SYBASE®
An SAP® Company

Administrators Guide

Sybase Aleri Streaming Platform 3.2

DOCUMENT ID: DC01296-01-0320-02

LAST REVISED: December, 2010

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

1. Preface

This guide provides procedures for administering the Sybase® Aleri Streaming Platform . These include:

- managing user accounts
- configuring logging and managing log files
- backing up data and data models
- monitoring performance and dealing with problems
- managing authentication mechanisms
- reconfiguring to enhance security, maintain maximum availability and optimize performance

This guide is intended for administrators who will manage the Sybase Aleri Streaming Platform and applications used with it.

2. Related Documents

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

Product Overview	Introduces the Aleri Streaming Platform and related Aleri products.
Getting Started - the Aleri Studio	Provides the necessary information to start using the Aleri Studio for defining data models.
Release Bulletin	Describes the features, known issues and limitations of the latest Aleri Streaming Platform release.
Installation Guide	Provides instructions for installing and configuring the Streaming Processor and Aleri Studio, which collectively are called the Aleri Streaming Platform.
Authoring Guide	Provides detailed information about creating a data model in the Aleri Studio. Since this is a comprehensive guide, you should read the *Introduction to Data Modeling and the Aleri Studio*. first.
Authoring Reference	Provides detailed information about creating a data model for the Aleri Streaming Platform.
Guide to Programming Interfaces	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. These interfaces include: • 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
Utilities Guide	Collects usage information (similar to UNIX® man pages) for all Aleri Streaming Platform command line tools.
Administrators Guide	Provides instructions for specific administrative tasks related to the Aleri Streaming Platform.
Introduction to Data Modeling and the Aleri Studio	Walks you through the process of building and testing an Aleri data model using the Aleri Studio.
SPLASH Tutorial	Introduces the SPLASH programming language and illustrates its capabilities through a series of examples.
Frequently Asked Questions	Answers some frequently asked questions about the Aleri Streaming Platform.

Chapter 1. Basic Administrative Tasks

This chapter provides instructions and information for day-to-day maintenance of the Sybase Aleri Streaming Platform.

1.1. Provisioning Users

The Sybase Aleri Streaming Platform does not have its own concept of users or groups; it relies on the user/group provisioning mechanisms of the operating system. The Sybase documentation refers to "groups" as "roles" (a term from the RDBMS world). See Section 2.4.1, "Roles for Access Control" for more details about roles.

You can choose to use Pluggable Authentication Modules (PAM), RSA keys, or Kerberos® for authentication. See Section 2.1, "Configuring PAM Authentication", Section 2.2, "Configuring RSA Authentication", and Section 2.3, "Configuring Kerberos™ Authentication" for details.

Group maintenance depends on the PAM module in use. For example, when using the pam_unix module, you can modify (add, delete or change) groups by editing the `/etc/groups` system file. Maintaining groups for other PAM modules is a normal administration task for system administrators.

For a new user, you must create the account using the operating system's standard tools, Then you must provision the account to authenticate to the Sybase Aleri Streaming Platform. If using PAM authentication, this is done automatically. For RSA authentication, you must be create and install private and public key files. See Section 2.2, "Configuring RSA Authentication" for details.

1.2. 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 then 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.

1.3. Logging

This section provides instructions for managing the system's logging mechanisms.

1.3.1. Error Logging

1.3.1.1. Default Error Logging

If the Streaming Processor is run in foreground mode, error messages are logged to stderr by default.

If the Streaming Processor is run as a daemon (if started with **sp** or **sp-opt** using the `-s` switch), error messages are logged to syslog by default.

In either case, the default logging level is 4.

1.3.1.2. Configure Error Logging

Log Message Destination

To configure error logging explicitly, start the Streaming Processor with **sp** or **sp-opt** using the `-l`

switch. See the *Utilities Guide* for details.

Logging Level

To set the logging level explicitly, start the Streaming Processor Configuring with **sp** or **sp-opt** using the `-d` switch. The value for the `-d` switch should be an integer between 0 and 7. The command turns on logging messages from level zero up to the specified level, inclusively.

The **sp_cli *`loglevel`*** command can be used to change the logging level while the Streaming Processor is running.

These are the logging levels used by the Streaming Processor:

Name	Level	Description
`LOG_EMERG`	0	/* system is unusable */
`LOG_ALERT`	1	/* action must be taken immediately */
`LOG_CRIT`	2	/* critical conditions */
`LOG_ERR`	3	/* error conditions */
`LOG_WARNING`	4	/* warning conditions */
`LOG_NOTICE`	5	/* normal but significant condition */
`LOG_INFO`	6	/* informational */
`LOG_DEBUG`	7	/* debug-level messages */

See the *Utilities Guide* for more details about the `-d` option to the **sp_cli** command.

Error Log Messages

Each log message starts with a fixed format header which indicates the logging level and the location within the Sybase Aleri Streaming Platform, and contains a unique message number. The format for the log message header is:

`[SP-`*`XX`*`-`*`MMMNNN`*`] (`*`time`*`) sp(`*`pid`*`)`

Where

`XX` is the logging level

`MMM` is the module in which the message is logged (useful for troubleshooting)

`NNN` is the message type number

`time` is the number of seconds after the Sybase Aleri Streaming Platform startup that this message was generated (useful for troubleshooting)

`pid` is the process id for the Streaming Processor process

The message text follows this fixed header.

Two typical Streaming Processor logging messages:

```
[SP-06-114000] (0.040) sp(6328) Platform(fxr)::Platform(): Created Platform.
[SP-06-114039] (0.040) sp(6328) Platform(fxr)::XML Model Version: NULL
```

1.3.2. Bad Data Logging

The Streaming Processor generates an error message whenever it discards a bad record. The Streaming Processor can also be configured to save the bad record itself (see Section 1.3.2.2, “Configure Bad Data Logging”).

1.3.2.1. Defaults for Streaming Processor Bad Data Logging

By default, the Streaming Processor logs an error message whenever data is discarded. Bad data error messages show up at logging level 4.

1.3.2.2. Configure Bad Data Logging

To configure the Streaming Processor to save discarded records to a Bad Records File, start the Streaming Processor with **sp** or **sp-opt** using the `-B` switch. See the *Utilities Guide* for details.

1.3.2.3. Bad Data Log Messages and Bad Data File Entries

The error messages related to bad data have the same header as other Streaming Processor error messages (see above for details). The following examples show what bad data error messages look like:

```
... StoreIndex(BaseInput)::put() bad insert, tid=1.
... StoreIndex(BaseInput)::put() -- roll back transaction of size 5.
```

Each bad data log message includes a transaction id (`tid=1` in the example above). If the Streaming Processor is configured to write the actual bad records to a separate file, each record in the file also references the transaction id, as in the following examples:

```
[2007-01-17 16:53:15] Bad insert writing to store. (tid=1)
        <BaseInput a="1" b="e"/>
[2007-01-17 16:53:15] roll back (tid=1)
        <BaseInput a="6" b="a"/>
        <BaseInput a="7" b="b"/>
        <BaseInput a="8" b="c"/>
        <BaseInput a="9" b="d"/>
        <BaseInput a="1" b="e"/>
```

This allows you to correlate message in the Streaming Processor log files and the Bad Records File.

1.4. Backing Up and Restoring Data

Data backup is an integral part of a data management and protection strategy. The system administrator must back up Sybase Aleri Streaming Platform data regularly. Two types of data should be backed up:

Data Models — The XML files that define the data streams and how they interact (plus associated

files)

Log Stores — An Sybase Aleri Streaming Platform stream can be configured to write its contents to a log store. This data should be backed up so that the Sybase Aleri Streaming Platform's state can be restored after a system failure.

The best way to make backups is to stop the Sybase Aleri Streaming Platform and do an “off-line” backup. It is possible, however, to perform an “on-line” backup of the data models and log stores while the Sybase Aleri Streaming Platform is running.

Some notes that pertain to both types:

- In the simplest case, the data model only exists as an `.xml` file. Depending on how it was created, however, a single data model may also be stored as a `.sql` file (the SQL version) and/or have an associated `.notation` file (stored settings related to how the data model is displayed in the Aleri Studio). Typically, these files all have the same name (but different extensions) and all are stored in the same directory. All of these files should be added to the backup set.
- Although the Sybase Aleri Streaming Platform only runs one data model at a time, the users of your Sybase Aleri Streaming Platform installation may have developed more than one data model. Make sure you back up all the data models and associated log stores your users have created.

1.4.1. Off-line Back-up

Follow this procedure to back up the data model and log files while the Sybase Aleri Streaming Platform is down.

1. Verify the locations of all data model and log store files.

   There is no default directory for data model files.

   Each data model may have zero, one or more log stores (because each stream can have its own Log Store, or many streams can use the same log store, or all streams can be configured with no log store).

   Find out where the data model files are, and what type of Store is defined for each stream, before you shut down the Sybase Aleri Streaming Platform.

2. Make sure no data streams are publishing or subscribing to the Sybase Aleri Streaming Platform.

3. Shut down the Streaming Processor.

```
$PLATFORM_HOME/bin/sp_cli -p port -c aleriusr [:password] stop
```

Where:	
port	is the number of the port used by the Streaming Processor
aleriuser	is the login of a user account
password	is the password for that user account

4. Back up the files for each data model and its associated log stores.

   a. On Linux® or Solaris®, use the **tar** system utility.

      Suppose the Sybase Aleri Streaming Platform data model is named `algorithmic1.xml` and located in the `/home/aleri/models/trading/` directory. All the streams in this data model are configured to write to a single log store, `algorithmic1`, located in the `/fastraid` directory.

      Back up the model and its log file to the file `backup.tar`.

      ```
      cd /

      tar -cvf backup.tar home/aleri/models/trading/algorithmic1.xml \
      fastraid/algorithmic1
      ```

   b. On Windows® use the **pkzip** freeware utility or some equivalent.

      Suppose the Sybase Aleri Streaming Platform data model is named `algorithmic1.xml` and is located in the `c:\user\aleri\models\trading` directory. Also suppose all the streams in this data model are configured to write to a single log store, `algorithmic1` located in the directory `d:\fastraid`.

      Back up the model and its log file to the `backup.zip` file.

      ```
      c:

      cd \

      pkzip -r backup.zip c:\user\aleri\models\trading d:\fastraid\algorithmic1
      ```

   ## Note:

   These directories may be copied to an alternate hard disk (or CD/DVD ROM drive), and any other backup tool (WinZip, Windows backup, ...) may be used. As long as all the files in `c:\user\aleri\models\trading` and `d:\fastraid\algorithmic1` are saved, the backup is complete.

The default extension for data model files is `.xml`. In the example commands shown, it was assumed that that there were no other data model files. However, data model files can include associated `.sql` and/or `.notation` files. If you have any of these associated files, you must back them up along with the `.xml` files.

See Section 1.4.4, “Restore Backup Files” for information about restoring the data from backup files.

1.4.2. On-line Back-up

An on-line backup can be done without stopping the Sybase Aleri Streaming Platform.

Note:

Although you do not have to stop the Sybase Aleri Streaming Platform, its operation will be suspended while the backup files are created. (The length of this suspension depends on the amount of data accumulated in the log stores). This may cause disruption to the users. For this reason, online backups should be performed only at times when such short disruptions are acceptable.

To perform an on-line backup of the Sybase Aleri Streaming Platform data models and log stores:

1. Make sure that the interruption in service will be acceptable at that time.
2. Use the **sp_cli** utility to create a backup copy of log store files, as in the following example:

   $PLATFORM_HOME/bin/sp_cli -p *9100* -c *aleriusr*[:*password*] backup

   Where *9100* is replaced by your installation's Command & Control port number. And, *aleriuser* and *password* identify a valid Sybase Aleri Streaming Platform user account.

   This command creates a set of backup files in the log store directories, each with extension `.bak`. These files are created as sparse files: only the current contents of the stores are copied over.
3. Use the **tar** system utility (on Linux or Solaris) or the **pkzip** tool (on Windows) to make backup files for each data model and the files associated with its log stores.

   Typically, data model files have the extension `.xml`. There is no default directory for data model files.

   Each data model may have zero, one or more log stores.

   Example for Linux and Solaris:

   In this example, the data model is `algorithmic1.xml`, in the directory `/home/aleri/models/trading/`. All the streams in this data model are configured to write to a single log store, `algorithmic1`, in the directory `/fastraid`.

   You would back up the model and its log file to the file `backup.tar` by executing the following commands:

   ```
   cd /

   tar -cvf backup.tar home/aleri/models/trading/algorithmic1.xml \
   fastraid/algorithmic1/*.bak
   ```

   The backup of this log store would be created in the same directory.

   Example for Windows:

   In this example, the data model is `algorithmic1.xml`, in the `c:\user\aleri\models\trading\`. directory All the streams in this data model are configured to write to a single log store, `algorithmic1`, in the directory `d:\fastraid`.

   You would back up the model and its log file to the file `backup.zip` by executing the following commands:

   ```
   c:
   ```

```
cd \
pkzip -r backup.zip c:\user\aleri\models\trading d:\fastraid\algorithmic1
```

Notes:

- In the Windows example, these directories may be copied to an alternate hard disk (or CD/DVD ROM drive), and any other backup tool (WinZip, Windows backup, ...) may be used. As long as all of the files in `c:\user\aleri\models\trading` and `d:\fastraid\algorithmic1` are saved, the backup is complete.
- In both examples, the data model exists only as an `.xml` file. Make sure that you back up any `.sql` and/or `.notation` files associated with the data model file.

See Section 1.4.4, “Restore Backup Files” for information about restoring the data from backup files.

1.4.3. View and Store Backup Files

To view the contents of a backup file (for example, `backup.tar`) on a Linux or Solaris system, use the **tar** command.

```
tar -tvf backup.tar
```

To view the contents of a backup file (for example, `backup.zip`) on a Windows system, use the **pkzip** command.

```
pkunzip -v backup.zip
```

1.4.4. Restore Backup Files

Use the appropriate method for your operating system.

1.4.4.1. Restore Backup Files on a Linux or Solaris System

To do this:

1. Stop the Streaming Processor (if it is not already stopped).

```
$PLATFORM_HOME/bin/sp_cli -p port -c aleriusr[:password] stop
```

where *`port`* is the port number used by your Streaming Processor and *`aleriuser`* and *`password`* identify a valid Sybase Aleri Streaming Platform user account.

