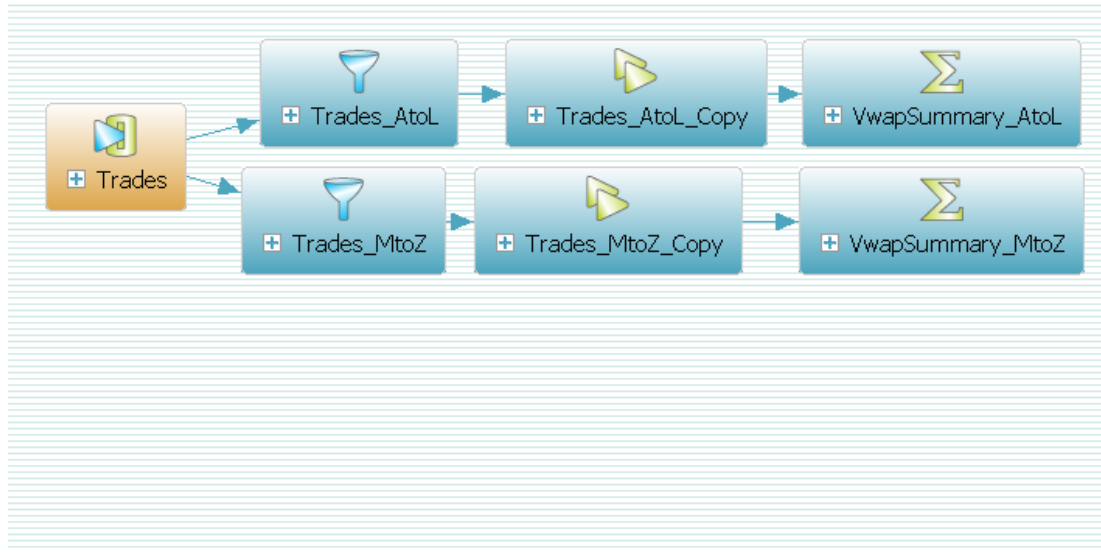
Streams can be assigned to a module in two ways:

- Select the stream and use the context menu item Modify->Change Parent. The Change Parent dialog will allow you to either choose an existing module or create a new module to assign the stream.
- In the outline view, select the stream to move and drag the stream into the parent module.

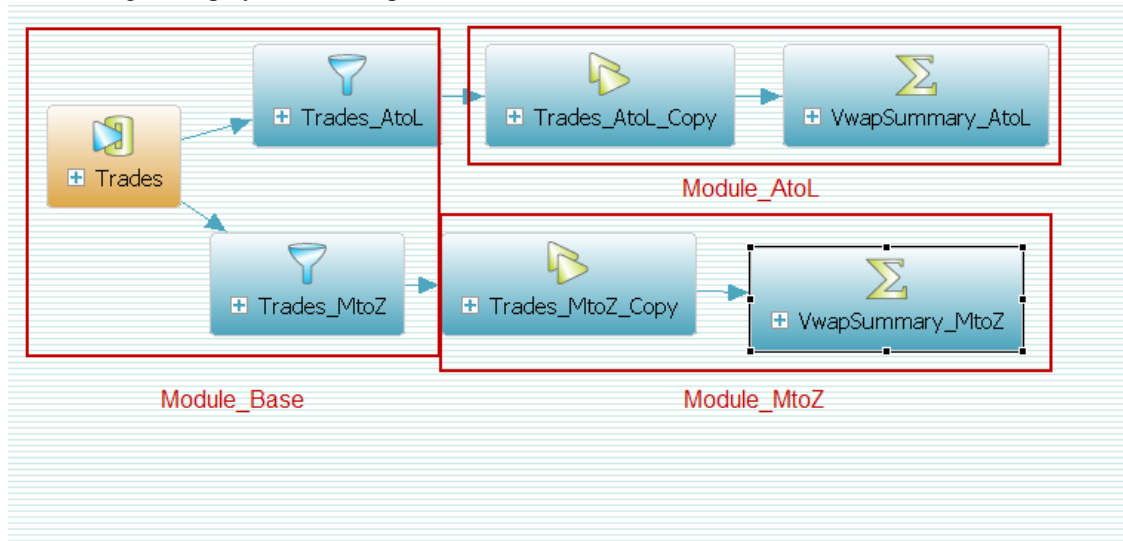
The following figures show how a model can be divided so that the first module consists of the **Trades**, **Trades_AtoL** and **Trades_MtoZ** streams. The second module has the **VwapSummary_AtoL** stream. And the third module contains the **VwapSummary_MtoZ** stream. It's recommended that the number of edges leaving a given module and entering another module be minimized, since these edges represent data flow between servers.



To make clustering work, you must add a Copy Stream as the receiving stream at each point where the data enters one module from another. In this example, two new streams are introduced, with the first named **Trades_AtoL_Copy** and the second **Trades_MtoZ_Copy**. They have the streams **Trades_AtoL** and **Trades_MtoZ**, respectively, on their input streams.



The next figure displays the final representation of this distributed model.



Using Aleri SQL

See the *Authoring Reference* for a complete definition of the Aleri SQL syntax.

Using AleriML

To declare a module in an AleriML configuration file, use the <Module> tag.

```
<Module id="submodelname">
  Store declarations
  Stream declarations
</Module>
```

Note:

Streams defined in a module can only access stores defined within the same module. There

must be at least one store per module.

For the above example model above, it could have:

```
<Module id="module1">
  <Store id="store1" kind="memory">
    <SourceStream id="Trade" store="store1" ....>
    <AggregateStream id="AveragePrices" store="store1" ...>
  </Module>

<Module id="module2">
  <Store id="store2" kind="memory">
    <CopyStream id="CopyAveragePrices" store="store2" ....>
    <SourceStream id="Book" store="store2" ...>
    <JoinStream id="IndividualPositions" store="store2" ...>
    <AggregateStream id="BookPositions" store="store2" ...>
  </Module>
```

On the other hand, rules, parameters and so forth referenced by streams may be declared anywhere in the Sybase Aleri Streaming Platform configuration file. They can be declared within the module they are referenced in, or in another module, or outside all modules. The only restriction is that streams in a module definition can only refer to stores in the same module definition.

5.5.2. Mapping Modules to Servers

A distributed model can be run in various cluster configurations. There may be instances when ServerA and ServerB are used, instances when ServerB and ServerC are used, or instances when all modules may run on ServerA or ServerB. The Sybase Aleri Streaming Platform supports this flexibility with ease. Module to server mapping is done in the <Cluster> sections of the configuration file. It is suggested that you indicate the clusters should be specified as the first elements in the <Platform> container.

For example:

```
<Platform>

<Cluster id="serverSet1">
  <Node module="module1" machine="servA.sybase.com"
  commandport="31415">
  <Node module="module2" machine="servB.sybase.com"
  commandport="31415">
</Cluster>

<Cluster id="serverSet2">
  <Node module="module1" machine="servB.sybase.com"
  commandport="31415">Configuring
  <Node module="module2" machine="servC.sybase.com"
  commandport="31415">
</Cluster>

<Cluster id="serverSet3">
  <Node module="module1" machine="servA.sybase.com"
  commandport="31415">
  <Node module="module2" machine="servA.sybase.com"
  commandport="31416">
</Cluster>
```

```

<Cluster id="serverSet4">
  <Node module="module1" machine="servB.sybase.com"
  commandport="31415">
    <Node module="module2" machine="servB.sybase.com"
    commandport="31416">
  </Cluster>

<Module ...>
...
<Module ...>

</Platform>

```

This data model can be started in two different ways:

- If you start the model in a non-clustered configuration, pass the configuration file to the Sybase Aleri Streaming Platform without any cluster or module information on the command line. The Sybase Aleri Streaming Platform will ignore the module and cluster tags and run this as a non-distributed model.
- You must pass the cluster name (-C option to sp) and module name (-M option to sp) to the Sybase Aleri Streaming Platform when starting a cluster.

When running a cluster, the proper authentication command line parameters depend on which authentication method will be used between nodes in the cluster. The following table lists the options to be used with each method.

None	-V none
PAM	-V pam -a <username> <password>
RSA	-V rsa -K <username> <private key file>
GSSAPI (Kerberos® V5)	-V gssapi -G <username>

The Sybase Aleri Streaming Platform will start the specified module on the current host. It checks that the module name and current host are consistent with what is specified in the cluster chosen. The above sample defines four different cluster configurations; this model can be executed according to any of those definitions.

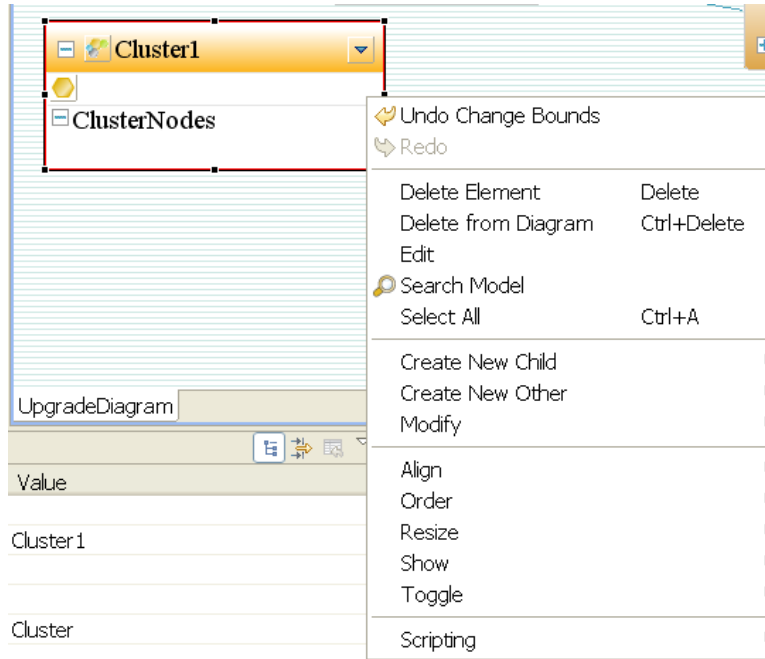
For example:

- To start `servA`, pass the configuration file to the Sybase Aleri Streaming Platform and use `-C serverSet1 -M module1`.
- To start `servB`, pass the configuration file to the Sybase Aleri Streaming Platform and use `-C serverSet1 -M module2`.

The two modules will run independently on the individual nodes and forward the required data between the servers to maintain consistency across the entire model.

5.5.3. Create a Cluster

Clusters are used in conjunction with Modules to run a distributed model. Refer to [Section 5.5, “Distributed Models and Clustering”](#) for information about when to use clusters.

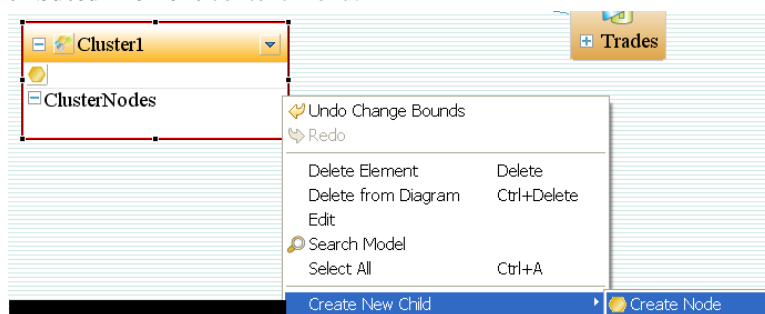


5.5.3.1. Create a Cluster using the Outline view:

1. From the Outline view, right-click **Platform Container**, move the mouse cursor over **Create Distributed Element** and click **Cluster** in the submenu.
2. In the Properties area, edit the ID of the Cluster in the Id property.
3. Update any of the remaining properties as necessary.

5.5.3.2. Create a Node for a Cluster

A Node can be created as a child element of a Cluster from the Outline view or from the **Create Distributed Element** context menu.



5.5.3.3. Create a Node using the Outline view:

1. Right-click on the name of a Cluster in the Outline view.
2. In the context menu, move the mouse cursor over **Create New**, and click **Create Node** from the submenu.
3. In the Properties area, edit the ID of the Cluster in the *Id* property.

4. Edit the following properties (see [Section 5.5.2, “Mapping Modules to Servers”](#)):
 - commandport
 - machine
 - module (this is a selection box)

5.6. Creating a Persistent Subscribe Pattern

A “persistent subscribe pattern” is a set of derived streams that can be created automatically to attach a client application with a “persistent subscription”. A persistent subscription provides a way of ensuring that a client receives all transactions even if the client is unreachable for a period of time. When the client reconnects, it can receive all transactions that occurred while the connection was down.