Or, you can click the **Stop Platform** button in the **Start/Stop** tab panel in the Run-Test perspective.

2. Extract the files from the backup archive. For example, you could extract the files from `backup.tar` file with the following commands:

```
cd /

tar -xvf backup.tar
```

3. If the backup was created in online mode (see Section 1.4.2, "On-line Back-up"), rename all the `.bak` files to `.log` files. Assuming a Korn-type shell (such as `ksh` or `bash`) and the same locations of the files, this can be done with the command:

```
for i in fastraid/algorithmic1/*.bak; do mv $i ${i%%.bak}.log; done
```

Notes:

- The Sybase Aleri Streaming Platform must be stopped in order to perform this restoration procedure. The Sybase Aleri Streaming Platform can be restarted after all logfiles are restored and renamed.
- If the original versions of the log store files still exist, you can over-write them.
- After archived files are extracted, the `*.bak` files are not sparse any more, and may take up much more actual disk space than they originally did. A compression program such as **gzip** can be used to reduce the size of the backed-up files.

4. Restart the Streaming Processor.

1.4.4.2. Restore Backup Files on a Windows System

To do this,

1. Stop the Streaming Processor (if it is not already stopped) by executing the following command:

   $PLATFORM _HOME/bin/sp_cli -p *port* -c *aleriusr*[:*password*] stop

   Where *port* is the port used by the Streaming Processor and *aleriuser* and *password* identify a valid Sybase Aleri Streaming Platform user account.

   Or, you can click the **Stop Platform** button in the **Start/Stop** tab panel in the Run-Test perspective.

2. Use **WinZip** or **pkunzip** to extract the files from the backup files. The following commands are for **pkunzip**:

```
c:

cd \

pkunzip backup.zip
```

3. If the backup was created in online mode (see Section 1.4.2, "On-line Back-up"), rename all the

`.bak` files to `.log` files, using Windows Explorer.

Notes:

- The Streaming Processor must be stopped in order to perform this restoration procedure. The Streaming Processor can be restarted after all logfiles are restored and renamed.
- If the original versions of the log store files still exist, you can overwrite them.

4. Restart the Streaming Processor using the standard port number, user account, and data model name for your installation.

1.5. Monitoring the Streaming Processor

You can monitor the Streaming Processor in several ways:

- using the **sp_monitor** command from the command line
- using the **Monitor** tool in the Aleri Studio
- looking at the system log files

Both the **sp_monitor** command and the **Monitor** tool, enable you to monitor performance on a per-stream basis. They report on the following statistics:

- CPU utilization percentage
- number of rows processed in the last monitoring interval
- number of records queued for stream processing
- total number of records processed by a stream

You can configure the Streaming Processor to send messages to stderr, syslog or both. These messages can be monitored for changes or abnormal conditions.

1.5.1. Monitoring with sp_monitor

The **sp_monitor** command reads performance data from a running instance of the Sybase Aleri Streaming Platform, and displays it in XML format on the standard output. A set of performance records, one per stream, is obtained from the Streaming Processor every *n* seconds. The time interval *n* is specified when the Streaming Processor is started. (See the *Utilities Guide* .) These records are displayed in the following XML format:

```
<Aleri_Streams_Monitor ALERI_OPS="i"  stream="AveragePrices"
cpu_pct="0.000000" trans_per_sec="0.000000" rows_per_sec="0.000000"
inc_trans="0" inc_rows="0" queue="0" store_rows="0"
last_update="2008-07-10 21:28:38"/>
```

where:

`ALERI_OPS`	is the operation performed on the stream with this record: "d" for delete, "i" for insert, "u" for upsert
`stream`	is the name of the stream being reported on
`cpu_pct`	is the percentage of CPU (one core) that this stream's thread used in the time period since the last update
`trans_per_sec`	is the stream's performance in transactions per second, in the time period since the last update
`rows_per_sec`	is the stream's performance in rows per second, in the time period since the last update
`inc_trans`	is the Number of transactions processed since the last update
`inc_rows`	is the number of rows processed since the last update
`queue`	is the current input queue size
`store_rows`	is the current number of records in the specified stream's store
`last_update`	the date and timestamp of the current update to the specified stream, in YYYY-MM-DD hh:mm:ss format

For example, to monitor a Streaming Processor running on a host `amazon.sybase.com` with the Command & Control port set to `31415`, use the following command:

```
sp_monitor -c aleriusr:password -p amazon.sybase.com:31415
```

Note:

In order for **sp_monitor** to work properly on Windows, the Streaming Processor must have been started in one of the following ways:

- With the **sp_server** script, or
- with `<NUL` at the end of the command line, as in the following example:

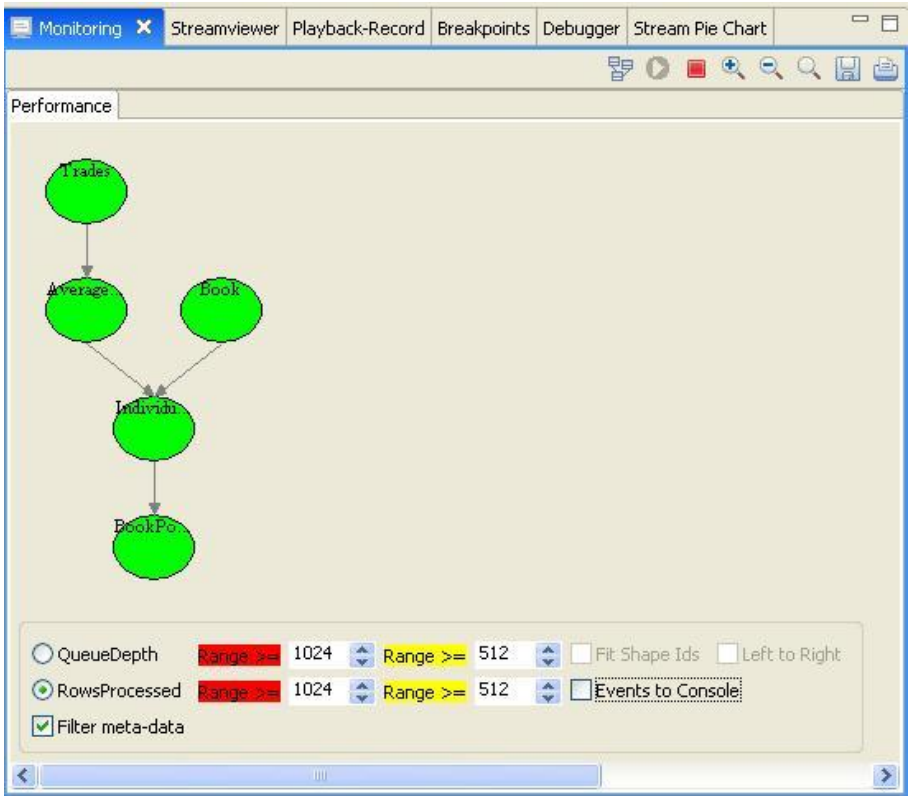
```
sp-opt -d 7 -f ...<filename> -c 31415 ... <NUL
```

See the *Utilities Guide* .) for more information about **sp_monitor**, **sp_server**, and **sp_opt**.

1.5.2. Monitoring the Streaming Processor Using Aleri Studio

You can monitor the Streaming Processor from the Aleri Studio using the Diagram Monitor window. To monitor all the available streams, select `All Streams` in the following dialog box.

Once you have selected a diagram (or `All Streams`), the corresponding image is displayed in the Aleri Studio Monitoring area:



In this figure, each node represents a stream in the model, and each arrow represents data flowing from one stream to another. Each node is colored dynamically red, yellow, or green based on values for either **Queue Depth** (that is, the number of rows queued up to be processed by this stream) or **Rows Processed**. These values (and colors) change over the course of time.

You can control the ranges for red, yellow, and green displays. For example, if you set the range for the red for **Queue Depth** to be >= 125, and the yellow to be >= 20, then the window will display node with color based on following properties:

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

Note:

If a stream is not ready to process a record it receives, the stream that sent the record deposits it in the receiving stream's input queue. If the receiving stream is working slowly, its input queue may grow large. A stream's Queue Depth is the measure of the size of this input queue.

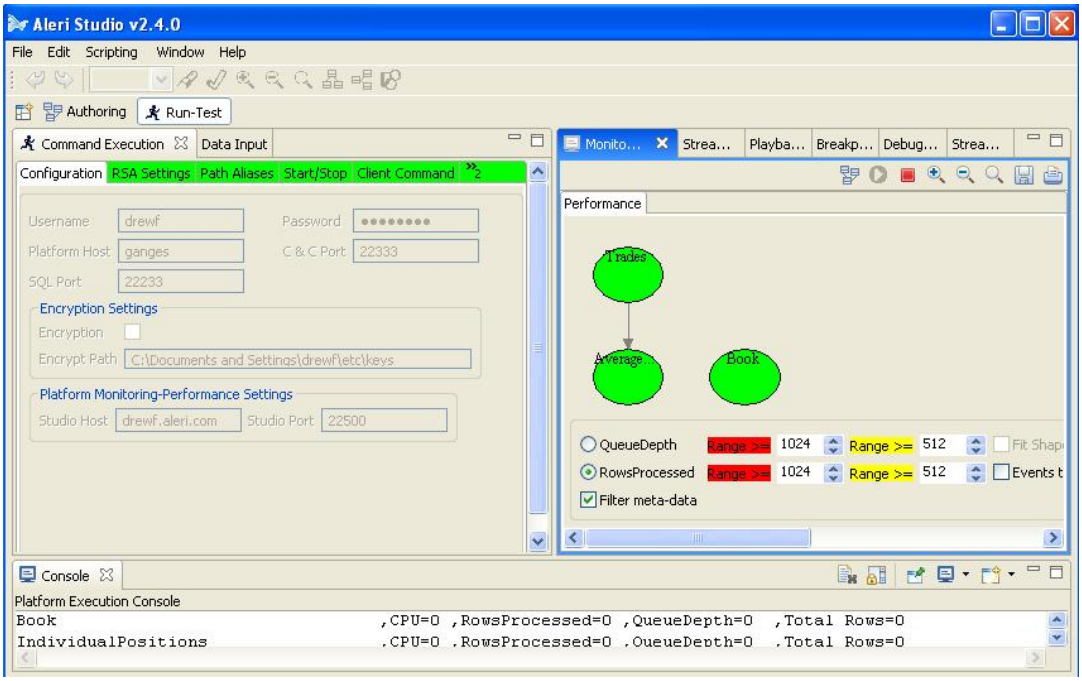
The receiving stream's queue does not grow unbounded. Once it reaches a certain limit (that can be seen or adjusted with the **sp_cli** throttle property), the senders stop and wait for the queue size to drop before depositing more data. This means that the sender streams' input queues begin to back up — and the effect can propagate all the way to the Source Streams. If the Monitoring tool shows several high Queue Depth values, this doesn't mean that all these streams are bottlenecks — just the last stream in the sequence is.

In this display, the Aleri Studio also depicts CPU utilization as the black pie wedge in the ellipse. The CPU utilization represents the use of one CPU core; for machines with multiple processors or multiple cores or both, the wedges may add up to well over 100%.

1.5.2.1. Monitoring Performance

Follow these steps to set up the **Monitor** tool:

1. Open the Aleri Studio Run-Test Perspective. (Refer to the *Authoring Guide* for instructions).



2. Click the **Configuration** tab in the Aleri Studio Command Execution area.
3. Enter the following values to begin the monitoring session:

- Type a valid username and password for the Sybase Aleri Streaming Platform in the *Username* and *Password* fields.
- Type the host name for the Sybase Aleri Streaming Platform in the *Platform Host* field.
- Type the port name for the Sybase Aleri Streaming Platform in the *C & C Port* field.
- Type the host name for the Aleri Studio in the *Studio Host* field.
- Type the port name for the Aleri Studio in the *Studio Port* field.

4. Click on the **Monitoring** tab in the Aleri Studio Performance Monitoring area.

   -or-

   From the **Windows** menu, point to Show View and then click on the **Monitoring** tab

5. Click the **Select Streams** button.

   The **Select Streams** window appears.

6. Select the diagram with the stream whose performance you want to monitor.

   -or-

   Select **All Streams** to monitor the performance of all streams in one diagram.

7. Click on the **OK** button.

8. Select the **QueueDepth** or **RowsProcessed** button to determine how to color each node in the performance diagram.

   - Use the up and down arrow buttons in the **(Red) Range** >= field to specify the range that will result in a red node.
   - Use the up and down arrow buttons in the **(Yellow) Range** >= field to specify the range that will result in a yellow node.

   ## Note:

   Nodes will be green when they fall within the range that is not covered by the **(Red) Range** >= or **(Yellow) Range** >= fields.

9. Click on the **Start Monitoring** button. The Sybase Aleri Streaming Platform colors the nodes in the diagram accordingly.