To create a new persistent subscribe pattern for an existing stream, right-click a stream shape on the diagram and then select **Create persistent subscribe pattern** from the context menu. The following new elements will be created in the model:

- A new FlexStream is created with an id equal to *InputStream.id+_log*.
- A new Source Stream is created with an id equal to *InputStream.id+_truncate*
- Two methods are created in the FlexStream, one called `logMethod` and the other called `truncateMethod`. These methods are derived from code templates.

Pick an existing store or create a new one of type log store as the store for the newly created Flex and Source Streams to complete the process.

There is a current limitation for the FlexStream created as part of the persistent subscribe pattern. You must manually modify the column datatypes of the FlexStream to correspond to the data types of the stream that were the target of the pattern.

Chapter 6. Running and Testing a Data Model

6.1. The Run-Test Perspective

The Run-Test Perspective provides a useful environment to test, debug and examine a data model. It can be used with all three authoring environments.

You can also use the perspective to control the Sybase Aleri Streaming Platform whether it is run locally on your computer or remotely on a server machine. If you use it on a server, you must check the **USE SSH for Studio Execution** check box in the Execution Defaults section of the Preferences dialog box.

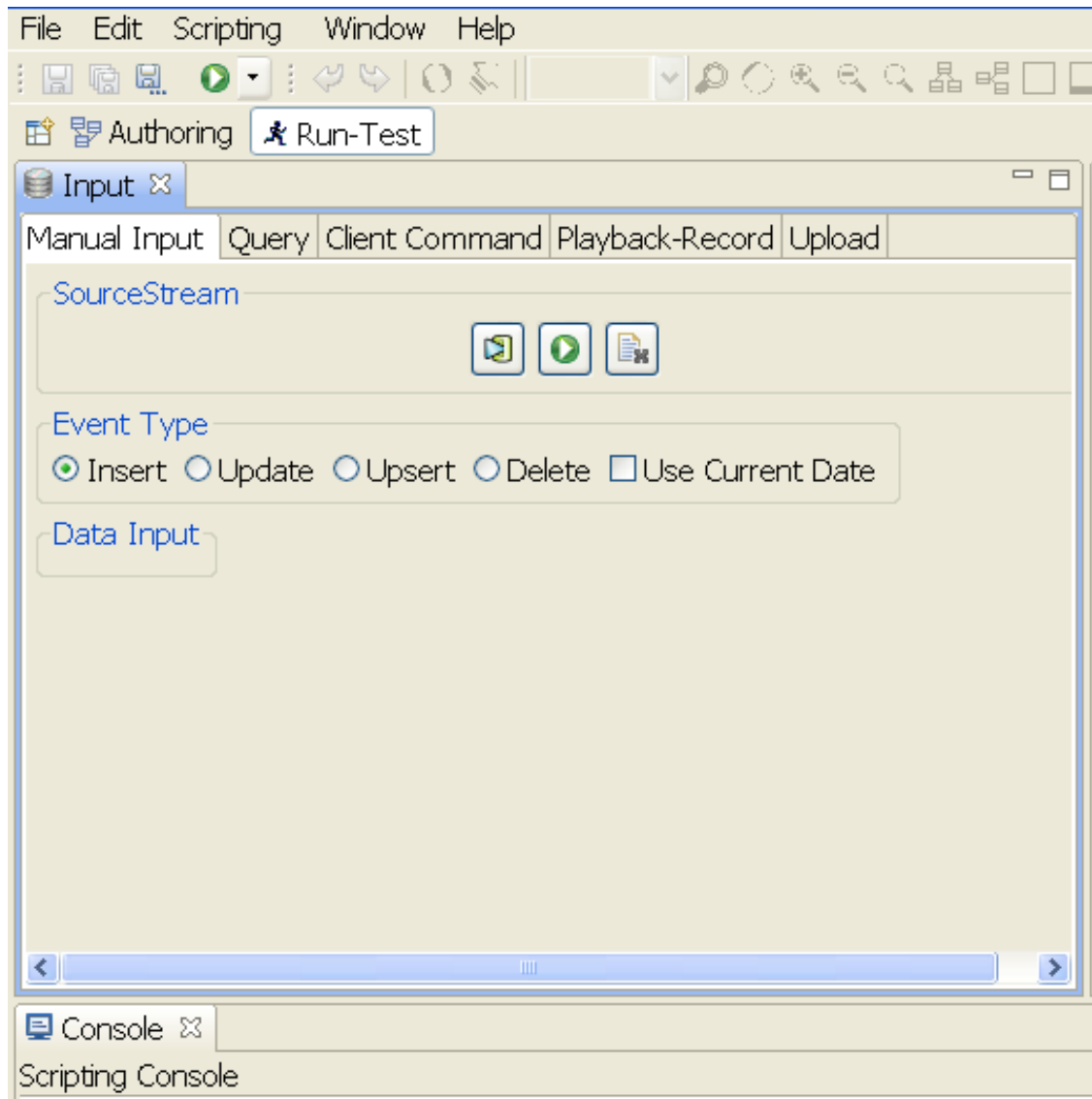
- From the **Window** menu, click **Open Perspective**. Select *Run-Test* and click **OK**

-or-

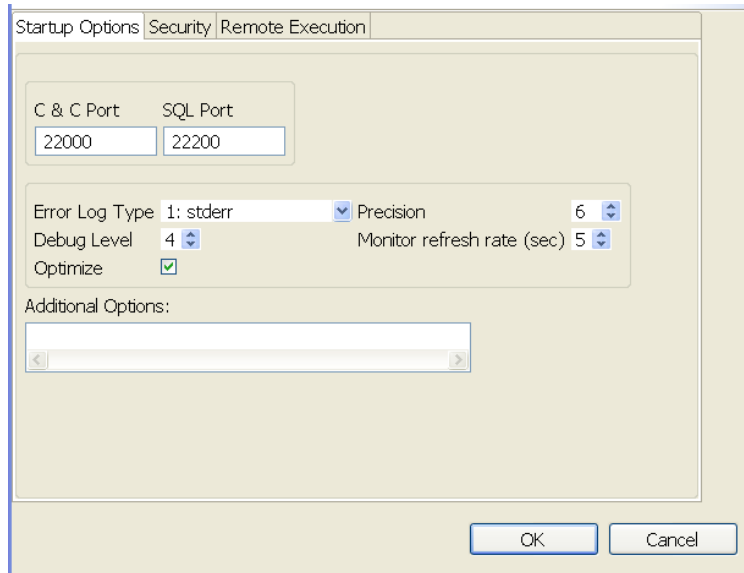
Click the **Run-Test** tab at the top of the Aleri Studio main window.

The Run-Test Perspective consists of the following views:

- **Monitor** allows you to monitor performance and identify bottlenecks.
- **Streamviewer** lets you see the contents/output of any streams.
- **Breakpoints** lets you add breakpoints and watch items to the model. It also provides the ability to step through the model.
- **Debugger** shows a directed graph of the streams in a model. It lets you examine the events (insert, update, delete) flowing through a model.
- **Input** lets you manually enter a single event in a running model.



6.1.1. Configure the Run-Test Perspective



The Configuration tab of the Run-Test perspective contains the following fields:

- **Username, Password** is where to enter the username and password for the account under which the Stream Processor, **sp_server**, will run.
- **C&C port** is the Command and Control port for the server. It defaults to 22000; normally there is no need to change it unless you are connecting to a server that has been configured to use a different C&C port.
- **SQL port** is the port on the server used for ad-hoc SQL queries (on demand queries). It defaults to 22200. The only reason to change it is if you are connecting to a server that has been configured to use a different SQL port.
- **Encryption settings** are for unencrypted operation. Simply leave these fields blank. For encrypted operation, check the **Encryption** box and specify the directory that contains the SSL keys and certificates. The default value is `.../etc/keys`. This is equivalent to using the `-e` option on the **sp_server** tool. Refer to the *Utilities Guide*.
- **Monitoring Settings:** in order for the Tools in the Run-Test Perspective (Monitor, Streamviewer, Debugger) to work correctly, the server needs to be able to send data to the Aleri Studio. The Studio Port defaults to 22500; the only reason to change it is if the Aleri Studio configuration has been changed. You can leave the value in the **Host** field as `localhost` if the Aleri Studio and the Sybase Aleri Streaming Platform are running on the same host. If they are running on different hosts (remote execution), enter the name of the host on which the Aleri Studio is running.

6.1.2. Start

The **Start** tab lets you start the Stream Processor either locally or on a remote server (depending on the settings in the Configuration tab).

To start the Sybase Aleri Streaming Platform from this tab panel:

1. Ensure that all the settings on the Configuration tab have been set properly. If a drive alias is used, make sure that the alias has been set on the **Path Aliases** tab.
2. Check the filename in the **File** field. It will already be filled out with the name of the file that is

open in the Aleri Studio. You can start the Stream Processor with a different configuration file by changing the filename here.

3. Select one of the following values from the dropdown list in the Error Log Type field to control where log messages are sent:
 - 0 No log messages
 - 1 Send to stderr only
 - 2 Send to syslog only
 - 3 Send to both stderr and syslog
4. Select the number of decimal places in streaming output from the dropdown list in the Precision field. The default value is 6.
5. Select the debug level from the dropdown list for the Debug Level field. The valid range is 0 - 7, with 7 being verbose. The default value is 0.
6. Select a value from a dropdown list in the **Monitor Duration** field if there is a requirement to turn on Monitoring statistics. The requirement to turn on performance monitoring statistics. The statistics will be updated every *n* seconds where *n* is the value specified in this field. This is equivalent to the `-t` option passed in when executing `sp_server` directly. Refer to the *Utilities Guide* . for more information.
7. Add **Additional Options** as needed. This is a catch-all provided so that additional command-line arguments can be passed to `sp_server`. The default value is an empty string. Here is the list of additional options.
 - `-B path`: Set the name of the file to which all rejected records will be written.
 - `-F path`: Set the XML Schema file (default is `$PLATFORM_HOME/etc/Platform.xsd`).
 - `-k path`: Set the path to a directory of RSA public keys for RSA authentication.
 - `-r true|false`: Turn on/off access control (default is `false` for off).
 - `-s` Run as a daemon. This option forces "-l 2" (to send log messages to syslog only).
8. Click **Start Platform**.

The Sybase Aleri Streaming Platform starts the Stream Processor.

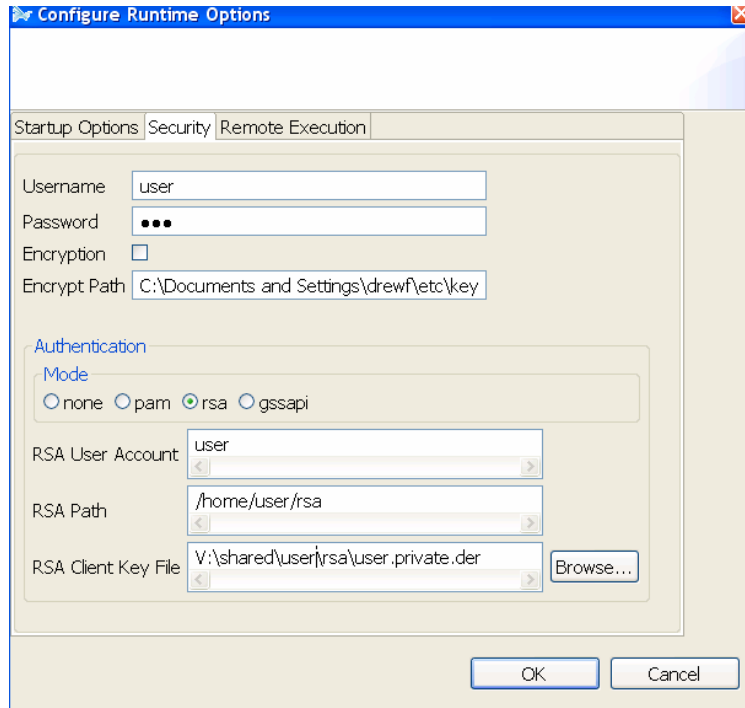
The Console area displays the output from the Stream Processor (the `sp_serverprocess`).