10. If necessary, click on **Zoom In** or **Zoom Out** or **Zoom Page** to see a larger or smaller view of the diagram.

11. If you wish to send the performance data to the console, click on the **Print Performance Data to Console** button.

12. Click on the **Stop Monitoring** button to stop the monitoring.

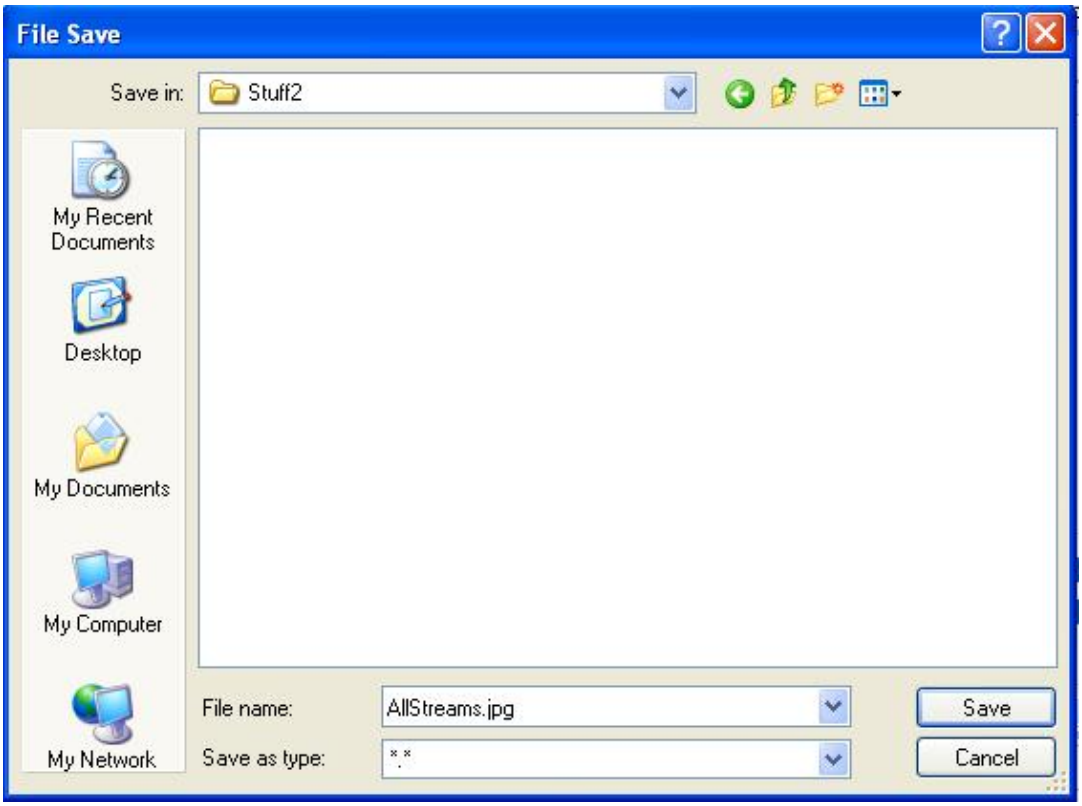
1.5.2.2. Save an Image of the Performance Diagram

You can save the performance diagram as a JPEG file. When the image is saved, the Console window

also displays a report of all stream performance information.

To do this:

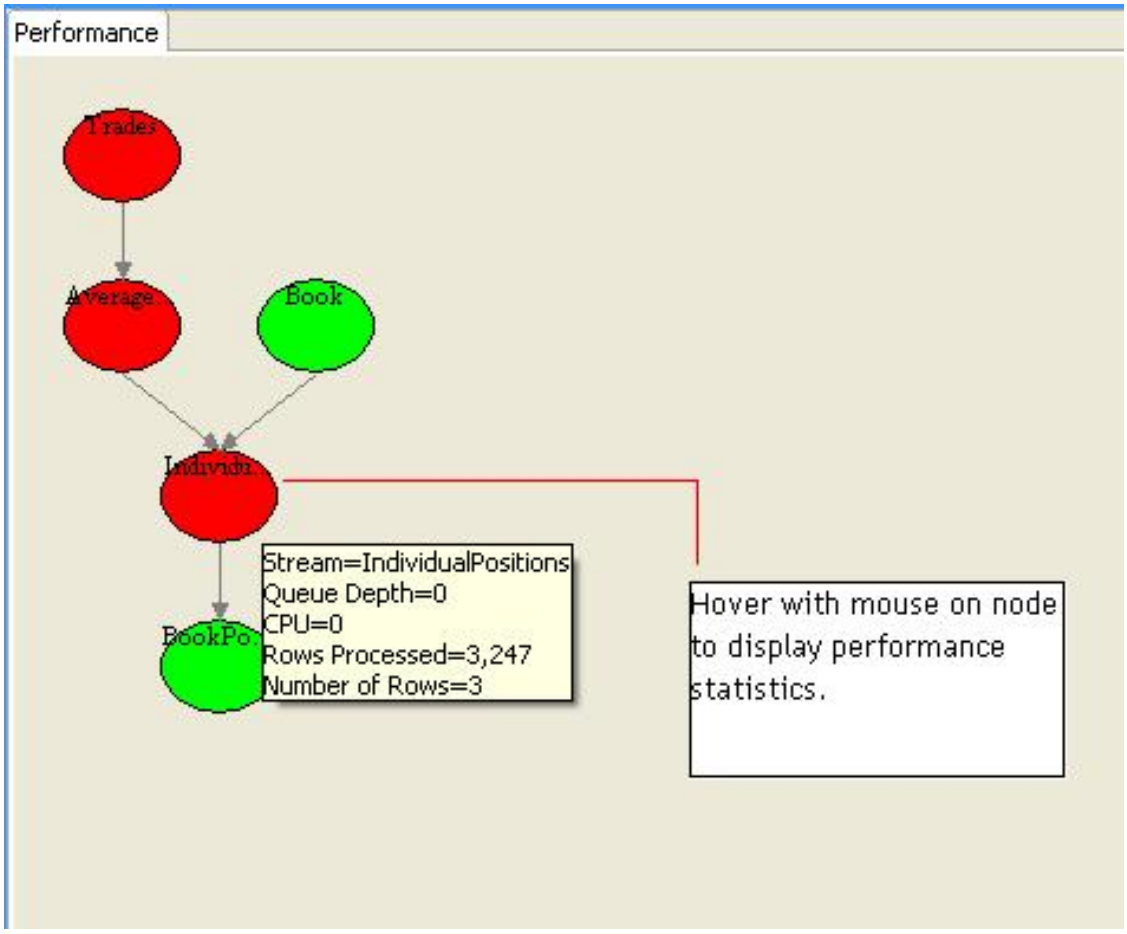
1. Set up the Monitoring tool to monitor the performance of a Sybase Aleri Streaming Platform instance. Refer to Section 1.5.2.1, “Monitoring Performance” for more information.
2. Click the **Monitoring** tab in the Monitoring area.
3. Click **Save**. The **Save Diagram as JPG File** window appears.
4. Perform the following in the **Save Diagram as JPG File** window:



- Type the fully qualified name for the file in the **Enter File Name** field.
- Click **Save**.

1.5.2.3. Display the Performance Statistics of a Node

You can also display a tool-tip that contains the performance statistics of an individual node within a performance diagram.



1. Use the Monitoring area to monitor the performance of a Sybase Aleri Streaming Platform instance. See Section 1.5.2.1, “Monitoring Performance” for more information.

2. Click the **Monitoring** tab in the Monitoring area.

3. Move the mouse pointer over a node. The Sybase Aleri Streaming Platform displays a tooltip with this node's performance statistics.

1.6. Managing Clients

You can obtain information about clients connected to the Sybase Aleri Streaming Platform by querying the metadata streams.

If the `-q` option, for SQL query support, was used at startup, you can enter a query like the following:

```
echo 'select * from Aleri_Clients' | $PLATFORM_HOME/bin/sp_query \
-c user:password -q 9201
```

If SQL queries are not enabled, you can get the same information as a data snapshot. For example,

```
$PLATFORM_HOME/bin/sp_query -c user:password -p 9200 -S \
```

```
-s Aleri_Clients
```

or, in the SQL form,

```
$PLATFORM_HOME/bin/sp_query -c user:password -p 9200 -S \
-Q 'select * from Aleri_Clients'
```

These examples assume that the Sybase Aleri Streaming Platform control port is 9200 and the SQL port is 9201.

The Aleri_Clients stream lists all the clients currently connected to the Sybase Aleri Streaming Platform through the Pub/Sub interface. The information about which clients are subscribed to which stream is available from the Aleri_Subscriptions metadata stream, or in a more convenient denormalized form from Aleri_Subscriptions_Ext.

If some clients are misbehaving, you may terminate their connection. To do that, you first need to find the client's handle in Aleri_Clients. Then use **sp_cli** to kill that connection. For example, to kill the handle 130, use:

```
$PLATFORM_HOME/bin/sp_cli -c user:password -p 9200 "kill {130}"
```

To make your life easier, you can assign a tag name to the connection using the `-m` option with the **sp_upload** and **sp_subscribe** tools. If you develop your own applications, use the Pub/Sub call `setName()`, and provide some option to specify the name. When querying Aleri_Clients, the tag name is shown in the `conn_tag` column and enables you to identify these clients easily. Multiple clients may have the same tag name. This may be convenient if there are multiple closely-coupled programs, or if one program opens multiple connections.

The **sp_cli** command can kill all the clients with a specified tag name in one go. For example, to kill all the clients with the tag `SomeName`, use the command:

```
$PLATFORM_HOME/bin/sp_cli -c user:password -p 9200 "kill every {SomeName}"
```

The metadata streams provide much more information about the state of the Sybase Aleri Streaming Platform For a full description, see Appendix F in the *Authoring Reference* .

1.7. Managing Log Stores

The Sybase Aleri Streaming Platform provides a way to persistently store information even when it is not running, and restart with the data already loaded. This is done with log stores (available as an option).

Unlike memory stores, log stores don't extend automatically. Their size must be specified in advance in the model XML file. Sizing the log stores correctly is important. A store that is too small requires more frequent cleaning cycles, which severely degrades the performance. In the worst case, the log store can overflow and cause the processing to stop. A store that is much too large also causes performance issues due to the larger memory and disk footprint, but nowhere as severe as the log stores being too small.

To size a log store, follow this procedure:

1. Estimate the maximum amount of data that may be collected in the log store, as both record count and volume in bytes.

   If you are reasonably certain about both the number of records arriving in the source stream(s) and the size of the records, you can simply do the calculation. If not, you might want to use the Play-back feature to record and play back real data to get a better idea of the amount of data you need to handle. Also the log store reports in syslog the amount of data in it as “liveSize” when the model exits (with log level 3 or higher) and after every compaction (with log level 6 or higher).

2. Multiply the record count by 96 bytes to calculate the basic indexing overhead. Add the result to the volume in bytes.

   The informational messages in the Sybase Aleri Streaming Platform syslog report the “liveSize” size with the indexing overhead already included. So if you have used these messages to estimate the store usage of a data sample, skip this step.

3. Choose the value of the parameter *`reservePct`*. The reserve is kept as intermediate space that is used at the time of periodic cleaning of the store. It is also required to perform the correct resize of the store. If the reserve is chosen too small and the model allowed to run until the store fills with data, the following resize attempt may cause the store to become wedged. A wedged store can not be resized any more, and the data may be extracted from it only by Sybase technical support. It's safer to have too much reserve than too little.

   The default of 20% would be adequate for most situations. The multi-gigabyte stores may use a reduced value, as low as 10%. The small stores (under 30 MB), especially those with many streams, may require a higher reserve, up to 40%. If you find that 40% is still not enough, increase the size of the store.

   The Sybase Aleri Streaming Platform automatically estimates the required reserve size and increases the reserve if it is too small. It mostly affects the small stores. However, increasing the reserve reduces the amount of space left for the data. Monitor the syslog messages for these automatic adjustments when you start a new model for the first time. You may need to increase the store size.

   The required store size in bytes including the reserve will then be calculated from the data size found in the previous step as:

   ```
   storeBytes = dataBytes * 100 / (100 - reservePct)
   ```

   Round it up to the next megabyte.

4. Make sure that the reserve could not be overrun by the uncheckpointed data.

   As the store runs, more records are written into it until the free space falls below the reserve. (The details are more complicated, as the exact point is auto-adjusted to optimize the performance, but the lowest it can get is the reserve size). Then the source streams are temporarily stopped, the streams quiesced, and the checkpoint and cleaning cycle are performed. The streams don't quiesce immediately, first they must process any data collected in their input queues. Any data produced during quiescence will be added to the store, and this means that the reserve must be big enough to accommodate this data and still have enough space left to perform the cleaning cycle. If this data overruns the reserve, the store will become wedged, since it won't be able to clean itself.
   The automatic reserve calculation mentioned in the previous step does not account for the uncheckpointed data, since it does not have the required knowledge about the model.

Estimate the maximum amount of uncheckpointed data that can be produced if the input queues of all the streams (except the source streams) are full. The records in the queues that are located early in the sequence must be counted together with any records they produce as they are processed through the model. The pass-through ratio needs to be taken into account as well: how many output records are produced by the stream for each of its input records.

The following example shows the process with the Sybase Aleri Streaming Platform stream queue depth set to the default of 1024, for a log that contains four streams ordered like this:

```
source --> derived1 --> derived2 --> derived3
```

a. Determine the number of records that may be produced by each stream as it consumes the contents of its queue. In our example,

   - 1024 records may end up in derived1's input queue. Assuming that it produces one output record for one input record, it may produce 1024 records.
   - 2048 records may end up in derived2's input queue (1024 that may be already collected on its own queue and 1024 more from derived1 after it processes its queue). Suppose that derived2 is a join and generates on average 2 output records for each input record. Then it may produce (1024 + 1024) * 2 = 4096 records.
   - 5120 records may end up in derived3 (1024 from its own queue and 4096 from derived2 after it processes all its records). Assuming the pass-through ratio of 1, it would produce 5120 records.

   When the model topology is not linear, all the branches have to be taken into the account. The pass-through ratio may be different for data coming from the different parent streams. The data from all the input paths has to be added up. Each stream has only one input queue, so its depth is fixed, no matter how many parent streams it's connected to. However the mix of records in it may vary. To be on the safe side, assume the whole queue composed from the records that produce that highest amount of output. Some input streams may contain static data that gets loaded once and never changes during the normal work. These inputs don't need to be counted. In the example derived2 may be such a join stream, having static data as its second input.

b. Then calculate the space required by multiplying the total number of records by the average record size of that stream. If the records in derived1 average at 100 bytes, in derived2 at 200 bytes, and derived3 at 150 bytes, the calculation would be:

   (1024*100) + (4096*200) + (5120*150) = 1689600

   The tracing of the record count must be done through the whole model, starting from the SourceStreams and down to at least all the streams in the LogStore. Of course, only the data size from the streams located in the log store is summed up.

c. Multiply the record count by 96 bytes to calculate the indexing overhead and add the result to the volume in bytes, similarly to how it was done for the full data. Continuing the example, this will be:

   (1024 + 4096 + 5120) * 96 = 983040

1689600 + 983040 = 2672640

Compare the result with the reserve size and make sure that it is no higher than 1/4 of the reserve size:

uncheckpointedBytes < storeBytes * (reservePct/4) / 100

If it did come out higher, increase the reserve percent and repeat the store size calculation. The uncheckpointed data becomes a concern mostly for the smallish stores.

As the amount of data in the store grows, and the free space falls below 10% (excluding the reserve), the Sybase Aleri Streaming Platform starts reporting the "log store is nearing capacity" warnings to syslog.

If the data is deleted from the store in bursts, such as collecting the data during the day and discarding the data older than a week at the end of the day, the messages about the log store nearing capacity may appear intermittently even after the old data has been flushed. As the cleaning cycle rolls over the data that has been deleted, they will disappear.