Note:

When you start the Sybase Aleri Streaming Platform under some conditions — in particular, if you checked the **Encryption** check box on the **Configuration** tab panel — the Sybase Aleri Streaming Platform may take longer to start up completely. This may occur even after the **Start Platform** button has responded. To make sure the Sybase Aleri Streaming Platform's startup process is complete, wait until you see the Platform Server alive & ready for services message in the **Console** tab panel before you go on to other activities.

To shut down the Stream Processor, click **Stop Platform**.

6.1.3. RSA/PAM Settings



You have four options for authentication security with the Sybase Aleri Streaming Platform: RSA, PAM (Pluggable Authentication Model), Kerberos V and none. You must be in the Authoring perspective to get the Security tab panel on the top toolbar. Next, click on the downward arrowhead on the top toolbar to get  for **Startup Options** and then select **Security**. If you want to use PAM, make that choice in the Security Tab. You must fill the following fields in the tab panel if you want to use RSA:

- **RSA User Account** is the RSA username for authentication. This is the username passed to the Sybase Aleri Streaming Platform to create Pub/Sub connections. It does not have to be the same username as the name on the Configuration tab. (The Configuration tab username/password combination is used for SSH execution and PAM authentication).
- **RSA Path** is the directory on the server used as the `-k` argument passed to the Sybase Aleri Streaming Platform on startup.
- **RSA Client Key File** is the fully qualified file path on the Aleri Studio machine to the client key file that is created in the generation steps listed below. This file is used for the Pub/Sub connection from the Aleri Studio to the Sybase Aleri Streaming Platform.

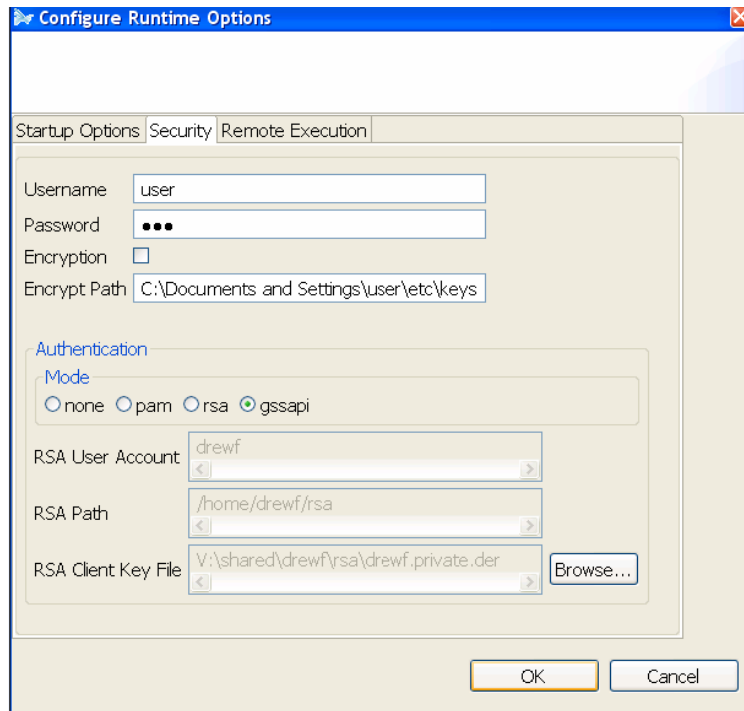
Steps to create the RSA Client Key file:


```

.....
bin/openssl genrsa 2048 > theUserName.private
bin/openssl rsa -inform PEM -outform PEM -pubout
< theUserName.private >\
theUserName
bin/openssl pkcs8 -in theUserName.private -nocrypt -topk8
-outform DER -out \
theUserName.private.der

```

6.1.4. Kerberos/GSSAPI Authentication



You can use Kerberos for network authentication security with the Sybase Aleri Streaming Platform. You must be in the Authoring perspective to get the Security tab panel in the top toolbar. Next, click on the downward arrowhead on the top toolbar to get  for **Startup Options**, and then you would select **Security**. Next, choose **gssapi** as your option in the Security tab panel.

The default `login.config` file for GSSAPI is supplied in the `$PLATFORM_HOME/etc` directory. If you choose authentication with GSSAPI, the default settings are:

- GSSAPI Application Name is the application name specified in the `login.config` file. This file is used during the startup of the Aleri Studio, and the default GSSAPI name is **SpStudio**.
- GSSAPI User is the username utilized by the various command-line tools run from the Aleri Studio, such as **sp_server**, **sp_cli**, **sp_convert**, and **sp_upload**.
- GSSAPI Password is utilized by the various command-line tools run from the Aleri Studio, such as **sp_server**, **sp_cli**, **sp_convert**, and **sp_upload**.

If you want to use a different configuration for GSSAPI authentication, you can replace the `login.config` file supplied in the `$PLATFORM_HOME/etc` directory. You can also modify the `sp_studio.bat` for Win32 or `sp_studio` shell script for UNIX and change the Java JVM argument `-Djava.security.auth.login.config=%PLATFORM_HOME%\etc\login.config` to point to a different path for the `login.config` file.

6.1.5. Kerberos User IDs and Role-Based Authorization

Kerberos authentication has two forms for user identification:

- **userid** for when the client location is in the same realm as where the server is running.

- **userid@REALM** for when the client location is in a different realm than where the server is running.

In both cases, only the **userid** portion of the login credential is used in group lookups when performing role-based authorizations. Therefore role-based authorization can be exclusively driven by user IDs.

6.1.6. Path Aliases

This tab panel gives you the option to create an alias for a file location on another machine. The best way to do this is to map a network drive to the directory on the server that contains the Sybase Aleri Streaming Platform. You can then set an alias in the Aleri Studio to substitute the mapped drive letter for the Sybase Aleri Streaming Platform path.

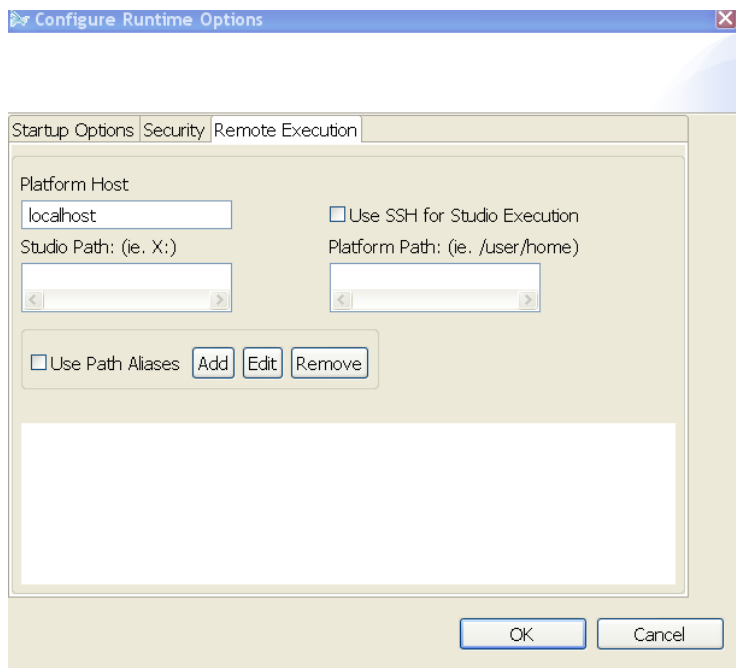
Note

To map a network drive, the UNIX server must be running Samba or equivalent to allow the UNIX server file-system to be seen from Win32.

Example:

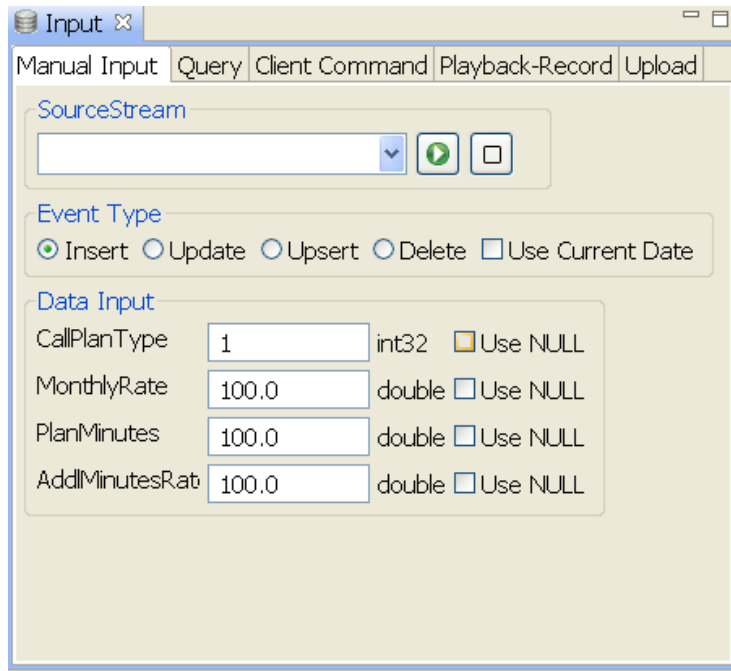
- UNIX Server Name = myserver
- Win32 file open = x:\shared\user\vwap\vwap.xml
- Actual Location = /opt/shared/user/vwap/vwap.xml (on myserver)
- Mapped drive on Win32 = x: (myserver:/opt)
- Path alias = x: ==> /opt

As a result, whenever the execution framework encounters x:, it is replaced with /opt. This occurs when Start Platform calls **sp_server**, or when data files are used via the **Upload** panel.



6.1.7. Data Input view

Data Input allows you to manually enter a single event into the running Sybase Aleri Streaming Platform.



Data Input is available:

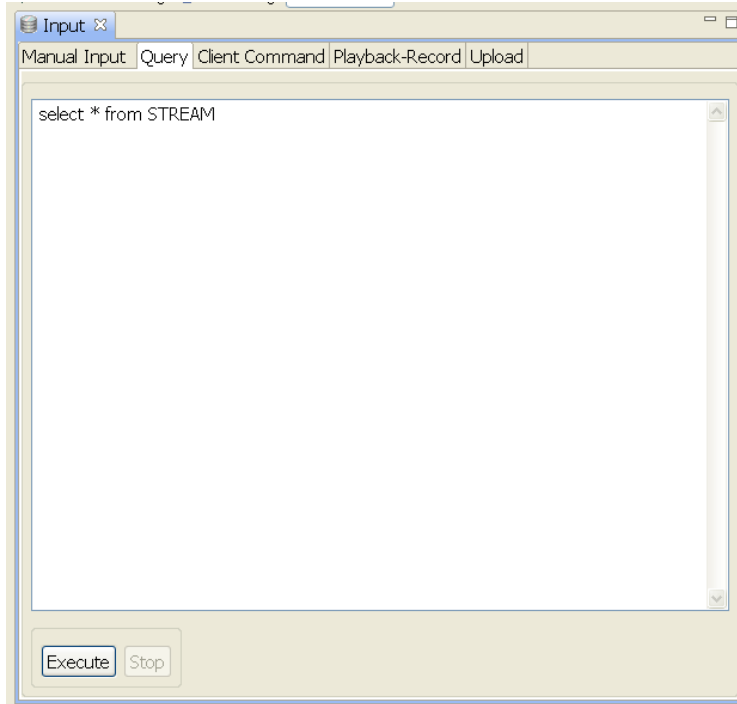
- As the last tab to the right of the Debugger when the Execution Scenario Perspective is active.

To use the Data Input view, select a source stream from the **source stream** combo box.

- Use the **EventType** radio buttons to indicate the type of event to be sent to the Sybase Aleri Streaming Platform.
- Click the **Publish Data** button (the white arrow inside the green circle in the upper right corner of the **Data Input View** tab panel) to publish data to the running Sybase Aleri Streaming Platform.
- Click the **Reset Input** button (to the right of the **Publish Data** button) to reset the values in the View to the default values.

6.1.8. Run an Ad-hoc SQL Query

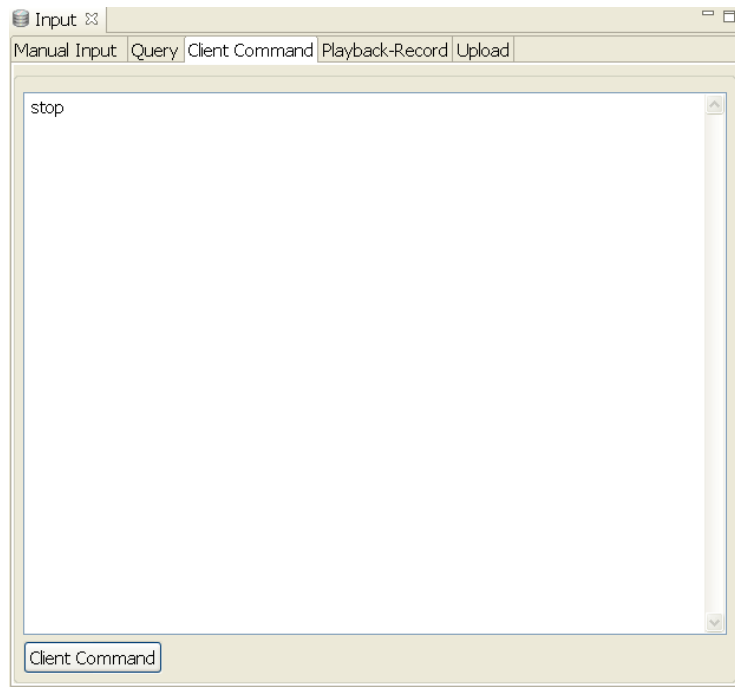
You can execute ad-hoc SQL queries against the stream processor from the **Query** tab panel. Simply enter an SQL SELECT statement to query data from a specific stream (source or derived). The stream you are querying must have retained data to be queryable (streams assigned to stateless stores are not queryable).



1. Click the **Query** tab in the Command Execution area.
2. In the Query area, type the SQL query to send to the Sybase Aleri Streaming Platform. The query should be of the form `SELECT <expression> FROM <stream id>`.
3. Click **Execute**. The Sybase Aleri Streaming Platform uses the `sp_query` utility to execute the SQL query. The Console area displays the status of the query.
4. If necessary, you can click **Stop** to stop a long-running query.

6.1.9. Send Commands to the Stream Processor

You can send commands to the Stream Processor via the Command and Control interface by typing them into the **Client Command** tab panel.

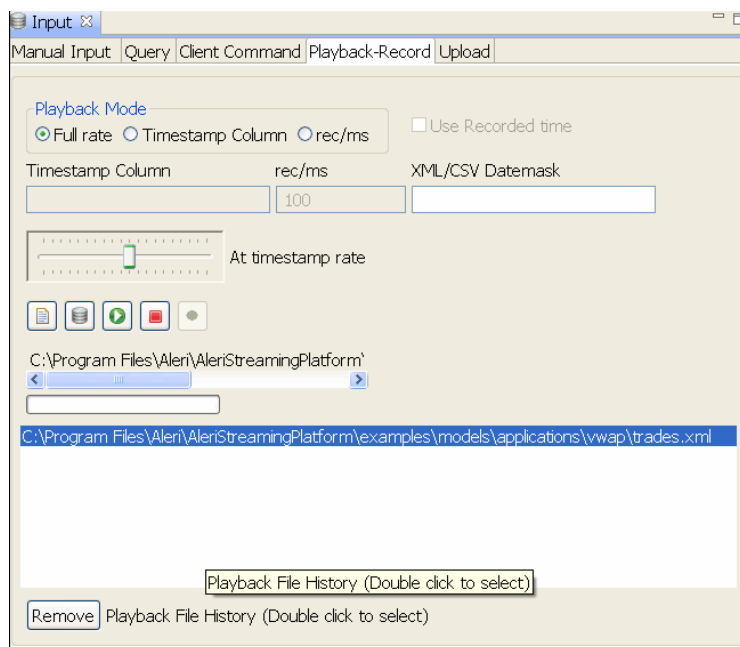


To do this:

1. Click the Client Command tab in the Platform Command Execution area.
2. Enter the command and arguments in the Client Command area.
3. Click **Client** Command. The Sybase Aleri Streaming Platform invokes the **sp_cli** utility to execute the command.

6.1.10. Aleri Studio Playback

In the Aleri Studio Playback view, you can record in-flowing data to a playback file and play the captured data back into a running Sybase Aleri Streaming Platform instance.



The following buttons are available in the Aleri Studio Playback view:

Start Playback	Starts playing the current playback file.
Stop	Stops playback or record, closes the associated file and closes the associated playback or record context.
Record	Prompts the user to select the name of the file in which to store recorded data, then starts the Aleri Studio data recorder. Progress is updated in the status bar.
Adjust Playback	This slider is used during playback to vary the rate. The range is from -10 times the playback file timestamp to +10 times the playback timestamp rate.
Select Playback File	You can choose the file format to use with the Aleri Studio Recorder. See below for more information on format choices.
Select Playback DataSource	It lets you use the Playback feature with data sources. See below for more information about data source options.

The Sybase Aleri Streaming Platform has three Playback modes:

Full Rate	Full Rate means the speed is not imposed by Playback, rather it is dependent on factors such as the computer which is running Aleri Studio or the network's latency.
Timestamp	The Timestamp column mode is when Playback picks up the timing rate information from a specified column. You must fill out the Timestamp column when selecting this option. During playback, timestamps are used to determine the time intervals between records.

If you select **Checkbox Use Recorded Time**, the Sybase Aleri

Streaming Platform assumes playback runs as if it is the time when the data was recorded. Otherwise the current time is used so records are played back as if produced now.

You should decide whether to choose the **Checkbox** option based on the logic used for your model. For example, if your model's logic refers to the current time using the functions `sysdate()` or `sys-timestamp()` and expects the incoming data to match it, then it would purge the previous day's information. The model would get the current time with `sysdate()` and then go through the records and delete those containing a value within 24 hours from the current time. You would want to use the **Checkbox** option since data recorded during the previous day would be deleted right away if the option is not selected.

If you want to play back the data from one file and then from another file that was recorded at a different time, you should not use the **Checkbox** option since the Sybase Aleri Streaming Platform's notion of time would jump suddenly when the second file starts playing.

records-per-millisecond

The records-per-millisecond (rec/ms) mode lets playback occur at a records per millisecond rate. It allows you to set an initial rec/ms rate that can then be changed using the slider tool on the Playback panel.

There is a Playback File History feature that lets you add or delete items in the history.

- If you want to add an item to the Playback File History list, click on the button for **Select Playback file**. Each time a file is selected for playback, it is added as an entry in the history list.
- In order to select a file for playback from the history, you have to double-click the history entry to activate it.
- You can delete an item with the **Delete Key and Remove** button. Modifications to the playback history are permanent.

You can choose one of the following file formats for the Aleri Studio Recorder:

- AleriML Format (.xml extension)
- Comma Separated Value files (.csv extension)
- Aleri Binary file (.bin extension)
- Aleri Recorded File (.rec extension)

You can use the Playback feature with the following data sources:

- Sybase ASE®
- ODBC®

- Teradata®
- kdb+
- Netezza®

Drivers for Sybase ASE, Netezza and Teradata are not distributed with the Sybase Aleri Streaming Platform. You must obtain the driver from the vendor. For more information on how to configure these databases with the Sybase Aleri Streaming Platform, refer to the *Administrator's Guide*.

kdb+ does not work with the Solaris® x86 operating system.

After selecting which data source you want to use, you'll need to configure additional parameters for your choice. The parameters are displayed in the **Configure DataSource** Dialog that is activated when pressing the **Select Playback Datasource** button.

Note

Data saved in binary files, created either by the recorder or the sp convert command, are sensitive to the data model. If you alter the model, such as by adding streams or changing advanced options like the precision of the "money" data type, the data may no longer work.

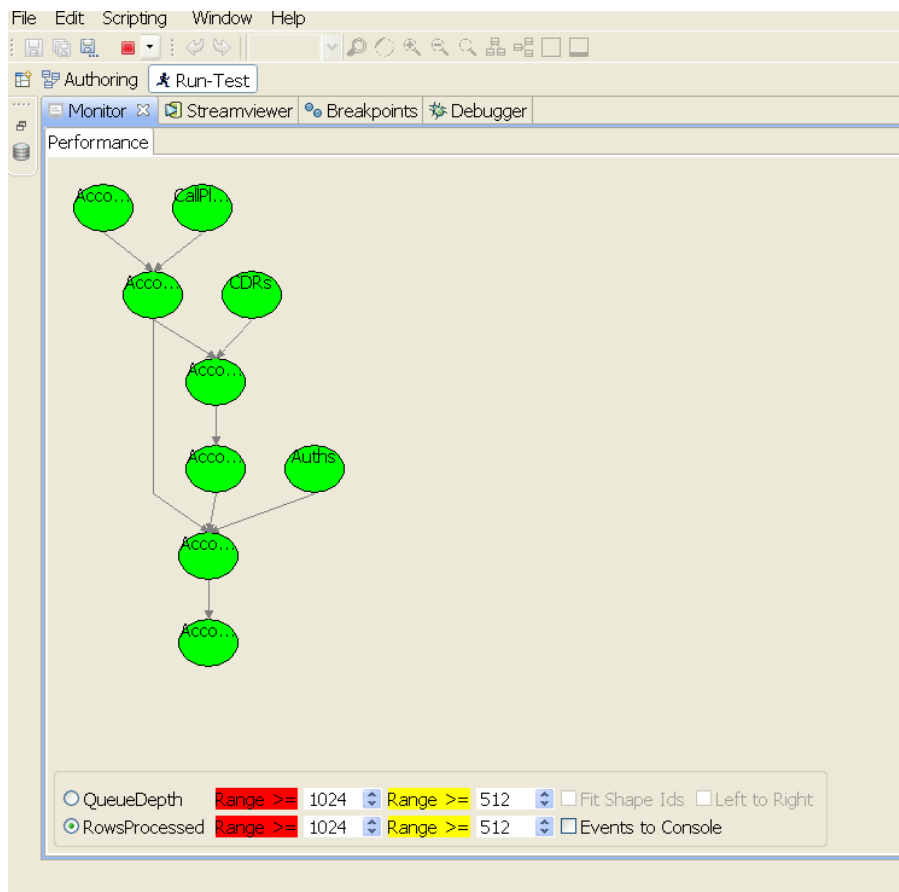
6.1.11. Upload Test Data

The Upload tab can be used to upload test data stored in files and move it into the stream processor. Follow this procedure to upload test data from this Tab Panel.

1. Make sure the Stream Processor is already running.
2. Click **Browse** and select a data file to be uploaded in the Input File field. More than one input file can be loaded, but no files will actually be uploaded until you click **Upload**. To remove a file from the Input File field, select the file and click **Remove**.
3. Check the **Convert** check box if the input data file contains data in either XML or delimited text format. Uncheck this box if the file contains data in the Aleri binary message format.
4. If the data file is in delimited text format, check the **Convert** box and then enter the Field Delimiter.
5. Optionally, select a buffer size from the dropdown list in the Record Buffer field. This corresponds to the -t option for sp_upload. Refer to the *Utilities Guide* for more information.
6. Select one or more files in the Input Files list and click Upload. The data in these files will move into the Stream Processor.

6.2. Monitor

The monitoring tool allows runtime visualization of the data model with visual indicators of throughput, queuing and CPU utilization. This tool is very useful for identifying bottlenecks in the data model.



Each node represents a stream in the data model with the lines delineating the data flow paths. The colors of each node indicate either *QueueDepth* or *Throughput* (Rows Processed per Second).

For example, if you select the **Color Queue Depth** option, the (Red) Range >= field defaults to 125, and the (Yellow) Range >= field defaults to 20. This means:

- If the queue depth of the stream node is greater than or equal to 125, the node is colored red.
- If the queue depth of the stream node is between 20 and 124, the node is colored yellow.
- If the queue depth of the stream node is less than 20, the node is colored green.

In addition to the specified colors, the Sybase Aleri Streaming Platform depicts CPU utilization as a black pie wedge in the ellipse of the node. The remainder of the ellipse is red, yellow or green, based on the options chosen above.

You can also take a quick look at a node's performance statistics by moving the mouse cursor over the node in the diagram. When you do this, the performance numbers appear in a tooltip.

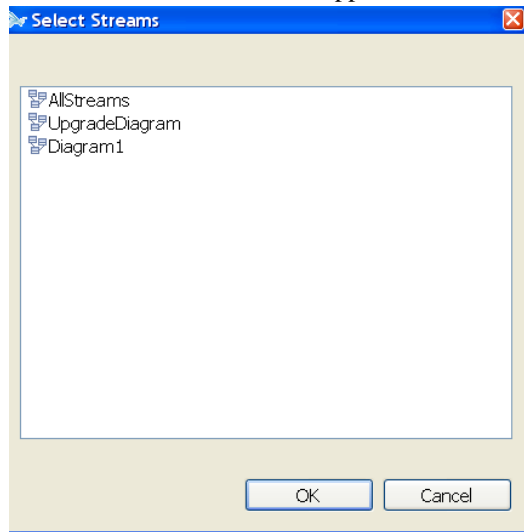
6.2.1. Run the Monitor

To do this:

1. Ensure that the Stream Processor is running.

2. Click on the **Monitoring** tab in the **Monitoring** area.
3. Click on the **Select Streams** button.

The **Select Streams** window appears.