If the log store is not very small, there will be time to see these warnings before it runs out of space. If you see them, stop the Sybase Aleri Streaming Platform at the first convenient moment and increase the store size. Otherwise the Sybase Aleri Streaming Platform will abort when the free space in the model falls below the reserve size.

If the store was mis-sized particularly badly to start with, the whole reserve may become used up. A store in this condition is called “wedged” and can not be resized with the preservation of content. If the store became wedged, the only way out is to delete its files and restart with a clean store. The content may still be extractable by the Sybase technical support if a backup of the store files is made before deleting them.

To resize the store, change the store size in the model, and it will be resized on restart. The store size may be only increased, not decreased. When the model is restarted for the first time after resizing the store, it is likely to produce the syslog messages about the free space being below the reserve until the cleaning cycle assimilates the newly added free space.

Another alternative is the Dynamic Services, where a modified model is applied directly to a running Sybase Aleri Streaming Platform. However the Dynamic Services require twice the amount of memory, enough to contain both the old and the new copy of the store while the modification is applied.

If a stream (such as a FlexStream) uses the context of Local or Global variables in its logic, such a stream should generally use a memory store. Otherwise, when the Sybase Aleri Streaming Platform is restarted, the stream's store would be preserved but the values of the variables would be reset. If these variables are used to create unique keys, these keys won't be unique!

In general, it's a good idea to either place only the source streams into the log stores, or place a source stream and all the streams that are derived from it (directly or indirectly) into the same log store. If the stores are mixed in the sequence of processing, an abrupt halt and restart may cause messages about bad records with duplicate keys detected on restart. With Local or Global variables, it may cause even bigger inconsistencies.

For performance reasons, keep the streams that change at substantially different rates in different log stores. If a log store contains a large but nearly-static stream and a small but rapidly changing stream, each cleaning cycle would have to go through a lot of data from the static stream. Keeping them separate makes the cleaning cycles fast. Note that this advice comes at odds with the approach of keeping the

source stream and all the streams derived from it in the same log store. That's why keeping only the source streams in the log stores and the derived streams in the memory stores is usually a better idea.

The *ckcount* (checkpointing count) parameter affects the size of the uncheckpointed data. The count shows how many records may be updated before writing the intermediate index data. Setting it to a large value amortizes the overhead over many records, to make it almost constant, averaging at about 96 bytes per record, as was assumed in the procedure above. Setting it to a small value increases the overhead. With the count set to zero, the index data is written after each transaction, and for the single-transaction records the overhead becomes:

```
96 + 32 * ceiling(log2(number_of_records_in_the_stream))
```

If a stream is small (for example, it contains only 1000 records), the overhead for each record would be:

```
96 + 32 * ceiling(log2(1000)) = 96 + 32 * 10 = 416
```

In many cases, the record itself would be smaller than its overhead of 416 bytes! Since the effect is logarithmic, the large streams are not affected too badly. A stream with a million records would have a logarithm of 20 and incur an overhead of 736 bytes per record.

Other than through the uncheckpointed data size, this overhead does not significantly affect the store size calculation because the cleaning cycle removes it and compacts the data. However the increased overhead does affect the performance by writing extra data and increasing the frequency of store cleaning.

The *sweepamount* parameter determines how much of the logfile is “swept through” during each cleaning pass. It is always forced to be between 5% to 20% of the *fullsize* parameter. A good lower bound for the sweep size is half the size of the write cache on your storage array. Usually, it indicates a sweep size of 512 to 1024 megabytes. Smaller sweep sizes minimize spikes in latency at the expense of a higher average latency. High values give low average latency, with higher spikes when reclaiming space.

If the value of the *sweepamount* parameter is too small, the system will perform excessive cleaning; in some cases, this does not allow the log store to free enough space during cleaning.

The size of the sweep is also limited by the amount of free space left in reserve at the start of the cleaning cycle. If the reserve is set lower than the sweep amount and the sweep does not encounter much dead data, the sweep will stop when the relocated live data will fill up the reserve (the swept newly cleaned area will become the new reserve for the next cycle). Unless other factors override, keeping the sweep and the reserve sizes close to each other is usually the best approach. Note that the parameter *reservePct* is specified in percent while *sweepamount* is specified in megabytes.

1.8. Configuring Non-standard Data Locations and Playback

1.8.1. Configuring Non-standard Data Locations

The Sybase Aleri Streaming Platform comes with partial implementations of four Data Locations:

- IBM® Websphere™ MQ
- Netezza®
- Teradata®

After installation, these Data Locations can be used for authoring, but cannot be used for discovery or execution without obtaining the vendor's JDBC® driver and installing it. These JDBC drivers are not included with the Sybase Aleri Streaming Platform because Sybase has not been able to obtain the distribution rights.

To fully enable one or more of these Data Locations:

1. Obtain the appropriate JDBC libraries (jar files) from the vendor(s).
   - IBM MQ requires the `com.ibm.mq.jar` and `com.ibm.mq.jms.jar` files.
   - Netezza requires the `nzjdbc.jar` file.
   - Teradata requires the `tdgssconfig.jar`, `tdgssjava.jar`, and `terajdbc4.jar` files.
2. Copy the JDBC libraries (jar files) to the `$PLATFORM_HOME/lib` directory.
3. Test the Data Locations.

1.8.2. Configuring Non-standard Playback Sources

The Sybase Aleri Streaming Platform comes with one partial implementation of a playback source for Teradata. It requires libraries from the vendor that are not included.

To enable this playback source:

1. Copy the file `$PLATFORM_HOME/lib/playback/other/libaleriteradata.so` (`$PLATFORM_HOME/lib/playback/other/aleriteradata.dll` on Windows) to the directory `$PLATFORM_HOME/lib`.
2. Copy the Teradata libraries to the directory `$PLATFORM_HOME/lib/playback`.
3. Test the new source. The new source **Teradata** should appear in the Aleri Studio dialog “Configure Playback Parameters” opened from the **Select Playback DataSource** button.

Chapter 2. Increasing Security

This chapter details modifications the administrator can make to increase the security of the Sybase Aleri Streaming Platform.

2.1. Configuring PAM Authentication

One of the three mechanisms that client applications can use to authenticate to the Streaming Processor is the Pluggable Authentication Module (PAM) package found in Linux and Solaris. (The others are RSA and Kerberos: see Section 2.2, “Configuring RSA Authentication” and Section 2.3, “Configuring Kerberos™ Authentication” for details.)

In PAM authentication, client applications connecting to the Streaming Processor through the Command and Control interface, the Gateway interface, or the SQL Query interface must supply a user name and password before they can issue commands or requests. The Streaming Processor uses PAM to verify that the username and password are correct. It is up to the administrator to specify whether this password check should be made against the UNIX® password file, an LDAP server, or a NIS system. The processes for configuring authentication for the Streaming Processor on Linux and Solaris are described below.

2.1.1. Configure PAM on Linux

To configure authentication on Linux, log in as `root` and create a new file `/etc/pam.d/sp`. The contents of the file will specify the kind of authentication required. Some examples follow.

For no authentication, the contents of the file `/etc/pam.d/sp` should be:

```
auth  required  pam_permit.so
auth  required  pam_warn.so
```

The second line tells the system to log authentication attempts to the standard syslog file.

For UNIX ® password authentication, the contents of the file `/etc/pam.d/sp` should be:

```
auth  required  pam_unix.so
```

Note:

For authentication on Linux and Solaris, the `/etc/shadow` file must be readable by the user that starts the Streaming Processor.

For LDAP authentication, the contents of the `/etc/pam.d/sp` file should be:

```
auth  required  pam_ldap.so
```

2.1.2. Configure PAM on Solaris 10

To configure PAM on Solaris 10, log in as `root` and edit the `/etc/pam.conf` file as described.

For no authentication, add the following lines at the end of the file:

```
#
# Support for Sybase Aleri Streaming Platform
#
sp  auth  required   pam_sample.so always_succeed
```

For UNIX authentication, add the following lines at the end of the file:

```
#
# Support for Sybase Aleri Streaming Platform
#
sp  auth  required  pam_authtok_get.so.1
sp  auth  required  pam_unix_auth.so.1
```

> **Note:**
>
> For authentication on Linux and Solaris, the `/etc/shadow` file must be readable by the user that starts the Streaming Processor.

For LDAP authentication, add the following lines at the end of the file:

```
#
# Support for Sybase Aleri Streaming Platform
#
sp  auth  required  pam_ldap.so
```

2.2. Configuring RSA Authentication

One of the three mechanisms that client applications can use to authenticate to the Streaming Processor is RSA authentication, which uses public and private keys instead of passwords, to authenticate with the Streaming Processor. (The others are the Pluggable Authentication Module (PAM) and Kerberos: see Section 2.1, “Configuring PAM Authentication”, and Section 2.3, “Configuring Kerberos™ Authentication” for details.)

The public key file and a private key file are generated using an **openssl** utility. The public key is copied to a location for the Sybase Aleri Streaming Platform. When a client application connects to the Sybase Aleri Streaming Platform, it sends a user name and a special string encrypted with the private key. The server then checks the signature using the public key file.

2.2.1. Generate an RSA Key

Use the **openssl** utility (provided with the Sybase Aleri Streaming Platform distribution) to generate RSA keys.

For example, if the Sybase Aleri Streaming Platform has been installed in the directory `/home/aleri` and you want to generate RSA keys for a user named *aleriusr*, you would create the private and public key files as follows:

1. Generate the private key file.

   ```
   /home/aleri/bin/openssl genrsa 2048 > aleriusr.private
   ```

2. Generate the associated public key.

```
/home/aleri/bin/openssl rsa -inform PEM -outform PEM \
-pubout < aleriusr.private > aleriusr
```

3. Copy the public key to the appropriate Sybase server directory.

```
cp aleriusr /home/aleri/etc/keys/rsa/
```

4. Restrict the private key file's permissions (so that it is only readable by the owner).

```
chmod 0400 aleriusr.private
```

Note:

The name of the public key file (`aleriusr` in this example) should be the same as the user name. The private key file (`aleriusr.private` in this example) can have any name.

Convert the private RSA key file generated in the example above into a PKCS8 DER formatted file.

```
/home/aleri/bin/openssl pkcs8 -topk8 -nocrypt -in aleriusr.private \
-outform DER -out aleriusr.private.der
```

When starting the Aleri Studio, the `aleriusr.private.der` form of the private key file must be used. It must also be used to perform RSA authentication from other Java® client applications (including those that use the Java Pub/Sub API).

Copy `aleriusr.private.der` onto any client machine that will run a Java client application attempting RSA authentication against the Sybase Aleri Streaming Platform.

2.2.2. Configure the Server

The server executables (**sp**, **sp-opt**, **sp-server**) have an extra flag, `-k` to allow the user to specify a directory that contains the public key files. The public keys created by users in the RSA Key Generation step must be copied to this directory.

For example, if the file “aleriusr” is moved to the directory `/home/aleri/etc/keys/rsa` and the server is started with the command

sp -f config.xml -d 7 -k /home/aleri/etc/keys/rsa
then the user *aleriusr* can use RSA authentication.

2.2.3. Start an Aleri Client With RSA Authentication

Each of the major client executables (**sp_cli**, **sp_cnc**, **sp_convert**, **sp_histexport**, **sp_query**,

sp_subscribe, **sp_upload**) accepts an optional flag `-k` to allow the user to specify the location of the private key file. For example,

```
sp_cli -c aleriusr:password -k /home/aleri/aleriusr.private
```

specifies that the private key file for the user *aleriusr* is located in directory `/home/aleri/aleriusr.private`. The password after the colon following the `-c` flag can be any string, but some value must be included. The client encrypts a special string using the private key, and sends it along with the user name to the Streaming Processor. The server finds the public key with the same user name in the directory specified with its `-k` flag. If the string is decrypted successfully, the server grants access to the user.

2.2.3.1. RSA Authentication and SSL Encryption

The RSA authentication and the SSL encryption of the client connections to the Streaming Processor are independent. To achieve proper security, both RSA and SSL should be used at the same time; RSA alone is not enough. If anything, PAM authentication with SSL encryption is more secure than RSA authentication without SSL encryption. See Section 2.5, “Configuring Secure Socket Layer (SSL) Connections” for the details on the SSL encryption.

2.3. Configuring Kerberos™ Authentication

One of the three mechanisms that client applications can use to authenticate to the Streaming Processor is Kerberos authentication. (The others are the Pluggable Authentication Module (PAM) and RSA: see Section 2.1, “Configuring PAM Authentication”, and Section 2.2, “Configuring RSA Authentication”, for details.)

Kerberos V provides single-sign-on authentication for an entire network. It uses the Generic Security Services Application Program Interfaces (GSSAPI) as an application programming interface for programs to access security services and Simple Authentication and Security Layer (SASL) for authentication and data security in Internet protocols.

2.3.1. Configure Kerberos on Linux and Solaris

1. On Linux systems, install the Heimdal Kerberos™ package, release 1.2.1, which is available at http://www.h5l.org. (There is no need to install Kerberos on Solaris systems.)

2. Verify that the Kerberos configuration file, `krb5.conf`, is properly configured for your REALM/domain, and refers to a valid Kerberos Domain Controller (KDC). On Linux machines, this file is in the `/etc` directory. On Solaris machines, this file is in the `/etc/krb5` directory.

   The following example `krb5.conf` file is for a machine with a fully qualified domain name of krbnj.sybase.com

   ```
   [logging]
    default = FILE:/var/log/krb5libs.log
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmind.log

   [libdefaults]
    default_realm = SYBASE.COM
    dns_lookup_realm = false
    dns_lookup_kdc = false
    ticket_lifetime = 24h
    forwardable = yes
   ```

```
[realms]
 SYBASE.COM = {
   kdc = krbnj.sybase.com:88
   admin_server = krbnj.sybase.com:749
   default_domain = sybase.com
 }

[domain_realm]
 .sybase.com = SYBASE.COM
 sybase.com = SYBASE.COM
```

3. Make sure the Sybase Aleri Streaming Platform picks up the GSSAPI SASL plug-in that is shipped with it. You can do this in two ways (the first of which is recommended):

   a. Set the environment variable `SASL_PATH` to the directory `lib/sasl2` under the directory where you installed the Sybase Aleri Streaming Platform. If you accepted the default location, add the following line to the `.profile` file in each user's home directory.