4. Select the diagram with the streams whose performance is to be monitored, or, select **All Streams** to monitor the performance of all streams in the model.
5. Click **OK**.
6. Select **QueueDepth** or **RowsProcessed** option to determine how to color each node in the performance diagram. For either option:
 - Use the up and down arrow buttons in the (Red) Range > field to select the range resulting in a red node.
 - Use the up and down arrow buttons in the (Yellow) Range >= field to select the range resulting in a yellow node.

Note:

Nodes are green when they fall within the range that is not in either the (Red) Range >= or (Yellow) Range >= fields.

7. Click the **Start Monitoring**. The Sybase Aleri Streaming Platform colors the nodes in the diagram accordingly. The diagram updates continuously.

Note

If the nodes remain white, it's an indication that the monitor is not receiving data from the Stream Processor. Check that the Local Host information in the **Configuration** tab has been set properly.

8. If necessary, click the **Zoom In** button or **Zoom Out** button to see a larger or smaller view of the diagram.
9. Click the **Stop Monitoring** button to stop the monitoring.

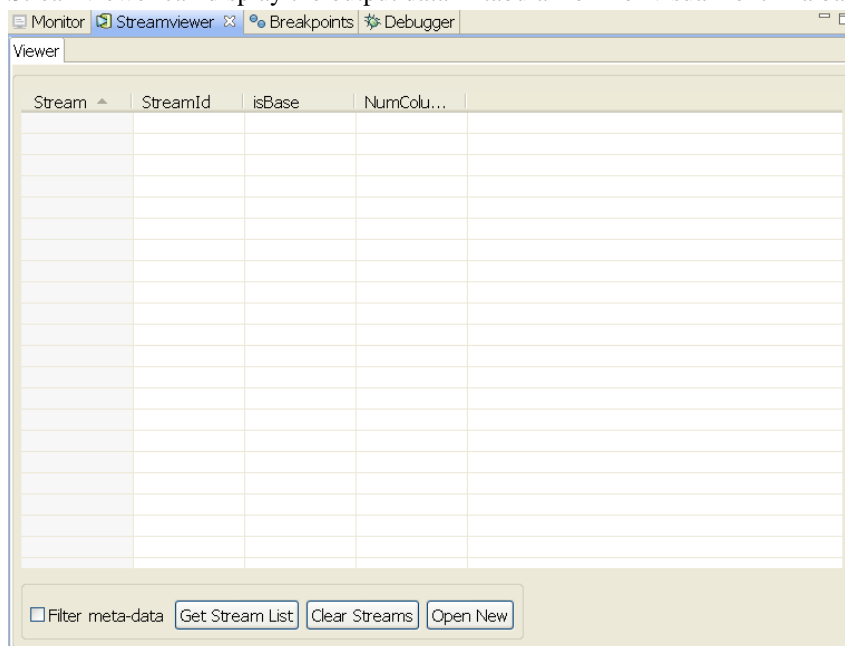
6.2.2. Save the contents of a performance diagram as an image

To save a JPG version of current performance diagram,

1. Click the **Monitor** tab. the Platform monitoring area.
2. In the **Monitor** tab, click the **Save** button. The **Save Diagram as JPG File** window appears.
3. In the **Save Diagram as JPG File window**:
 - Type the fully qualified name for the file in the Enter File Name field.
 - Click **Save**.

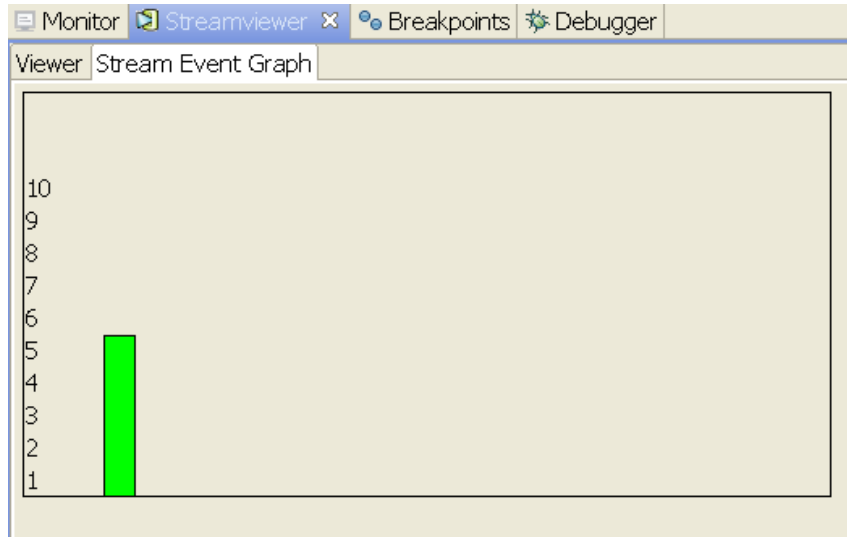
6.3. Streamviewer

The Streamviewer allows you to see and monitor the output of any stream in the data model. The Streamviewer can display the output data in tabular form or visualize it in a bar graph.



The following figure displays a bar graph of selected input streams. A key to the bar graph:

- Each stacked bar is dynamically linked with the insert, update, and delete events of a stream.
- The green stack represents the insert events.
- The blue stack displays the updates.
- The red stack represents the delete events.
- The height of the bar represents the total number of events that are processed for the stream.



To use Streamviewer:

1. Ensure that the Stream Processor is running.
2. Click the Sybase Aleri Streaming Platform Streamviewer tab in the monitoring area.
3. Click the **Get Streams** button. The Aleri Studio lists the streams that are currently running on the Sybase Aleri Streaming Platform.
4. Right-click on the stream to be viewed and select either **View Stream Data** (for tabular data) or **View Event Graph** from the context menu. You can do this with more than one stream — each stream will open in a separate tab.

6.3.1. Copy and Paste from the Streamviewer Window

If you want to use copy and paste to save and get data from a Streamviewer window:

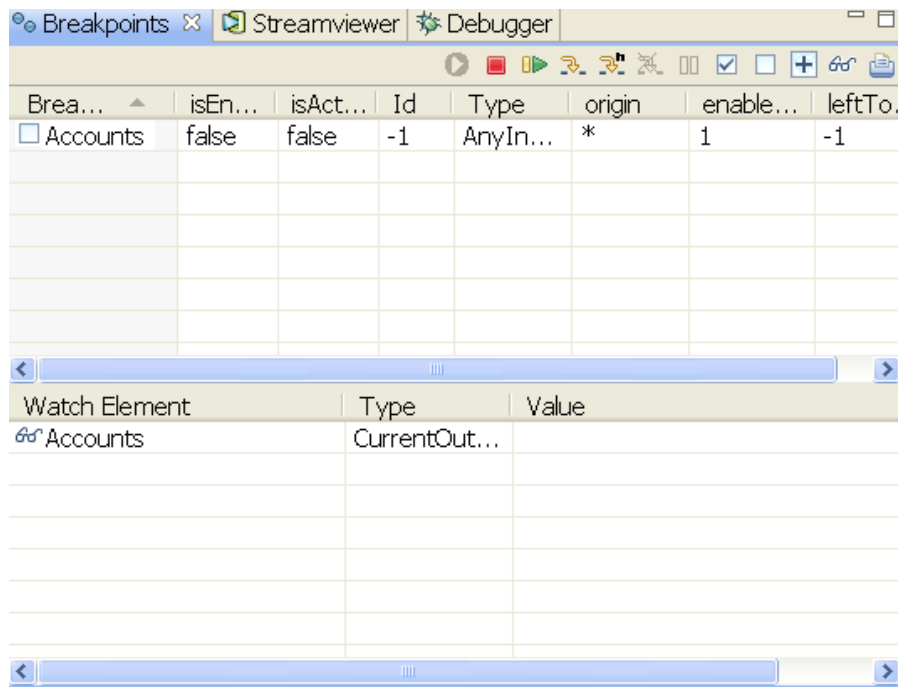
1. Select the Streamviewer user preference **Use Max Available Precision for Streamviewer Save and Copy to Clipboard** to control the precision display of the saved double fields in the clipboard text.
2. When displaying a data table for a stream, you can select one or more rows in the table and then use the context menu choice, **Copy Selection to Clipboard** to place the selected rows on the clipboard in CSV format, which is the same format for saving to a file.
3. You can then go to another application, such as Notepad, and press paste, and the selected data should be pasted as text into the application.

6.4. Debugging with Breakpoints

The Aleri Studio supports breakpoints and watches. In the Aleri Studio Breakpoints view, you can:

- Inspect stream watches.
- Inspect breakpoints.

- Control a running platform instance by enabling tracing, pausing, resuming and stepping of data flow through the Sybase Aleri Streaming Platform streams.



The following buttons are available in the Aleri Studio Breakpoints view:

- | | |
|---|--|
| Trace On | Instructs the Sybase Aleri Streaming Platform to begin tracing(debugging). This parameter must be set in order to use the Sybase Aleri Streaming Platform breakpoint APIs. |
| Trace Off | Stop tracing (debugging). |
| Step Platform | Step the running Sybase Aleri Streaming Platform. |
| Pause Platform | Pause the running Sybase Aleri Streaming Platform. |
| Enable All Breakpoints | Enable all breakpoints in the list. |
| Disable All Breakpoints | Disable all breakpoints in the list. |
| Insert Breakpoint | Insert a breakpoint item into the watch table. |
| Insert Watch | Insert a watch item into the watch table. |
| Print Breakpoint Data to Console | Print the breakpoint and pause state data for the current Sybase Aleri Streaming Platform to the console. |

The following Breakpoint commands initiate potentially long-running operations. Each of these can be canceled before completion by clicking the **Cancel Current Step** button.

- | | |
|-------------------------------|---|
| Step Quiesce from Base | Automatically step all the derived (non-base) streams until their input queues are empty. |
|-------------------------------|---|

Step Quiesce	Automatically step the stream and all its direct and indirect descendants until all of them are quiesced.
Step Transaction	Automatically step until the end of transaction.
Step Quiesce Downstream	Similar to Step Quiesce Stream , except that the descendants of the stream are stepped while the stream itself is not.

Note:

Breakpoints and watches are persisted to the user's workspace.

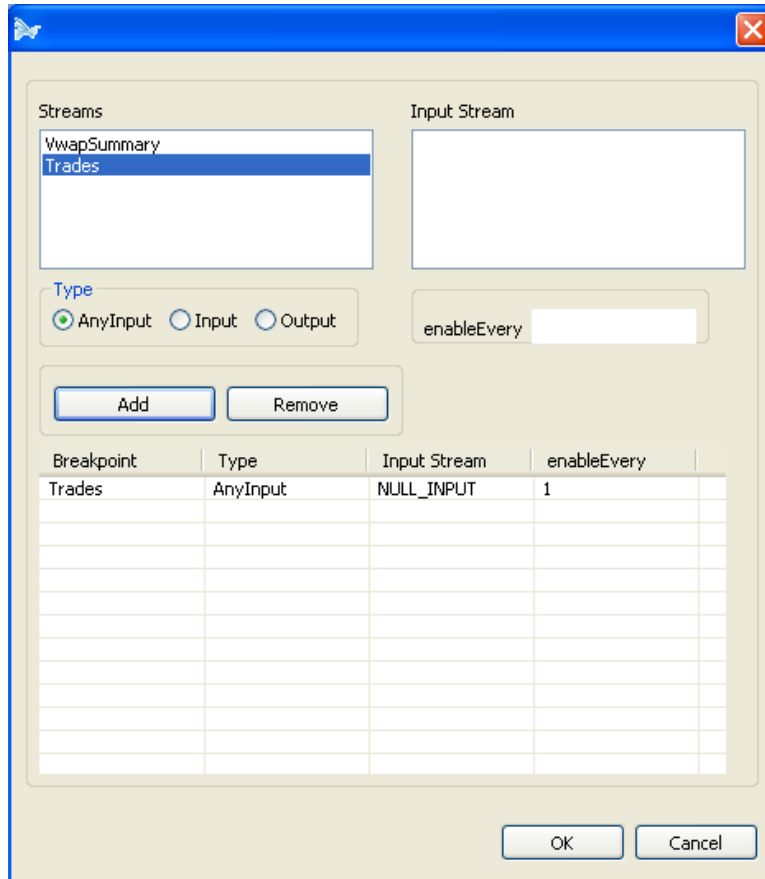
During a Sybase Aleri Streaming Platform run, a `BadRecord_exception` may be added to the breakpoints list when a "bad record" exception occurs in the Sybase Aleri Streaming Platform. One example of a bad record is when an update event is sent for a record that does not exist (no keys match).

6.4.1. Managing Breakpoints

A breakpoint can be inserted for any stream in the data model. There are three different kinds of breakpoints:

AnyInput	will break on input to the stream
Input	used to break on a specific input stream to a stream (only Flex, Join, and Union can have multiple input Streams)
Output	will break when data is output from the Stream

A breakpoint can also have a counter associated with it (`enableEvery`). When a counter n is associated with a breakpoint, the breakpoint will only be triggered after n events flow through the breakpoint (it is then reset to zero).



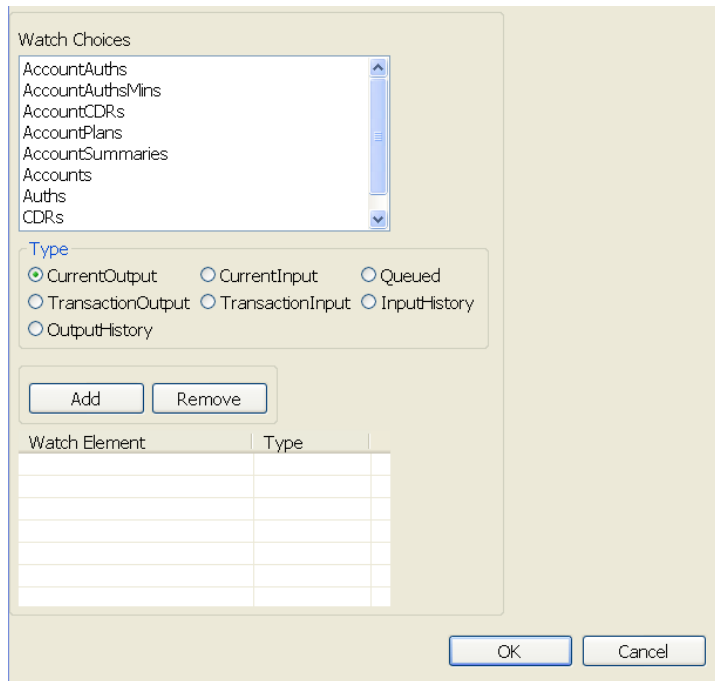
6.4.2. Managing Watches

There are various types of watch items that can be inserted into the watch table of the Breakpoints view. A watch can correspond to any of the following:

- the Current Input of a stream
- the Current Output of a stream
- the Queue of a stream
- the Transaction Input of a stream
- the Transaction Output of a stream
- the Output History of a stream
- the Input History of a stream
- a RowLocalStorage item of a Compute Stream or FlexStream
- a variable of a FlexStream

Before running the Sybase Aleri Streaming Platform, add the watches you want the Aleri Studio to monitor to the watch table. When the Sybase Aleri Streaming Platform runs, the watch table is dynamically updated as run-control events (run, step, pause, and others) are sent through the Sybase Aleri Streaming

Platform.

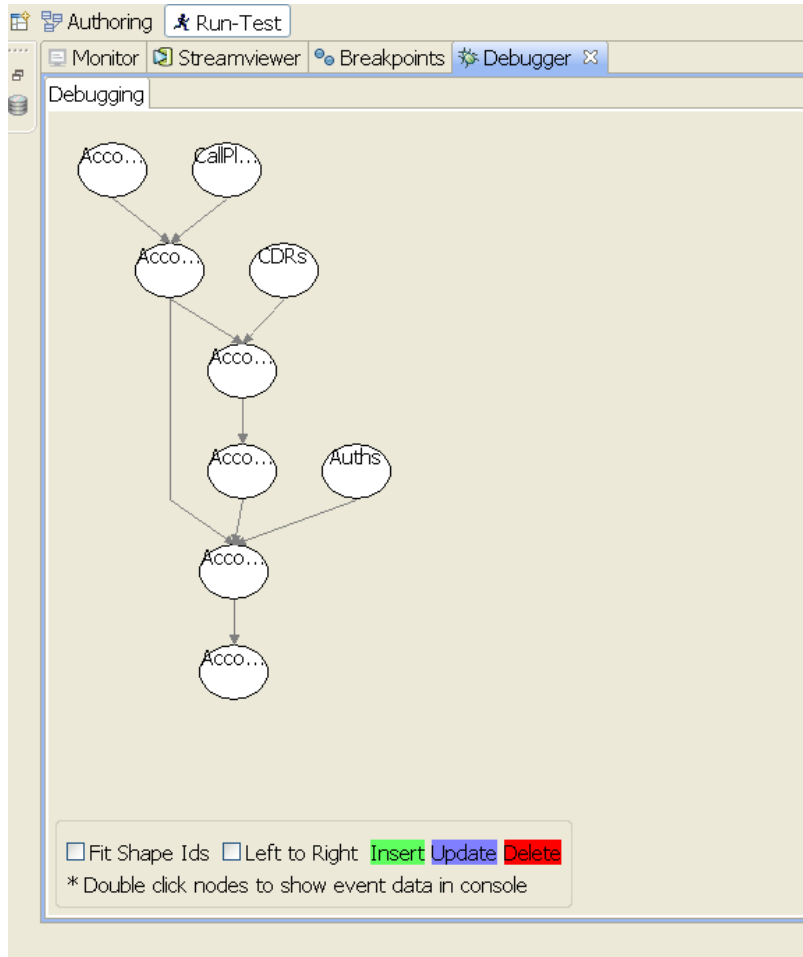


6.5. Debugger

The Aleri Studio Debugger is a useful tool for debugging a data model. It shows the transaction flow through the model and allows the data in each node (stream) to be viewed.

Note:

The Sybase Aleri Streaming Platform debugger only works within a single-server installation of the Sybase Aleri Streaming Platform.



The Debugger area has the following buttons:

- Select Streams** Chooses the streams that have to be part of the visualization.
- Start** Engages the Debugger and collects events on each node. The color of the node will change corresponding to the last type of event that the node received. A green node indicates that an insert occurred, a blue node indicates an update occurred, and a red node indicates that a delete occurred.
- Stop** Ends monitoring events, clears the data on the nodes, and resets the node colors.
- Save** Keeps the image as a JPG file.
- Zoom In** Enlarges the size of the image.
- Zoom Out** Reduces the size of the image.
- Clear** Cleans out the data on the nodes and resets the node colors.
- Click on Node** Prints to the debug console all the events that have been collected for these streams since the last time that **Clear** was clicked.

6.5.1. Run the Debugger

To run the Debugger:

1. Ensure that the Sybase Aleri Streaming Platform is running.
2. Click the **Debugger** tab in the Platform monitoring area.
3. Click **Select Streams** for the Select Streams window to appear.
4. Select the diagram that displays the streams you want to debug, or select **All Streams** to display the entire model.
5. Click **Start** to start the Debugger.

As a transaction is processed by each node, the color of the node changes to reflect the type of transaction. Click on a node to display the corresponding stream's data in the Console. You can use the **Zoom In** and **Zoom Out** buttons to enlarge or shrink the diagram.

Once the Debugger is running, the upload tab can be used for test data. The most useful test data will be files with a single transaction or small sets of transactions. Alternatively, you can use the Data Input view (see [Section 6.1.7, "Data Input view"](#)) to manually enter individual transactions and then view the impact on each stream in the Debugger.

6.6. Running/Testing a Sybase Aleri Streaming Platform Data Model from the Command Line

The Sybase Aleri Streaming Platform offers an extensive suite of tools for running and testing a data model from the command line. See the *Utilities Guide* for details.

Chapter 7. Monitoring, Tuning and Optimization Techniques

This section contains information to help optimize data models for maximum performance.

7.1. General Optimization Techniques

- In an aggregation expression, use the `any` function instead of `max` or `min` wherever possible. This makes the processing more efficient.

Note:

To use `any` reliably, confirm that all the values in the column for a particular group have the same values. Otherwise, the result is not deterministic.

- Unlike a traditional RDBMS (relational database management system), the Sybase Aleri Streaming Platform does not require the use of an aggregation function in non-group-by columns. The Sybase Aleri Streaming Platform still picks one value from the group. This is functionally equivalent to using the `any` function. The above note also applies in this situation.
- When two or more derived streams use very similar and compatible joins and filters, consider creating an intermediate derived stream that is the result of this join and filter. Then use this derived stream for further processing.
- To make an exact copy of a table in Aleri SQL, use the `SELECT * from tablename` syntax. When using XML, use the Union Stream to achieve the same affect. This is much more efficient than writing individual rules to copy columns.
- Filter any unwanted records as soon as possible. This speeds up downstream processing and uses fewer resources.

7.2. Log Store Optimization Techniques

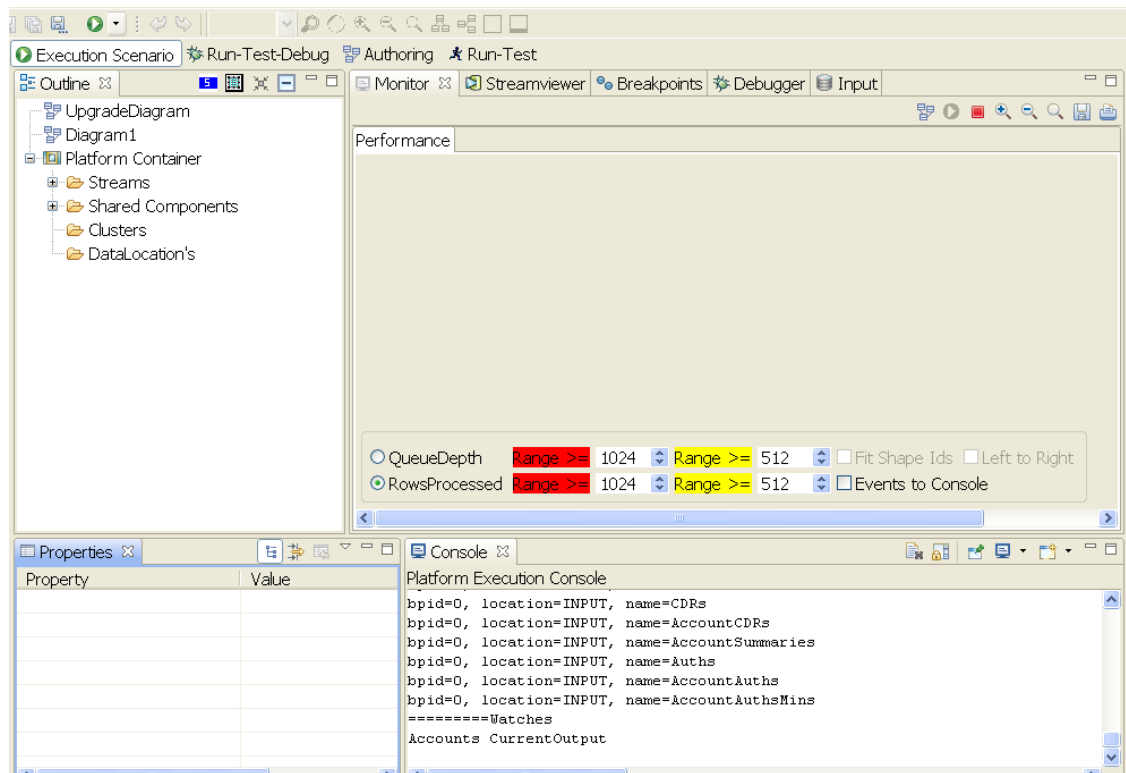
- Whenever possible, create a small Log Store to store static (dimension) data and one or more larger Log Stores for dynamic (fact) data.
- If multiple Log Stores are being used for larger, rapidly changing, dynamic (fact) data, try to organize the stores on different RAID volumes.
- The correct sizing of Log Stores is extremely important. The procedure for determining the correct size of a Log Store is described in the *Administrator's Guide section on Managing Log Stores*.