   ```
   export SASL_PATH=/home/aleri/aleri/platform/3.2.0/lib/sasl2
   ```

   b. Make `/usr/lib/sasl2` a symbolic link to the `lib/sasl2` under the directory where you installed the Sybase Aleri Streaming Platform. If you accepted the default location, enter the following command.

   ```
   ln -s /home/aleri/aleri/platform/3.2.0/lib/sasl2 /usr/lib/sasl2
   ```

4. If this machine will be running an instance of the Streaming Processor, perform the following steps as well.

   a. Add the principal to the Kerberos Domain Controller. For a workstation with a fully qualified domain name (FQDN) of **foo.bar.baz** and a Kerberos realm of **REALM**, the principal would be declared as

   ```
   alerisp/foo.bar.baz@REALM
   ```

   b. Add the principal to the keytab file, `krb5.keytab`, on the workstation. On Linux workstations this file is in the `/etc` directory. On Solaris workstations, this file is in the `/etc/krb5` directory.

2.3.2. Configure Kerberos on Windows

Note

If you have SonicWall software that creates an additional network interface with a new IP address, you may have to disable it in order to run the Sybase Aleri Streaming Platform with Kerberos authentication enabled (-V GSSAPI).

Note

You can obtain multiple valid Kerberos tickets by:

1. authenticating using a Kerberos server other than your current domain controller
2. logging in to one user account and using a different one to connect to the Sybase Aleri Streaming Platform

If you have more than one valid Kerberos ticket, you must ensure that the ticket issued to the account used to connect to the Sybase Aleri Streaming Platform is the default ticket.

1. Install the MIT Kerberos for Windows package, kfw-3.2.2, which is available at http://web.mit.edu/Kerberos.
2. Verify that the Kerberos initialization file, `krb5.ini`, is present in all of the following locations:

   - `C:\Winnt`
   - the `bin` directory beneath the directory where you installed Kerberos, for example, `C:\Program Files\MIT\Kerberos\bin`
   - the `%SystemRoot%` directory for example, `C:\Windows`
   - the `%systemRoot%` directory beneath your home directory, for example, `C:\Documents and Settings\cimtest\Windows`

   The following example `krb5.ini` file is for a machine with a fully qualified domain name of krbnj.sybase.com

```
[libdefaults]
        default_realm = SYBASE.COM
        dns_lookup_realm = false
        dns_lookup_kdc = false
        ticket_lifetime = 24h
        forwardable = yes

[realms]
        SYBASE.COM = {
                kdc = krbnj.sybase.com:88
                admin_server = krbnj.sybase.com:749
                default_domain = sybase.com
        }

[domain_realm]
        .sybase.com = SYBASE.COM
        sybase.com = SYBASE.COM

[logging]
#       kdc = CONSOLE
        default = FILE:C:\Windows\krb5libs.log
```

```
        kdc = FILE:C:\Windows\krb5kdc.log
        admin_server = FILE:C:\Windows\kadmind.log
```

3. Verify that the GSSAPI SASL plug-in, `saslGSSAPI.dll`, is present in the `lib\sasl2` directory beneath the directory where you installed the Sybase Aleri Streaming Platform.

4. Verify that the following registry keys are present:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Carnegie Mellon\Project Cyrus\SASL Library]
    "ConfFile"="C:\\Progra~1\\Aleri\\AleriStreamingPlatform\\lib\\sasl2"
    "SearchPath"="C:\\Progra~1\\Aleri\\AleriStreamingPlatform\\lib\\sasl2"
```

   These registry keys must be present before starting and while running the Sybase Aleri Streaming Platform. if you have other software applications that provide SASL plug-ins.

5. If this machine will be running an instance of the Streaming Processor, perform the following steps as well.

   a. Add the principal to the Kerberos Domain Controller. For a workstation with a fully qualified domain name (FQDN) of **foo.bar.baz** and a Kerberos realm of **REALM**, the principal would be declared as

      ```
      alerisp/foo.bar.baz@REALM
      ```

   b. Add the principal to the keytab file, `krb5kt`, on the workstation. This file is located in all of the following locations:

      - the `bin` directory beneath the directory where you installed Kerberos, for example, `C:\Program Files\MIT\Kerberos\bin`
      - the `%SystemRoot%` directory for example, `C:\Windows` or `C:\Winnt`
      - the `%systemRoot%` directory beneath your home directory, for example, `C:\Documents and Settings\cimtest\Windows`

2.4. Configuring Access Control

The Streaming Processor can be configured and started so that it restricts a user's access, even if the user successfully authenticates. For instance, the Streaming Processor server can be configured to deny the authenticated user the ability to query certain streams through the SQL interface or subscribe to streams through the subscription interface, or to stop the server.

2.4.1. Roles for Access Control

A role is the name of a group of users. The Sybase Aleri Streaming Platform takes its notion of role from the underlying operating system. In Linux and Solaris, a role is the same as a group.

To define a role for the Sybase Aleri Streaming Platform, simply create a new group and add the users to the group. An existing group on a machine can also be used as a role. Refer to the **groupadd** utility on Linux, or the `/etc/group` file to find more information on how to create and modify a group.

Note:

The authentication mechanism in the Sybase Aleri Streaming Platform release for Windows does not support access restriction by role.

2.4.2. Restrict Access

An Sybase Aleri Streaming Platform configuration file (in Aleri SQL or in XML) can be annotated with attributes that restrict access to streams and other elements of the data model.

Access control restrictions at the Streaming Processor level can restrict a user's ability to send commands to the server.

For example, the **sp_cli** utility can be used for querying the Streaming Processor for metadata, setting certain flags in the server, or shutting down the server. A user's access to these functions can be limited through annotations. The access to **sp_cli** can be restricted by role using the optional restrictAccess attribute in XML configurations, or the GRANT mechanism in Aleri SQL scripts.

Access control restrictions at the stream level can be used to restrict access to the data held within a stream. Users may be granted or denied the ability to query streams via the server, or to subscribe to certain streams. The access can be restricted to roles using the optional restrictAccess attribute in XML configurations, or the `GRANT` mechanism in Aleri SQL scripts.

Refer to the *Authoring Reference Manual* for more details on the restrictAccess attribute and the GRANT mechanism.

2.4.3. Start the Server with Access Control

By default, access control is turned off when the Streaming Processor is started via the **sp**, **sp-opt**, or **sp_server** commands. Without access control, any user who successfully authenticates can control the server, query any stream, or subscribe to any stream.

To enable the server to perform access control checks, use the flag `-r true` when starting the Streaming Processor with the **sp**, **sp-opt**, or **sp_server** commands.

2.5. Configuring Secure Socket Layer (SSL) Connections

The Sybase Aleri Streaming Platform supports SSL connections over the network to ensure the privacy of communication between the client applications and the Streaming Processor. SSL connections use cryptographic keys and certificates to implement secure data transfer.

2.5.1. Keys and Certificates

To start the Streaming Processor with support for SSL connections, use **sp** or **sp_server** with the `-e` option and the location of the directory containing the server private key file (with file name `server.key`) and certificate (with file name `server.crt`). Enter either of the following commands to start the Streaming Processor in a mode that supports SSL connections:

```
sp -e /home/aleri/etc/keys

sp_server -e /home/aleri/etc/keys
```

Refer to the *Utilities Guide* to see all available options for the **sp** and **sp_server** commands.

2.5.2. Generate Private Keys

The Sybase Aleri Streaming Platform distribution package comes with pre-generated SSL keys and certificates in the `etc/keys/` directory. The keys and certificates in this directory can be used for testing purposes, but probably should not be used in a production environment.

The Sybase Aleri Streaming Platform also comes with a shell script, `genkeys`, that creates the key and certificate files. Production users should use this script to generate files whose filenames follow the format `server.*`, and copy those files to the `/home/aleri/etc/keys/` directory. The script takes one argument: the number of days before the key and certificate expire.

To create the key and certificate files manually:

1. Create a self-signed certificate.

   ```
   openssl req -new -text -out server.req -days n
   ```

   where *`server.req`* is a filename for the certificate and *`n`* is the number of days the certificate will be valid. You will be prompted for a passphrase (at least four characters long) and other information (for example, company name and location). The challenge password can be left blank. The program will generate a key that is passphrase protected.

2. Remove the passphrase.

   ```
   openssl rsa -in privkey.pem -out server.key rm privkey.pem
   ```

   where *`server.key`* is a filename for the key files. Use the passphrase you entered in the previous step to unlock the existing key.

3. Turn the certificate into a self-signed certificate.

   ```
   openssl req x509 -in server.req -text -key server.key \
   -out server.crt -days n

   chmod og-rwx server.key
   ```

   Where *`server.key`* is the name of the key file and `n` is the number of days that the certificate will be valid.

4. Convert the `server.crt` file to a format that Java can import on the client (where `n` is the number of days that the certificate will be valid).

   ```
   openssl x509 -in server.crt -out server.crt.der -outform der \
   -days n

   chmod og-rwx server.crt.der
   ```

where *server.crt* is the file you created in the previous step, *server.crt.der* is the resulting file, and *n* is the number of days the certificate will be valid.

5. Copy the files `server.key`, `server.crt`, and `server.crt.der` to the key directory that is referenced when you start the Sybase Aleri Streaming Platform with the `-e` option.

2.5.3. Configure JDBC and Java Clients for SSL

The file `server.crt.der` must be imported into the Java environment in order for JDBC and Java clients to use SSL connections. To do this:

1. If the Java Client (for example, the Aleri Studio) is running on a remote machine, copy the `server.crt.der` file from the `/home/aleri/etc/keys` directory of the computer running the Streaming Processor to any location on the remote machine.

2. Import the certificate into the Java keystore.

```
export JRE=/home/aleri/authoring/eclipse/jre

$JRE/bin/keytool -keystore $JRE/lib/security/cacerts \
-alias certificate_name -import -file server.crt.der
```

   where *certificate_name* is the name of the certificate file.

   Enter the password for the *cacerts* keystore (the default is “changeit”).

   When prompted, enter **yes** to trust this certificate.

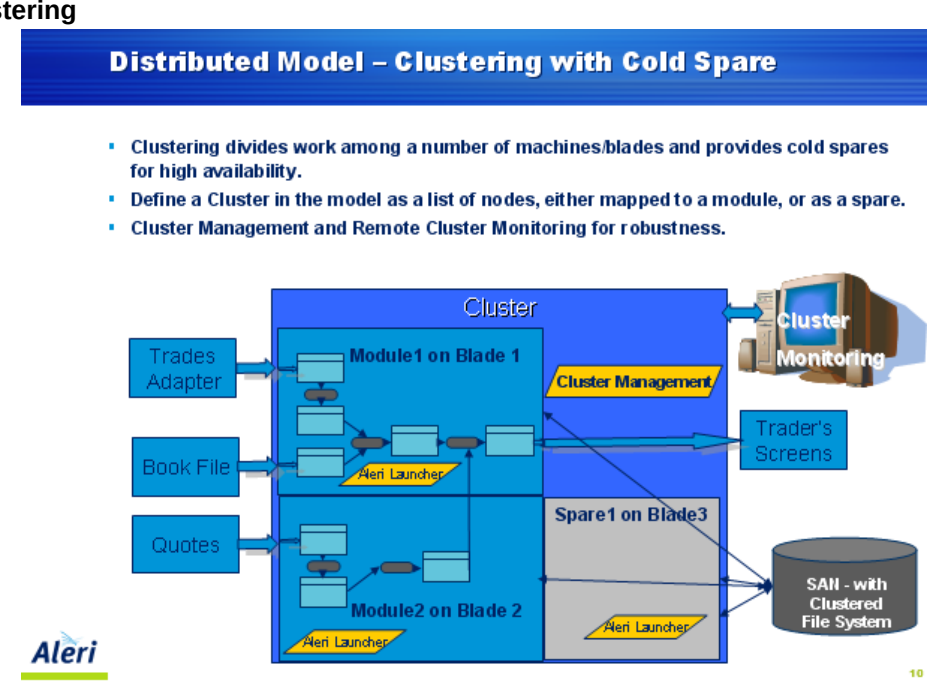
To use SSL in JDBC, add “?ssl” to the connection URL. For example,

```
jdbc:postgresql://localhost:22200/database?ssl
```

Chapter 3. Increasing Performance and Availability

The sections in this chapter provide information on modifying the basic Sybase Aleri Streaming Platform configuration to increase performance (by distributing the processing work among several servers and by performance tuning) or to minimize downtime (by setting up a Hot Spare server for the Streaming Processor).

3.1. Clustering



Clustering provides a way for the system administrator to divide processing work among a number of machines and makes it possible to set up “cold spare” servers that can take over processing if a Streaming Processor fails. In a clustering configuration, the data model is broken out into modules, and each module is mapped to a particular node (machine/port pair). See the *Authoring Reference* for more information on creating clustered models.

Each cluster appears in the configuration file as a list of nodes, with each node having an optional module associated with it. For example:

```
<Cluster id="network">
    <!-- default topology -->
    <Node machine="fc6-1" commandport="31415" module="streams" />
    <Node machine="fc6-2" commandport="31415" module="wholesale" />
    <Node machine="fc6-3" commandport="31415" module="cash" />

    <!-- Cold Spare nodes -->
    <Node machine="fc6-4" commandport="31415" />
    <Node machine="fc6-5" commandport="31415" />
</Cluster>
```

In the above example, the cluster has five nodes, and three modules, namely `streams`, `wholesale` and `cash`. The initial topology specified has the module `streams` which will run on node `fc6-1:31415`, module `wholesale` which will run on node `fc6-2:31415`, and module `cash`

which will run on node `fc6-3:31415`. The nodes `fc6-4:31415` and `fc6-5:31415` will initially remain idle, and will only be brought into active use in the cluster if one of the first three nodes fails.

Note:

The Sybase Aleri Streaming Platform debugging features (available from the command line or the **Breakpoints** panel in the Aleri Studio) do not work on a Sybase Aleri Streaming Platform installation that uses a clustered configuration.

3.1.1. Hardware Configuration and Requirements

The computers used in a clustered configuration do not have to be identical, but they all must have the same native word size (32 bit or 64 bit), and the same endianness. Mixing word sizes or little-endian with big-endian machines will produce an unusable cluster.

The Sybase Aleri Streaming Platform must be installed at the same location on each node in the cluster. Also, the Sybase Aleri Streaming Platform configuration file (that contains the data model) must be at the same location in the file system on each node of the cluster. This can be ensured using a simple NFS mount. If, for example, `/opt/aleri` is an NFS mount available to all nodes,

- the software can be installed in `/opt/aleri/install`
- `/opt/aleri/config/config.xml` could contain the configuration of the model
- `/opt/aleri/log` could be an empty directory in which to store the log files produced by each instance of the Streaming Processor.

The final configuration requirement is that an instance of the Aleri Launcher Daemon (**sp_ld**) be run on every node of the cluster. The **sp_ld** executable is part of the Sybase Aleri Streaming Platform installation. For each machine in the cluster, you should set up a system start up script (for example, `/etc/rc2.d/S99spld`) that will launch **sp_ld** as a daemon process each time the server is booted. **sp_ld** will also log messages (warning, errors and informational messages) using the syslog facility. and the `-p` *`port`* option allows you to specify the port to which the daemon will be bound; this port must be the same on each node of the cluster. See the *Utilities Guide* for more information concerning other command line options for the launcher daemon.

3.1.2. Cluster Management: Start, Stop and Retrieve the Status of Clusters

The **sp_clustermgr** application is used to start, stop and retrieve of the status for a running cluster. When you run this application (it does not require any arguments), it shows a prompt for further interactive commands.

sp_clustermgr supports the following command set:

```
show {commandLine, execPath, homeDir, logDir, ldPort}
set  {commandLine, execPath, homeDir, logDir, ldPort, debug} arg
status  <cluster name> <config file>
start   <cluster name> <config file>
stop    <cluster name> <config file> none
stop    <cluster name> <config file> pam  <username>
        <password>
stop    <cluster name> <config file> rsa <username>
        <private key file>
stop    <cluster name> <config file> gssapi <username>
```

3.1.2.1. Show and Set Commands

The **show** command, without any arguments, displays the value of all internal variables. To see the value of a particular variable (such as `commandLine`, `execPath`, `homeDir`, `logDir`, or `ldPort`) add the variable name as an argument.

The **set *<var name>* *<var value>*;** command sets the value of an internal variable within the cluster manager application.

3.1.2.2. Status Command

The **status *<cluster name>* *<config file>*** command requires the internal variable `ldPort` to be set. It scans all the nodes of the cluster and reports if the given node is up or down. If a node is up, it generates a report of modules that are running on the given nodes. Following is an example shows the status of the network:

```
./bin/sp_clustermgr
clustermgr> set ldPort 31420
clustermgr> status network config.xml
STATUS of cluster: network in config file: config.xml
  found the following modules: (cash,streams,wholesale)
* hosts:
   scanning host: (fc6-1) ... running modules: <>
   scanning host: (fc6-2) ... running modules: <>
   scanning host: (fc6-3) ... running modules: <>
   scanning host: (fc6-4) ... running modules: <>
   scanning host: (fc6-5) ... running modules: <>
clustermgr>
```

In this example, the cluster consists of three modules: `cash`, `streams` and `wholesale`, but no module is running on any of the nodes in the cluster.

3.1.2.3. Start Command

To start a cluster, you need the following information:

- The destination directory (a fully qualified directory name) that each module's log file will be written to. By default, all logfiles are named *<cluster name>*- *<module name>*.log.
- The fully qualified directory name of a directory to act as the home directory for the running instance of **sp** or **sp-opt**.
- The fully qualified path to use when launching each instance of **sp** or **sp-opt** on a given node.
- The port that each instance of **sp_ld** was started with.

for example:

```
clustermgr> set ldPort 31420
clustermgr> set homeDir /opt/aleri/install/run
clustermgr> set logDir /opt/aleri/log
clustermgr> set execPath opt/aleri/install/bin/sp
clustermgr> set commandLine -a user:pwd -d 7 -f /opt/aleri/run/config.xml
clustermgr> start network config.xml
```

Note a status command now returns:

```
clustermgr> status network config.xml
STATUS of cluster: network in config file: config.xml
found the following modules: (cash,streams,wholesale)
* hosts:
   scanning host: (fc6-1) ... running modules: <streams:31415>
   scanning host: (fc6-2) ... running modules: <wholesale:31415>
   scanning host: (fc6-3) ... running modules: <cash:31415>
   scanning host: (fc6-4) ... running modules: <>
   scanning host: (fc6-5) ... running modules: <>
```

3.1.2.4. Exit Command

The **exit *<cluster name>* *<config file>* *<user name>* *<password>*** command only requires that the internal variable `ldPort` is set. This utility issues an **exit** command to each running module, shutting down all members of the cluster.

3.1.3. Monitor a Clustered Configuration

The **sp_clustermon** application performs the real-time monitoring and maintenance of a cluster. This tool monitors all primary and spare nodes to keep the cluster in a fully functional state. It supports detection of failed modules and nodes, and automatic restart of modules.

An instance of **sp_clustermon** must be running for each cluster that is to be monitored. All command line arguments are required, and the syntax is:

```
sp_clustermon -p <sp_ld port> -C <cluster name> -f <config file>
```

> **Note:**
>
> Since every node in the cluster can potentially run any module defined in the data model, all Log Store files for the data model must be accessible and writable from each node. This is only feasible when running a clustered file system.

3.1.3.1. Monitor and Repair a Clustered Configuration in Real Time

Suppose the following cluster is running:

```
<Cluster id="network">
   <!-- default topology -->
   <Node machine="fc6-1" commandport="31415" module="cash"/>
   <Node machine="fc6-2" commandport="31415" module="wholesale"/>

   <!-- Cold Spare nodes -->
   <Node machine="fc6-3" commandport="31415"/>
   <Node machine="fc6-3" commandport="31416"/>
</Cluster>
```

Also suppose that all nodes in the cluster are alive and that there is an instance of **sp_ld** on each, running on port 31420, and that the cluster is running the data model `config.xml`. To retrieve the status for the cluster using **sp_clustermgr**, execute:

```
clustermgr> status network config.xml
STATUS of cluster: network in config file: config.xml
  found the following modules: (cash,wholesale)
* hosts:
   scanning host: (fc6-1.sybase.com) ...
      running modules: <cash:31415>
   scanning host: (fc6-2.sybase.com) ...
      running modules: <wholesale:31415>
   scanning host: (fc6-3.sybase.com) ...
      running modules: <>
```

The **sp_clustermon** may now be started on any machine that has network access to the nodes of the cluster:

```
sp_clustermon -p 31420 -C network -f /opt/aleri/run/config.xml
initial cluster configuration:
CLUSTER INFO:
   module: cash host:(fc6-1.sybase.com:31415) state:  Running
   module: wholesale host:(fc6-2.sybase.com:31415) state: Running
```

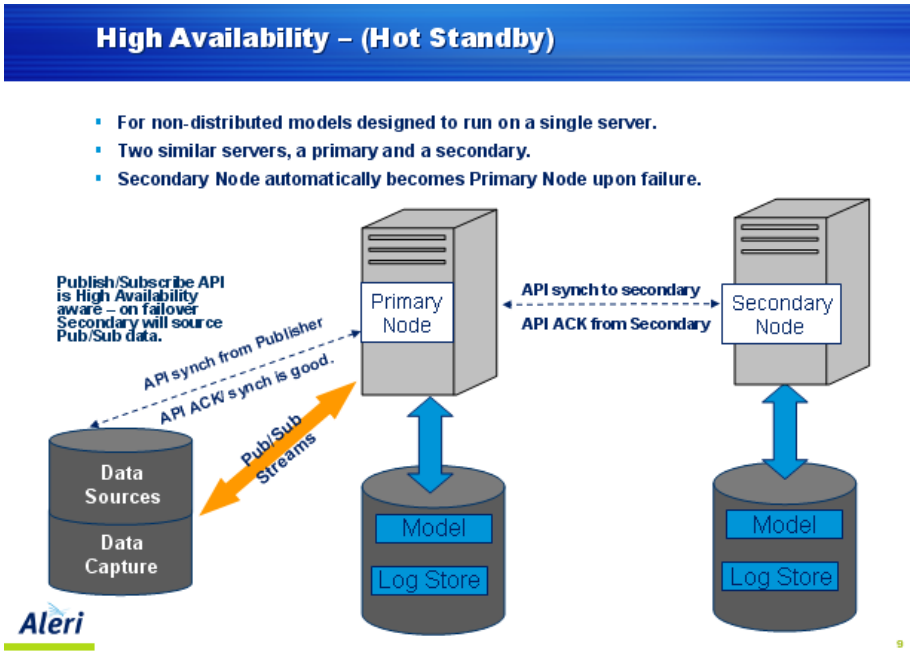
In this example, the cluster is healthy (the status of both modules is `Running`).

Now suppose the *`fc6-1`* node fails. The cluster monitor reports:

```
module: cash has failed, searching for a spare host....
   found host:(fc6-3.sybase.com:31415) attempting restart
   module
NEW MODULES started, broadcasting new topology to all live
members of the cluster:
   broadcasting topology change to module: cash on
      host:(fc6-3.sybase.com:31415)
      login attempt 0 failed will retry in 2 seconds.
      module: cash finished broadcasting new cluster topology.
   broadcasting topology change to module: wholesale on
      host:(fc6-2.sybase.com:31415)
      module: wholesale finished broadcasting new cluster topology.
Cluster repaired.
CLUSTER INFO:
  module: cash host:(fc6-3.sybase.com:31415) state: Running
  module: wholesale host:(fc6-2.sybase.com:31415) state: Running
```

In this example, the cluster monitor detects the failure, locates a cold spare node which is currently on-line, and restarts the failed module on the cold spare node. As a result, the cluster turns healthy again.

3.2. High Availability: Hot Spare Backups



Any data model that is designed to run on a single server (this excludes cluster configurations), may be run in a High Availability configuration, with a “Hot Spare” server. In this configuration, two machines are used. One is designated as the primary server, the other as a redundant secondary or Hot Spare. Data is constantly replicated from the primary to the secondary. If the primary server fails, the High Availability configuration detects the failure and promotes the secondary server to primary status with minimal interruption to clients.

All features of the Sybase Aleri Streaming Platform, including Sybase's publication/subscription APIs (for Java, C++, and .NET) provide full support for the High Availability configuration.

Ideally, to implement High Availability using the Sybase Aleri Streaming Platform, the primary and secondary should be identical machines (hardware, operating system, and, configured software). The use of identical machines is strongly recommended for better performance and ease of installation. If the two machines are not exactly the same, they must at least have the same native word size (both 32-bit or both 64-bit),, and the same endianness.

It is recommended that the directory holding the Streaming Processor installation be a Network File System (NFS) mounted on both the primary and secondary machines. This ensures that both machines have identical versions of the software. It is also recommended that the Sybase Aleri Streaming Platform configuration file that contains the data model reside in a directory that is cross-mounted on both machines; the Streaming Processor on the primary and secondary machines must be started with the same data model file.

Also, both machines must use the same authentication mechanism (PAM or RSA).

3.2.1. Example High Availability Configuration

In this example:

- There are two machines with DNS aware names *node1* (the initial primary) and *node2* (the initial secondary).
- The `/opt/aleri` directory hierarchy is available as an NFS mount on both the primary and secondary machines.

- The `/opt/aleri/install` directory contains the Sybase Aleri Streaming Platform installation.
- The `/opt/aleri/config/config.xml` file contains the data model.

To start the Sybase Aleri Streaming Platform in the High Availability configuration:

1. Log in to *`node1`* and start the Streaming Processor.

   ```
   cd /opt/aleri/install/bin

   sp-opt -V pam -a aleriusr:password -s -d 7 -c 31415 \
   -f /opt/aleri/config/config.xml
   ```

   Where the argument to `-a` specifies the login credentials of a valid account (shared on both the primary and secondary machines).

2. Log in to the secondary machine and start the Streaming Processor, using all the options used for the first machine, plus the `-H` option to specify the hostname and port of the primary machine.

   ```
   cd /opt/aleri/install/bin

   sp-opt -H node1:31415 -V pam -a aleriusr:password -s -d 7 \
   -c 31415 -f /opt/aleri/config/config.xml
   ```

   The **sp** and **sp_server** commands can also use the `-H` option to specify the host name and port of the primary machine.

   Both the primary and secondary instances of the Sybase Aleri Streaming Platform must use the same authentication mechanisms. The following table gives the appropriate command line arguments for each supported authentication mechanism.

Authentication Mechanism	sp/sp-Opt/sp-Server Command Line Arguments
none	-V none
PAM	-V pam -a <user>:<pass>
RSA	-V rsa -k <key directory> -K <user>: <private key file>

Clients that use the Pub/Sub Interface can be written to specify a pair of nodes. The interface will automatically publish to and receive subscriptions from the primary node, until the point when the primary machine fails. At that point, publication or subscription will immediately switch over to the secondary machine, and alerts will be triggered to the client applications, so that they are aware that the application has switched its source to the secondary. As soon as the primary fails, the secondary promotes itself to primary. See the *Guide to Programming Interfaces* for more information.

Note:

As the Streaming Processor runs, base data written to the primary is asynchronously transferred to the secondary in a very efficient manner. If the client requires a commit point, i.e., a guarantee that the data written to the primary has also been written to the secondary, the client must

make a call to synchronize Source Streams to the primary (there are Gateway, Command and Control, and Pub/Sub API commands to do this). When this call returns to the client, all base data will have been synchronized across both the primary and secondary, and committed to any Log Stores that may be used as the storage pool for Source Streams.

3.2.2. Restart a Failed Machine

In the above example, if `node1` fails, and all clients using the Aleri API transparently roll over to `node2`. After this happens, you can repair `node1`, bring it back on-line. Having done this, you can then restart the Sybase Aleri Streaming Platform on `node1`.