Chapter 8. Aleri Studio Execution Scenarios

An Execution Scenario Perspective is similar to the Run-Test Perspective, but it gives the ability to define, save and edit multiple execution scenarios.

The Execution Scenario Perspective provides the following areas:

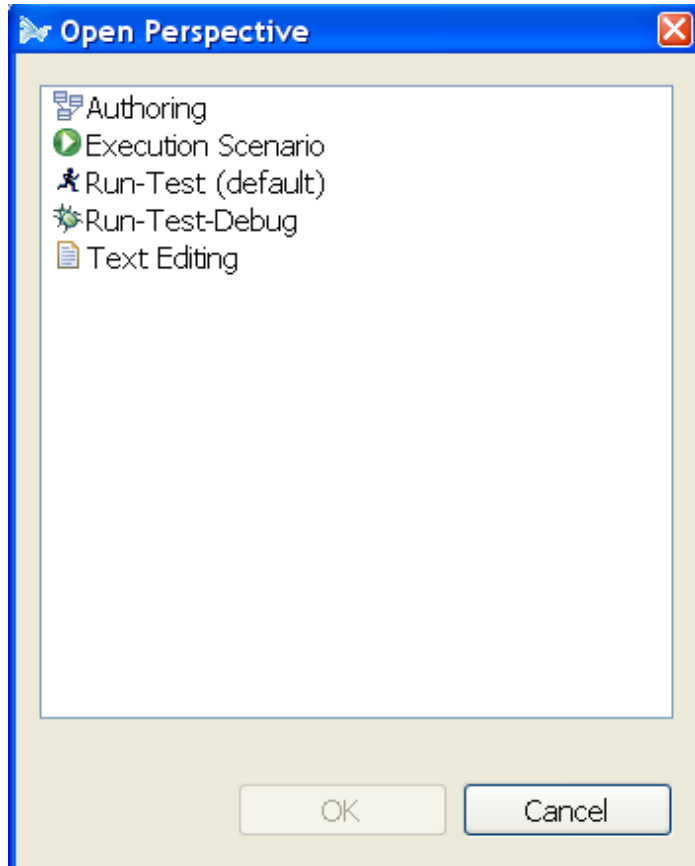
- Outline area** Allows you to add, edit, and delete execution scenario information and invoke various tools from the execution scenario and child nodes.
- Tools Area** Allows you to run the Monitor, Streamviewer, Record/Playback and Debugger.
- Properties area** Allows you to update the properties of an execution scenario and related child elements.
- Console area** Displays the results of tools that are run in Command Execution.



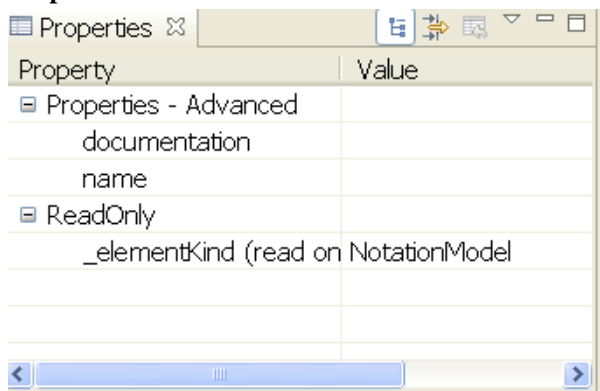
8.1. Creating Execution Scenarios

Execution scenarios provide an iterative development execution environment for testing data models. Each execution scenario contains all the information required to start the Sybase Aleri Streaming Platform. In addition, an execution scenario can store information such as file upload descriptors, path aliases, SQL statements, and client commands.

An execution scenario is created in the Outline area using the context (right-click) menu for a diagram or a NotationContainer.



Once an execution scenario is created in the Outline area, you can view and edit its properties in the **Properties** area.



8.1.1. Create an Execution Scenario

1. Open the Execution Scenario Perspective.
2. In the **Outline** area, right-click a diagram or a notation container and select **Create Execution Scenario**.
3. In the **Properties** area, update any of the following properties as necessary:

Property	Value
_elementKind (read only)	ExecutionScenario
commandControlPort	22333
comment	
debugLevel	debug4
encryptMode	false
encryptPath	
logType	stderr
monitorDuration	5
name	ExecutionScenario1
platformHost	ganges
precision	6
runAsDaemon	false
runOptimized	true
sqlPort	22133
studioHost	localhost
studioPort	22533
usePathAliases	true

8.2. Adding Child Elements to an Execution Scenario

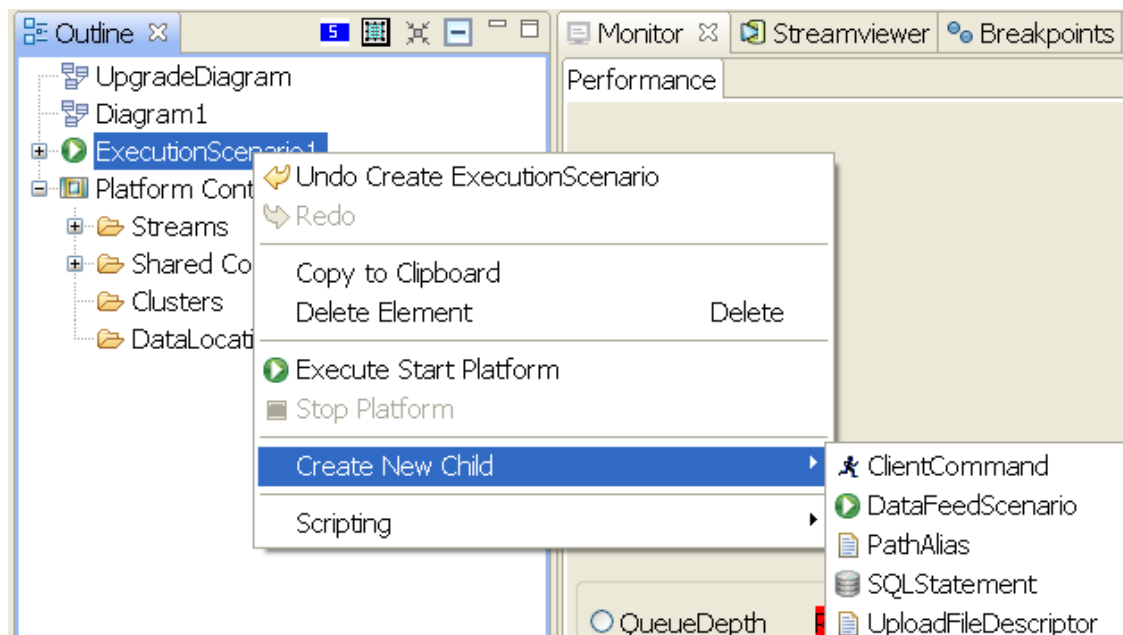
After creating an Execution Scenario and configuring its properties, you can add more child elements. The available elements are:

UploadFileDescriptor	Specifies an upload file that will be used as data input for the running Sybase Aleri Streaming Platform instance. The properties of the FileUploadDescriptor are essentially the same as the arguments used by the sp_upload utility function. Each UploadFileDescriptor can be modified using the Properties area and can also be executed using the Execute Upload File option.
Client Command	Its value is a command that will be sent to the Sybase Aleri Streaming Platform (using the <code>sp_cli</code> utility function). The results from the ClientCommand execution are printed to the Console area. Each ClientCommand can be modified in the Properties area and can also be executed using the “Execute sp_cli command” option.
SQLStatement	This property's value is a query that will be sent to the Sybase Aleri Streaming Platform when the Execute SQL statement is invoked. The results from the SQLStatement execution are printed to the Console area. Long-running queries can be stopped using the Stop running SQL statement.
PathAlias	Allows the user to create a mapping from one file location to another file location. If, for example, the user is running the Aleri Studio on a Win32 client and will be running the Sybase Aleri Streaming Platform on a UNIX server, a path alias is used to map the Win32 paths to UNIX paths. The following are the example values: <ul style="list-style-type: none"> • UNIX Server Name = myserver • Win32 file open = X:\shared\user\vwap\vwap.xml • Win32 mapping = X: (myserver:/opt)

- actualPath = /opt
- aliasPath = X: (myserver:/opt)

Path aliases are used when `ExecutionScenario.usePathAliases=true`. These are also used during the following operations:

- `UploadFileDescriptor` `filePath` is converted to use path aliases (if a matching alias exists).
- A filename is passed into `sp_server` during `Execute Start Platform`. The `<DataModel>.xml` file name is converted to use path aliases (if a matching alias exists).



8.2.1. To add child elements to an Execution Scenario:

1. In the Outline area, right-click an Execution Scenario and select one of the following:
 - **Create Upload File Descriptor**
 - **Create Path Alias**
 - **Create SQL Statement**
 - **Create Client Command**
 - **Create DataFeedScenario**
2. In the **Properties** area, update the applicable properties for the new child element.

8.3. Working with DataFeedScenarios

A `DataFeed Scenario` is a child element of an `Execution Scenario`. It is the root element that controls

how a simulated data scenario will function.

A Data Feed Scenario can be configured to run a specific set of test data. When the Data Feed Scenario is run, it injects the specified data set(s) into the Sybase Aleri Streaming Platform. This enables the user to iteratively run “simulated” data through the system with a minimum amount of configuration.

8.3.1. Create a Data Feed Scenario

To do this:

1. Create an Execution Scenario. See [Section 8.1, “Creating Execution Scenarios”](#) for details.
2. In the Outline area of the Aleri Studio, right-click an Execution Scenario, move the mouse cursor over **Create New** in the context menu that appears, and select **DataFeedScenario** from the sub-menu.

A new DataFeedScenario is created and populated with a collection of DataFeedScriptors, one corresponding to each source stream in the data model.

3. Configure each DataFeedScriptor to control how the associated source stream is populated with data as the Execution Scenario runs.

8.3.2. Parameters for Data Feed Scenarios

comment	A comment to describe the element
feedRateDelay	The number of milliseconds delay between each gateway IO publishing call
name	The name of the element
numberOfTimesToRun	The number of times to run the DataFeed
transactionBlockSize	In the Gateway IO field that describes the number of records to be fed into a block.
useTransactional	A Gateway IO flag. The value <code>true</code> indicates that one has to publish in transactional mode. <code>false</code> means publish in envelope mode. Refer the Gateway IO section for details.

Collection of DataFeedStreamDescriptor

A DataFeedStreamDescriptor is automatically created for each source stream in the model. These represent the base data that will be fed into the model when running the DataFeedScenario.

comment	A comment to describe the element.
name	The name of the element.
numberOfRowsToGenerate	An integer number of data rows to generate for this DataFeedStreamDescriptor, when run.

Collection of ColumnDataFeedDescriptorValues

A `ColumnDataFeedDescriptorValue` describes how each column will be generated for the `DataFeedScenario`.

<code>fillKind</code>	{random, constant, set, step} Describes how the data will be filled on a per column basis
<code>random</code>	Random values will be generated for each column
<code>constant</code>	A constant value will be used for the column. The constant value is set in the <code>rangeBegin</code> field.
<code>set</code>	A set of values is defined. These are created by creating child <code>ColumnSetValue(s)</code> for the <code>ColumnDataFeedDescriptorValue</code> .
<code>step</code>	A step function is applied to each row generated. The value will start at <code>rangeBegin</code> and each new value will be <code>rangeBegin + (n * step)</code> .

A `ColumnSetValue` is used when defining the `fillKind` as 'set'. A new `ColumnSetValue` is defined for each element in the set and the value field holds the actual value that will be injected through the system. The running `DataFeedScenario` will just loop through the set values in order.