```
cd /opt/aleri/install/bin

sp-opt -H node2:31415 -V pam -a aleriusr:password -s -d 7 \
-c 31415 -f /opt/aleri/config/config.xml
```

After the restart, `node1` serves as the secondary and `node2` as the primary. This restores redundancy seamlessly; if `node2` fails, `node1` resumes operation as the primary.

3.3. Performance Tuning for the Sybase Aleri Streaming Platform

This section contains tips for monitoring and improving the performance of the Streaming Processor running on the Solaris 10 and Red Hat Enterprise Linux operating systems.

3.3.1. CPU Bottlenecks

3.3.1.1. Locale

The Sybase Aleri Streaming Platform performs locale-specific string comparisons. In the POSIX or C locales, the comparison operations are several times faster than in other locales (en_US.UTF-8 for example). For models with many string comparisons, running in the POSIX or C locale can yield significantly better performance.

The Sybase Aleri Streaming Platform checks the `LANG`, `LC_ALL`, and `LC_COLLATE` environment variables, and if performance is degraded due to non-POSIX specific string comparisons, it logs the following warning message:

```
Environment: LANG|LC_ALL|LC_COLLATE is set to current_value, causing
SLOW strcoll() behavior.
```

3.3.1.2. CPU Usage

The Streaming Processor automatically distributes its processing load across all the available CPUs on the machine. If the processing of a data stream seems slow, monitor each stream's CPU utilization using either the **sp_monitor** utility from the command line or the **Monitor** from the Aleri Studio. If the monitoring tool shows one stream in the model using the CPU more than other streams, you may want to refine the data model to ensure that the CPU is used evenly across the streams.

In addition to the CPU usage per stream as reported by the **sp_monitor** utility, the queue depth is also a very important metric to monitor. Each stream in the Sybase Aleri Streaming Platform is preceded by a queue of input records. All input to a given stream is placed in the input queue. If the stream processing logic cannot process the records as quickly as they are appearing in the input queue, the input queue can grow to a maximum size of 1,024 records. At that point, the queue stops accepting new records, which results in the automatic throttling of input streams. Since throttled streams require no CPU time, all CPU

resources are distributed to the streams with the full queues, in effect performing a CPU resource load balance of the running Stream Processor. When a stream's input queue is blocked, but the stream has managed to clear half of the pending records, the queue is unblocked, and input streams can proceed to supply the stream with more data.

If this load balancing inherent in the Sybase Aleri Streaming Platform is insufficient to clear the input queue for any given stream, the backup of the queue can percolate upward causing blockages all the way up the dependency graph to the source stream. If your monitoring indicates growing or full queues on any stream or arc of streams in the directed graph, this collection of streams should be examined closely for the cause of the slow processing.

3.3.2. Memory Usage

3.3.2.1. Main Memory Storage

There are no Streaming Processor configuration settings that directly set up or control RAM usage on the machine. However, the Streaming Processor reference counts records in the system, ensuring that at most one copy of a record is present in memory, although multiple references to that record may exist in different streams. Memory usage is directly proportional to the number of records stored in the model. To limit the amount of memory the entire instance of the Sybase Aleri Streaming Platform uses before it reports an out-of-memory condition, use the **ulimit** command to restrict the amount of memory available to each shell process.

3.3.2.2. Log Store Size

To optimize performance, ensure that the total size of all Log Store files does not exceed the size of the machine's available RAM. If it does, the machine resorts to paging and takes more time to process the data due to the large amount of disk I/O required. This causes all monitoring tools to display low CPU utilization for each stream, and standard UNIX commands such as **vmstat** to display high disk usage due to system paging.

3.3.2.3. Log Store File Locations

When storing Sybase Aleri Streaming Platform data locally using Log Stores, a high-speed storage device (for example, a raid array or SAN, preferably with a large dynamic RAM cache) is recommended. Putting the backing files for log stores on single disk drives (whether SAS, SCSI, IDE, or SATA) always yields moderately low throughput.

> **Note:**
>
> On Solaris systems, putting log files in `/tmp` uses main memory.

3.3.3. Network configuration

3.3.3.1. Nagle Algorithm

Nagle's algorithm is used to control congestion on Ethernet networks by reducing the amount of small packets that are exchanged between a server and a client. It also has the potential side effect of increasing network latency for those small packets.

The Nagle algorithm can interact with another TCP feature called a "delayed ACK" (typically 50-200ms per packet). This interaction can add significant delays to clients that are sending many very small messages to the Sybase Aleri Streaming Platform.

The preferred way to solve this problem is for the client to use the TCP_NODELAY option to disable the Nagle algorithm on the socket used to communicate with the Sybase Aleri Streaming Platform. The Pub/Sub library does this automatically.

On a Linux based server, using the TCP_NODELAY option from within a client is the only way to dis-

able the Nagle algorithm. There is no way to disable it for all clients.

On a Solaris based server, when dealing with a client that is exhibiting poor network performance, you can try disabling the Nagle algorithm at the operating system level using the following command:

```
ndd -set /dev/tcp tcp_naglim_def 1
```

This disables the Nagle Algorithm for all clients. If it provides no obvious benefit for your applications, re-enable it as it does have benefits for some workloads.

Changes made using **ndd** from the command line are not persistent across server restarts. To make the changes permanent, add the following command to the `/etc/system` file:

```
set ndd:tcp_naglim_def=1
```

3.3.3.2. TCP Buffer and Window Sizes

High throughput data transfers between clients and the Sybase Aleri Streaming Platform rely on the underlying operating system's TCP networking system being properly tuned. The data generated by clients for delivery to the Sybase Aleri Streaming Platform does not always arrive at a uniform rate. Sometimes the delivery of data is bursty. In order to accommodate large bursts of network data, large TCP buffers, and TCP send/receive windows are useful. They allow a certain amount of elasticity, so the operating system can temporarily handle the burst of data by quickly placing it in a buffer, before handing it off to the Streaming Processor for consumption.

If the TCP buffers are undersized, the client can see TCP blockages due to the advertised TCP window size going to zero as the TCP buffers on the Sybase Aleri Streaming Platform server fill up. To avoid this scenario, Sybase recommends tuning the TCP buffers and window sizes on the server on which the Sybase Aleri Streaming Platform is running to between one and two times the maximum size that is in use on all client servers sending data to the Sybase Aleri Streaming Platform.

To determine the current sizes of the TCP buffers and TCP windows on the Solaris 10 operating system:

```
/usr/sbin/ndd -get /dev/tcp tcp_max_buf
/usr/sbin/ndd -get /dev/tcp tcp_recv_hiwat
/usr/sbin/ndd -get /dev/tcp tcp_xmit_hiwat
```

To set these parameters on a Solaris system:

```
/usr/sbin/ndd -set /dev/tcp tcp_max_buf 16777216
/usr/sbin/ndd -set /dev/tcp tcp_recv_hiwat 8388608
/usr/sbin/ndd -set /dev/tcp tcp_xmit_hiwat 8388608
```

For SunFire V4xxx class hardware, the values shown are good maximum values, but they increase the overall memory footprint of a network application a small amount.

Changes made using **ndd** from the command line are not persistent across server restarts. To make the changes permanent, add the following commands to the `/etc/system` file:

```
set ndd:tcp_max_buff=16777216
set ndd:tcp_recv_hiwat=8388608
set ndd:tcp_xmit_hiwat=8388608
```

On a Linux (kernel 2.6) system, obtain the crucial network parameters with the following commands:

```
/sbin/sysctl -a | grep net.core.rmem_max
/sbin/sysctl -a | grep net.core.rmem_default
/sbin/sysctl -a | grep net.core.wmem_max
/sbin/sysctl -a | grep net.core.wmem_default
/sbin/sysctl -a | grep net.ipv4.tcp_mem
/sbin/sysctl -a | grep net.ipv4.tcp_rmem
/sbin/sysctl -a | grep net.ipv4.tcp_wmem
```

To set these parameters on a Linux system, add these lines to the `/etc/sysctl.conf` file:

```
net.core.rmem_max = 8388608
net.core.rmem_default = 8388608
net.core.wmem_max = 8388608
net.core.wmem_default = 8388608
net.ipv4.tcp_mem = 4096 8388608 16777216
net.ipv4.tcp_rmem = 4096 8388608 16777216
net.ipv4.tcp_wmem = 4096 8388608 16777216
```

and run the **`/sbin/sysctl -p`** command.

Chapter 4. Dynamic Service Modifications of the Sybase Aleri Streaming Platform

It is possible to change the data model of a running Sybase Aleri Streaming Platform, with less disturbance to its clients than a restart with a different configuration file would involve. In this type of modification, the existing streams preserve their connections and their handles whenever possible. The contents of each stream are regenerated after the change.

In a conventional relational database all the tables and views are interconnected, so any changes must be applied all at once to create a new usable model. For this reason, applications that access a common relational database must be stopped when the changes are applied, so that they never see a partially changed database.

However, the Sybase Aleri Streaming Platform can perform such modifications atomically while the Streaming Processor is running. That is, it applies the changes as a group: either all of the changes are applied successfully or none are. Processing is suspended while the set of changes is applied and the stream is regenerated, but the applications always see a wholly consistent configuration.

Unlike the common relational database modifications done with statements like ALTER TABLE, the Sybase Aleri Streaming Platform accepts the whole new configuration file. It then automatically finds and applies the differences from the running configuration. Normally, the changes of the tables and views are interconnected, and all need to be applied to create a new usable model. In the common relational databases the applications have to be stopped, so that the partially-changed database would not be seen.

Changes to the configuration of a data model can be introduced by creating a new version of the data model. This new configuration file may be created manually or through the SQL-to-XML conversion, as with any other configuration file. The new version of the data model can be applied to the Streaming Processor dynamically, without stopping it.

Note:

Sybase Aleri Streaming Platform installations configured for clustering and High Availability cannot be modified dynamically.

Note:

Sybase Aleri Streaming Platform cannot be modified dynamically while it is in the trace mode.

4.1. Performing Dynamic Service Modifications

To modify the Sybase Aleri Streaming Platform configuration dynamically:

1. Create the new configuration file (data model).
2. Initiate the loading of the new configuration with an XML RPC request to the Streaming Processor.

The **sp_cli** utility is the interface to use for this type of request. See *Guide to Programming Interfaces* for the details on XML RPC.

Which **sp_cli** command to use to load the configuration depends on whether the configuration file is located on the server.

4.1.1. Case 1: Changing a Configuration File

The following example describes the most typical (and recommended) case: placing the new configuration file on the server and using the **load_config**.

For this example, the Sybase Aleri Streaming Platform control port is "9100", the authorized user is "aleriusr" with password "password", and the new configuration file is `/home/aleri/models/trading/algorithmic2.xml`.

1. Start the **sp_cli** interactive shell.

```
sp_cli -c aleriusr:password -p 9100
```

   The system displays the `sp_cli>` prompt.

2. Load the new data model.

```
load_config {/home/aleri/models/trading/algorithmic2.xml}
```

Note

If you wish, you can enter the **load_config** command as an argument to your invocation of the **sp_cli** shell. In this case, the whole **load_config** command with its arguments must be enclosed in singe quotes.

```
sp_cli -c aleriusr:password -p 9100 \
'load_config {/home/aleri/models/trading/algorithmic2.xml}'
```

If the new configuration is invalid, **sp_cli** returns an error message. (Detailed error messages are written to the Sybase Aleri Streaming Platform log). If it is valid, information about the modification is logged.

If the Sybase Aleri Streaming Platform uses Log Stores, they are converted to contain the data from the new configuration file. You must specify the new configuration file on the next cold restart, to ensure correct operation with the modified Log Stores.

4.1.2. Case 2: New Configuration File on a Client Computer

If the new configuration file is located on an administration client machine which is different from the server (and has the **sp_cli** utility on it), use the **load_config_inline** command. This command allows you to "inline" the new configuration.

1. Create a file, `newconfig.cli`, that contains the command and the in-lined XML configuration.

```
load_config_inline {regen,nocompat,base} <<!
<?xml version="1.0" encoding="UTF-8"?>

<Platform>
```

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

    <SourceStream id="filterInput" store="store">
        <Column datatype="int32" key="true" name="a"/>
        <Column datatype="string" key="true" name="b"/>
        <Column datatype="double" key="true" name="c"/>
        <Column datatype="date" key="true" name="d"/>
        <Column datatype="int32" key="false" name="intData"/>
        <Column datatype="string" key="false" name="charData"/>
        <Column datatype="double" key="false" name="floatData"/>
        <Column datatype="date" key="false" name="dateData"/>
    </SourceStream>

    <FilterStream id="filter" istream="filterInput" store="store">
        <FilterExpression>
            filterInput.intData != 10
        </FilterExpression>
    </FilterStream>

</Platform>
!
```

In this example, the command line ends with `<<!`. The lines that follow it contain the configuration. The last line contains a `!` without any other characters. This syntax is similar to the shell syntax for specifying a termination word, but the termination word here cannot be changed (it is fixed to be `!`).

2. Invoke the **sp_cli** utility with the `-i` specifying the file containing the **load_config_inline** command and the new configuration.

```
sp_cli -c aleriusr:password -p server:9100 -i newconfig.cli
```

The **sp_cli** shell displays a different prompt when reading the inlined configuration than the one it displays when reading the commands.

4.1.2.1. Alternative: load_config_inline_conv

The **load_config_inline_conv** command is another **sp_cli** tool for modifying the Sybase Aleri Streaming Platform configuration. This command can be used to inline both the modified configuration file and the configuration file for the data conversion (instead of specifying it as a server file with the option **conv**). See more on data conversion issues in Section 4.6, “Modifications of Source Streams”.

Here is a simple example that could be placed into the file `newconfig.cli` and used as shown above.

```
load_config_inline_conv {regen,nocompat,base} <<!
<?xml version="1.0" encoding="UTF-8"?>

<Platform>

    <Store file="store" fullsize="20" id="store"/>
```

```
    <SourceStream id="filterInput" store="store" type="dynamic">
        <Column datatype="int64" key="true" name="a"/>
        <Column datatype="string" key="true" name="b"/>
        <Column datatype="double" key="true" name="c"/>
        <Column datatype="date" key="true" name="d"/>
        <Column datatype="int32" key="false" name="intData"/>
        <Column datatype="string" key="false" name="charData"/>
        <Column datatype="double" key="false" name="floatData"/>
        <Column datatype="date" key="false" name="dateData"/>
    </SourceStream>

    <FilterStream id="filter" istream="filterInput" store="store">
        <FilterExpression>filterInput.intData != 10</FilterExpression>
    </FilterStream>


</Platform>
!<<!
<?xml version="1.0" encoding="UTF-8"?>

<Platform>

    <Store id="store"/>

    <SourceStream convsrc="filterInput" id="input" store="store">
        <Column datatype="int32" key="true" name="a"/>
        <Column datatype="string" key="true" name="b"/>
        <Column datatype="double" key="true" name="c"/>
        <Column datatype="date" key="true" name="d"/>
        <Column datatype="int32" key="false" name="intData"/>
        <Column datatype="string" key="false" name="charData"/>
        <Column datatype="double" key="false" name="floatData"/>
        <Column datatype="date" key="false" name="dateData"/>
    </SourceStream>

    <FlexStream convdst="filterInput" id="compute" istream="input"
            store="store">
        <Column datatype="int64" key="true" name="a"/>
        <Column datatype="string" key="true" name="b"/>
        <Column datatype="double" key="true" name="c"/>
        <Column datatype="date" key="true" name="d"/>
        <Column datatype="int32" key="false" name="intData"/>
        <Column datatype="string" key="false" name="charData"/>
        <Column datatype="double" key="false" name="floatData"/>
        <Column datatype="date" key="false" name="dateData"/>
            <Method name="inputMethod1" stream="input">{
                output [a = cast(int64, input.a);
                    b = input.b;
                    c = input.c;
                    d = input.d; |
                    intData = input.intData;
                    charData = input.charData;
                    floatData = input.floatData;
                    dateData = input.dateData;];
            }</Method>
    </FlexStream>

</Platform>
!
```

The general syntax here is similar to that for **load_config_inline**, but in this case there are two in-lined arguments, separated by a line containing nothing but `!<<!` on it. The option **conv** may not be used

with the command **load_config_inline_conv**; the request for the conversion is implied in the command itself.