Adjusting the delay on a running `DataFeedScenario`

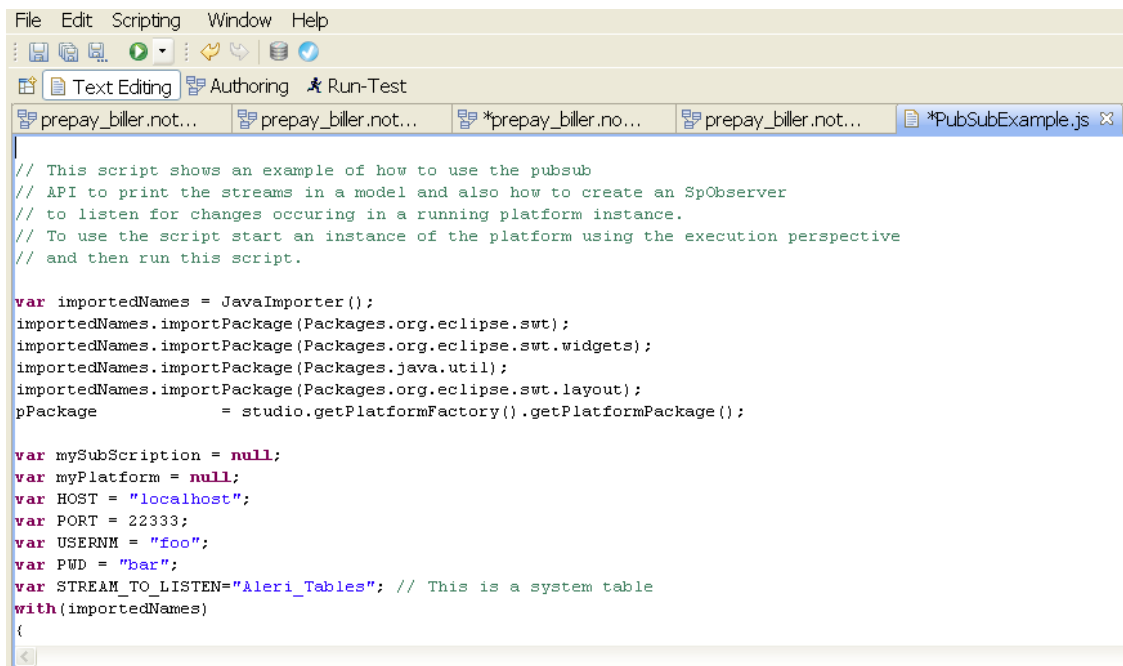
When a `DataFeedScenario` is running, you may wish to adjust its `feedRateDelay`. To do this, right-click the running `DataFeedScenario` in the Outline view and select **Adjust Running Data Feed Delay** from the context menu. A dialog is displayed and is initialized to the current value of `feedRateDelay`. Change the `feedRateDelay` of the running `DataFeedScenario` by changing this value. The value in the dialog is not persistent and only affects the running scenario.

Chapter 9. Studio Scripting

The Aleri Studio can run Javascript™-based scripts within its environment. Scripts can be written to create and modify models, present new dialogs to the user, access the Publish and Subscribe API, among other things (see also <http://www.mozilla.org/rhino/>).

The Script context menu provides a number of editing functions as well as the following options:

- **Run Script** runs the script in the text editor.
- **Check Syntax** checks the syntax of the script and places the errors in the Problem View and console.
- **Toggle Comments** places single line comment markers for the source lines selected in the editor.



The screenshot shows the Aleri Studio interface with a menu bar (File, Edit, Scripting, Window, Help) and a toolbar. The main window displays a text editor with a JavaScript script. The script includes comments and code for importing packages, setting variables, and using the 'with' statement to access properties of the importedNames object.

```
File Edit Scripting Window Help
...
Text Editing Authoring Run-Test
prepay_bill... prepay_bill... *prepay_bill.no... prepay_bill.not... *PubSubExample.js
// This script shows an example of how to use the pubsub
// API to print the streams in a model and also how to create an SpObserver
// to listen for changes occurring in a running platform instance.
// To use the script start an instance of the platform using the execution perspective
// and then run this script.

var importedNames = JavaImporter();
importedNames.importPackage(Packages.org.eclipse.swt);
importedNames.importPackage(Packages.org.eclipse.swt.widgets);
importedNames.importPackage(Packages.java.util);
importedNames.importPackage(Packages.org.eclipse.swt.layout);
pPackage          = studio.getPlatformFactory().getPlatformPackage();

var mySubScription = null;
var myPlatform = null;
var HOST = "localhost";
var PORT = 22333;
var USERNM = "foo";
var PWD = "bar";
var STREAM_TO_LISTEN="Aleri_Tables"; // This is a system table
with(importedNames)
{
```

9.1. Create a New Script File

- On the **File** menu, move the mouse cursor over **New File** and select **New Script File**.
- Type in a name for the script file and click **Finish**. This will create the new script file and open the script editor.

9.2. Run Scripts

You can run a script by selecting **Run Script** or **Run Registered Script** from the main tool-bar **Scripting** menu. The **Scripting** menu option is also available on the context menus for each shape and outline view item.

9.3. Register a Script

Register a script at the User Preference page. To register a script to participate in the **Scripting->Run**

Registered menus, open the **User Preference** window and add a new script to the list of registered scripts.

9.4. Studio Scripting Example

Here is an example HelloStudio script:

```
// This is the HelloStudio script
// This script prints the text Hello Studio to the
// Studio script console (out is an alias in Studio Scripting for the
// studio scripting console.
function helloStudio()
{
    out.println("Hello Studio");
}
// Main part of script
// just call the helloStudio function
helloStudio();
```


Appendix A. Aleri Studio Keyboard Shortcuts

The Studio Shortcuts view is a quick reference for the keyboard shortcuts available when authoring a model. The following table describes the keyboard shortcuts in detail.

Context	Description	Keyboard Action	Result
Any shape on a diagram	Toggle Images	ALT+t	Toggles Image/verbose mode for shape.
Any shape, outline node, or UI widget	Invoke Context Sensitive Help	CTRL+F1 (UNIX)	When a UI element (Shape, outline view tree node or a Studio window) is selected, pressing F1 or Ctrl+F1 will display the context sensitive topic for the appropriate underlying element.
Any shape on a diagram	Deep Delete	DELETE	Performs a Deep delete of selected items. This removes the selected shapes from the diagram and the associated semantic elements of each shape from the data model.
Any shape on a diagram	Delete from Diagram	CTRL+DELETE	Performs a delete of the selected shapes from the diagram. This does not remove the associated semantic elements of each shape from the data model.
Any shape, outline node, or UI widget	Invoke Context Sensitive Help	F1 (win32)	When a UI element (Shape, outline view tree node or a Studio window) is selected, pressing F1 or Ctrl+F1 will display the context sensitive topic for the appropriate underlying element.
Any shape on a diagram	Invoke Inline Edit	F2	Invoke inline editing of a shape compartment item
Compartment items	Invoke Inline Edit	F2	Invoke inline editing of a shape compartment item
Expression, method, pattern compartment items	Non-default Expression Edit	F3	Open non-default expression editor for Expression method and pattern
Compartment item	Insert New Compartment item	INSERT	Insert a new Compartment after selected item
Compartment item	Insert new Compartment	CTRL+SHIFT+ IN-	Insert a new Compartment

Context	Description	Keyboard Action	Result
	item	SERT	ment above selected item
Any selection on diagram or outline view.	Undo last command	CTRL+Z	Undo last command
Any selection on diagram or outline view.	Redo last command	CTRL+Y	Redo last command
Diagram	Select All	CTRL+A	Select all shapes on diagram

The following table shows keyboard shortcuts for the Aleri Studio Text Editor (SPLASH, SQL and Javascript).

Keyboard Action	Description
CTRL+/'	Toggle Comment
CTRL+k	Find Next
CTRL+Shift+k	Find Previous
CTRL+o	Display operators
CTRL+j	Incremental Find
Ctrl+Shift+j	Incremental Find Reverse
Alt+/'	Word Completion
CTRL+l	Go to line
Ctrl+Alt+Down	Copy lines
Ctrl+d	Delete line
Ctrl+'d'	Delete line
Ctrl+Delete	Delete Next Word
Ctrl+Backspace	Delete Previous Word
Ctrl+Shift+Delete	Delete to End of Line
Ctrl+Alt+Up	Duplicate Lines
Ctrl+Shift+Enter	Insert Line above current Line
Shift+Enter	Insert line below current line
End	Line End
Home	Line Home
Alt+Down	Move Lines Down
Alt+Up	Move Lines Up
Ctrl+Right	Next word
Ctrl+Left	Previous Word
Ctrl+Down	Scroll Line Down
Ctrl+Up	Scroll Line Up
Shift+End	Select Line End
Shift+Home	Select Line Start
Ctrl+Shift+Right	Select next word
Ctrl+Shift+Left	Select previous word

Keyboard Action	Description
Ctrl+End	Text End
Ctrl+Home	Text Start
Ctrl+Shift+Y	To Lower Case
Ctrl+Shift+X	To Upper Case
Ctrl+Space	Content Assist
Ctrl+C	Copy
Ctrl+Insert	Copy
Ctrl+X	Cut
Shift+Delete	Cut
Ctrl+V	Paste
Shift+Insert	Paste
Ctrl+A	Select All
Ctrl+F	Find
Ctrl+P	Print
Ctrl+S	Save

The following table shows keyboard shortcuts for Aleri Studio Expression Editor.

Keyboard Action	Description
CTRL+Space	Display input stream columns

Index

A

- Advanced Authoring> Concepts, 53
- Aggregate Stream, 23
- Aleri Studio
 - Authoring, 6
 - File Types, 6
 - Preferences, 41
- Aleri Studio Scripting, 95
- Authoring
 - Advanced Concepts, 53
 - Aleri Studio, 6
 - Environments, 5
 - Introduction to, 1
 - SQL, 44

B

- Breakpoints
 - Debugging with, 81

C

- Checking for Errors
 - Data Model, 34
- Clustering
 - Distributed Models, 58
- Compute Stream, 24
- Connections
 - Input, 17
 - Output, 17
- Copy Stream, 25
- Creating Data Model, 7

D

- Data Location
 - Adding, 18
 - Explorer, 17, 19
- Data Model
 - Checking for Errors, 34
 - Creating, 7
 - Monitor, 77
 - Opening, 8
 - Properties, 41
 - Running and Testing, 65
 - Streams
 - Building Blocks of, 2
 - String Search, 40
- Data Models
 - Distributed
 - Clustering, 58
- Data Storage Managers, 36
- Dataflow Programming
 - , 1
- Debugging
 - Breakpoints, 81

- Visual Event Flow, 85
- Defining Streams, 20
- Derived Streams
 - Types of, 4
- Diagrams, 37
- Distributed Models
 - Clustering, 58

E

- Errors
 - Data Model
 - Checking for, 34
- Event
 - Publish Single, 72
- Event Streams
 - vs. RDBMS Concepts, 1
- Execution Scenarios, 89
- Expiry, 53
- Explorer
 - Data Location, 17, 19
- Expressions, 48
- Extend Stream, 24

F

- File Types
 - Aleri Studio, 6
- Filter Stream, 26
- FlexStream, 30
- Functions
 - Global, 20
 - Local, 32

G

- Global Functions, 20
- Global Variables, 20

I

- Input Connections, 17
- Introduction
 - to Authoring, 1

J

- Join Stream, 27
- Joins, 55

K

- Keyboard Shortcuts, 97

L

- Local Functions, 32
- Local Variables, 32
- Log Store
 - Optimization Techniques, 88

M

Managers
 Data Storage, 36
Model
 Properties, 41
 String Search, 40
Models
 Distributed
 Clustering, 58
Modules
 , 39
Monitor
 Data Model, 77

O

Opening Data Model, 8
Optimization Techniques
 General, 88
 Log Store, 88
Output Connections, 17

P

Pattern Matching, 50
Pattern Stream, 29
Persistent Subscribe Pattern
 , 64
Perspective
 Aleri Studio
 Run-Test, 65
Preferences
 Aleri Studio, 41
Programming
 Dataflow, 1
Properties
 Data Model, 41
Publish
 Single Event, 72

R

RDBMS Concepts
 Event Streams vs., 1
Retention, 53
Run-Test Perspective
 Aleri Studio, 65
Running and Testing
 Data Model, 65

S

Scripting
 Aleri Studio, 95
Shortcuts
 Keyboard, 97
Source Stream, 21
SQL
 Authoring, 44
Stores, 54
Stream
 Aggregate, 23

 Compute, 24
 Copy, 25
 Extend, 24
 Filter, 26
 Flex, 30
 Join, 27
 Pattern, 29
 Source, 21
 Union, 28
Streams
 Data Model
 Building Blocks of, 2
 Defining , 20
 Derived
 Types of, 4
 Properties of, 3
 Streamviewer, 80
 String Search
 Data Model, 40
 Studio
 Preferences, 41
 Studio Execution, 65
 Studio Scripting, 95
 Subscribe Pattern
 Persistent , 64

U

Union Stream, 28

V

Variables
 Global , 20
 Local , 32
Visual Authoring
 Workspace, 8
Visual Event Flow
 Debugging with, 85

W

Workspace
 Visual Authoring, 8