Note:

When using the commands **load_config_inline** and **load_config_inline_conv**, it becomes especially important that the new configuration file be placed on the server for the future Streaming Processor restart. If the new configuration modifies any log stores, the old configuration will not work correctly with the modified log stores.

The **sp_cli** also allows you to save the current configuration to a file, using **save_config** or **get_config** as shown:

```
save_config {configFile}
get_config [configFile]
```

save_config saves the configuration (its `configFile` argument refers to the file on the server machine where the Streaming Processor is running). **get_config** prints the configuration to standard output or saves it to a file on the machine where **sp_cli** is running.

In either of these commands, the filename for **save_config** must be enclosed in braces, and the specified file must not exist yet on the Streaming Processor. (This is a safeguard against overwriting some important file accidentally.)

4.2. The Internal Logic of a Dynamic Modification

Normally, the Sybase Aleri Streaming Platform performs a dynamic modification as follows:

1. The inputs to the Streaming Processor are suspended and the Streaming Processor is quiesced.
2. The Log Stores are backed up as they would be with the **sp_cli backup** command. See Section 1.4.2, “On-line Back-up” for details. Depending on the volume of data, this step may take a considerable amount of time.
3. The new configuration file is read, parsed and compared to the current configuration file. If there are any errors, the modification fails, and the Streaming Processor continues unchanged.
4. If a conversion model is specified (see details in Section 4.6, “Modifications of Source Streams”), the conversion configuration file is read, parsed, and executed. The resulting converted data is preserved for the regeneration of the Source Streams. If the conversion model can not be parsed, or fails to produce consistent data, the whole modification is rolled back and refused.
5. If the new configuration contains any new objects, they are created and compiled. If any of the existing streams are modified, the modifications are applied and compiled. This entire set of modifications is applied as a single transaction: if any one fails to compile, the whole modification is rolled back and refused.
6. Any objects (other than stores) from the old configuration that are not present in the new configuration are deleted. When streams are deleted, their subscribers get a `STREAM_EXIT` message, after which those subscriptions are ended (the client connections will stay).
7. The contents of the new and modified streams are regenerated. The subscribers to the existing streams see the regeneration as a WIPEOUT message followed by the usual sequence of `START_SYNC` message, inserts for the new contents, and `END_SYNC` message. If the new configuration moves some source streams to different stores, each one's contents are moved to the new store whenever possible (the structure of the stream does not change). This move of the source stream

contents is transparent to the subscribers. Depending on the complexity of the model and the volume of data, this regeneration step may take a considerable amount of time.

Note:

Streams with time-based Input Windows have an issue with data regeneration: since a time-based Input Window sees all records as being inserted at the time of regeneration, the whole contents of the stream's input stream will be inserted initially into the Input Window.

8. Any stores from the old configuration that are not present in the new configuration are deleted. The disk files for the deleted log stores are not deleted; they just aren't used any more. At this point, the conversion model (if any) is deleted as well.

9. The Streaming Processor resumes operations.

Note:

The dynamic modifications also invalidate any SQL queries registered through the Postgres SQL ODBC® interface.

4.3. Options

You can change the modification sequence described above using the options for the **load_config** command. This command's option string consists of options separated by commas (no spaces). An example is `regen,nobackup`). In this string, the options to be enabled are listed by their names; options to be disabled are prefixed with `no`. Therefore, `regen` to enable this option, `noregen` to disable it. Some options may have arguments specified after `=`. For example, `optname=value`). If an option takes multiple arguments, these arguments are also separated from each other by equal signs, as shown: `optname=arg1=arg2=arg3`. If an option is not specified explicitly, its default state is assumed (see below for defaults).

The XML RPC and the Sybase Aleri Streaming Platform API accept the option string in the same format as **sp_cli** (except for the surrounding braces or back quotes).

The following options are currently supported:

`[no]regen` — Perform/skip the regeneration of contents. If the regeneration is skipped, only new data will be processed according to the new data model; the old data will be left unchanged (and will thus be inconsistent with the new data). The modifications that can by made dynamically with the `noregen` option are limited.

`noregen` must never be used in production; it is intended only for quick and dirty experimentation with models in development. Since the Source Streams can not be regenerated, this option does not affect them. Depending on the size of the data, the regeneration may take a long time to complete. Default: `regen`.

`[no]backup` — Perform/skip the backup step. If the backup is skipped and a crash occurs in the middle of the dynamic modification, the data in the Log Stores may be left in an unrecoverable state. Default: `backup`.

`[no]compat` — Limit/don't limit the changes to the existing streams to only “compatible” changes — that is, changes that don't require that the stream be deleted and then re-created in the new form with a new stream handle. (See the detailed discussion in Section 4.4, “Compatible and Incompatible Modifications”). Note that the option allowing the `incompatible` changes is `nocompat`, not `in-`

`compat`. (Think of it as an abbreviation of “no compatibility check”.) The `noregen` and `nocompat` options are mutually exclusive. The default is `compat`.

`[no]source` — Allow/disallow the changes to the source streams. Since the data in source streams can't be regenerated, their contents may become inconsistent with the new configuration. The *`nobase`* option prevents accidental changes that could cause inconsistency. The meaning of this option depends on the state of the *`compat`* option: with *`compat,nobase`* no changes of any kind are allowed on source streams at all; with `nocompat,nobase` only the compatible changes are allowed on source streams. Default: `nobase`.

`[no]verbose` — Affects the error messages in the *`nocompat`* mode. With this option enabled, the system prints messages about any incompatible change, even if the change is acceptable. This can be confusing (since these messages look like errors), but also helpful in the diagnostics of dependent changes. Default: `noverbose`.

`conv=`*`remoteFile`* — Requires a conversion of the source stream data, and specifies that the data conversion model is in *`remoteFile`* on the server. The conversion model is a special temporary data model that reads the data from the source streams of the original model, processes it to match the schema of the new data model's source streams, and then places this data into the new data model's Source Streams before regeneration. A conversion model is typically used when a source stream's schema changes in an incompatible way. See the more detailed description in Section 4.6, “Modifications of Source Streams”.

Another way to require a conversion is to use the **sp_cli load_config_inline_conv** command and specify the conversion model in-line. Default: empty.

4.4. Compatible and Incompatible Modifications

Some of the objects in the Sybase Aleri Streaming Platform (Streams and Stores) have state connected to them. The state of a store is the data stored in it. The state of stream is the schema and stream handle.

In some cases an object can be modified without affecting the schema of this state. For example, changing the FilterExpression in a Filter Stream would affect which rows of data pass through the filter, but it would not change the schema of these rows. A subscriber application could still understand the data coming out of the stream, even though the data itself would be refreshed. In this case, subscriptions to this stream could be kept intact, and the stream handle could be kept unchanged. This type of modification is called “compatible”.

In the other cases the modifications are more drastic, involving substantial changes to the schema of the object. In fact, the new version may be an object of a completely different type, with everything different except the object name. This type of modification is called “incompatible”. An incompatible modification is processed by deleting the original object and then creating an entirely new object.

Incompatible changes are more disruptive. If an incompatible modification is dynamically applied to a stream object, the Streaming Processor sends an `End-of-stream` indication to all of that stream's subscribers, and then ends all subscriptions to it. The stream's handle becomes unknown; any future attempts to subscribe or publish to this handle will fail. A new handle is allocated to the new stream, so that the two streams cannot be confused by any client. To continue working with this stream, a client application must look up the stream by name, obtain its new handle and schema, and re-subscribe.

To reduce the chances of unexpected disruptions, the Sybase Aleri Streaming Platform provides the `compat` option to control whether such incompatible changes are allowed. If **load_config** uses the default value of this option, `compat`, only compatible changes are allowed; any incompatible modifica-

tions are rejected. A stream can still be deleted in the “compat” mode, but it will just not immediately be replaced with another stream of the same name. To allow incompatible changes, use the option “nocompat”.

In some cases, the schema of an object does not change much, but the change is such that modification in place is too technically difficult. For example, the movement of a stream to a different store. This type of modification is treated as incompatible: the old object is destroyed and a new version is created its place. In future releases of the Sybase Aleri Streaming Platform, this and other types of changes may be reclassified as `compatible`.

4.5. Renaming Objects

Normally, changing the name of an object (its *`id`* attribute) leads to the deletion of the old object and creation of a new one with a different name. In some cases, however, this may be undesirable so this change to a Stream would disrupt the existing subscriptions and change the stream's handle. To work around this problem, the Sybase Aleri Streaming Platform provides a less drastic way to rename the objects.

Renaming works only on streams and stores. Renaming of Connections would be not trivial. Renaming other objects makes no sense, since they have no state to keep anyway.

You can rename an object by including the “oldid” attribute in the new configuration file. For example, suppose that the old configuration file contained the stream "OldStream":

```
<FilterStream id="OldStream" store="store" istream="input">
    <FilterExpression>filterInput.intData = 10</FilterExpression>
</FilterStream>

<CopyStream id="Copy" store="store" istream="OldStream">
</CopyStream>
```

In the new configuration file, OldStream can be renamed to NewStream as follows:

```
<FilterStream id="NewStream" oldid="OldStream"
        store="store" istream="input">
    <FilterExpression>filterInput.intData = 10</FilterExpression>
</FilterStream>

<CopyStream id="Copy" store="store" istream="NewStream">
</CopyStream>
```

The `oldid` attribute provides the name of the old stream to match up with, so the Sybase Aleri Streaming Platform can change the name without changing the internals. In reality, the stream will still need to be recompiled and its contents regenerated (because its name has changed and any internal references to it need to be changed too). The streams that get the input from the renamed stream would need to be recompiled and regenerated. These streams must refer to the renamed stream by its new name, as shown in the example.

Renaming an object is a compatible change. A renamed stream keeps the same handle, and any ongoing subscriptions and publications continue uninterrupted.

Renaming can be combined with any other compatible changes. It can even be combined with incompatible changes. In this last case, the point of renaming gets lost, since the object gets deleted and re-created anyway.

4.6. Modifications of Source Streams

Source streams cannot be regenerated, since their data comes from outside the Sybase Aleri Streaming Platform. This makes the dynamic modification of a Source Stream problematic in two ways:

- If a source stream's conditional expression is changed, its contents may be inconsistent with the new condition. For example, if the Filter Expression in a source stream is dynamically changed to its opposite, the change affects future incoming data, but not the data that was collected before the dynamic modification. Another example would be the Input Window in a source stream. If the Input Window shortens, the extra data is discarded; but if the window is extended, there is no way to get back the data that has already been discarded.

- If the schema of a source stream changes, the old data may not fit it any more; if so, it is discarded. The Streaming Processor tries to preserve as much data as possible. If a stream is moved from one store to another, but is otherwise unchanged, its stored data is moved over with it. If only the names of the columns change, the data is kept as well. However, replacing a column with another column of the same type but different meaning can lead to unpredictable results. If the keys change, the Sybase Aleri Streaming Platform tries to move the data over, but if any duplicate keys are found, the data are discarded.

Therefore, the `base` in the **load_config** command must be explicitly specified to make dynamic changes to Source Streams. This is done to avoid corrupting the data by accident.

The effect of the `base` option depends on whether the option `compat` (default) or `nocompat` goes along with it. In the `compat` mode, no changes of any kind are allowed to the Source Streams without the `base` option, and only the compatible changes are allowed with it. If a disallowed change is attempted, it is refused, and the Sybase Aleri Streaming Platform continues with its configuration unchanged.

The `nocompat` mode allows any compatible modifications on the source streams (but still disallows the deletion of source streams). The `base` option is still required, however, for incompatible modifications or deletions of source streams.

A conversion model can be used to preserve the data in a source stream through complicated incompatible changes. Without a conversion model, the data would have to be downloaded from the Sybase Aleri Streaming Platform before the modification, processed to fit the new schema, and uploaded back after the modification. But this kind of processing is exactly what the Sybase Aleri Streaming Platform is designed to do; the conversion model mechanism allows you to harness this power.

When a conversion model is applied as part of a dynamic modification, the process goes as follows:

1. The conversion data model is instantiated and populated with the data from the source streams of the main data model. This connection is made by specifying the attribute `convsrc="sourceMainStream"` in the source streams of the conversion model. This attribute is not allowed for any other streams, or for the main data model.

2. The activity of the main data model is suspended while the conversion model runs and produces the results in its other streams. These results are saved.

3. The main data model is modified. Its source streams are loaded with the data produced by the conversion model.

4. The conversion model is discarded.

The connection of the conversion result streams to the source streams of the modified model is specified by the attribute `convdst="destinationMainStream"` in the streams of the conversion model. This attribute can not be used outside of the conversion model. The schema and keys of the stream that

has a `convsrc` or `convdst` attribute must be exactly the same as those of the stream to which this attribute refers (except possibly for the names of the schema and the columns).

For example, consider a small simple data model:

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

<Platform>

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

    <SourceStream id="filterInput" store="store">
        <Column datatype="int32" key="true" name="a"/>
        <Column datatype="string" key="true" name="b"/>
        <Column datatype="double" key="true" name="c"/>
        <Column datatype="date" key="true" name="d"/>
        <Column datatype="int32" key="false" name="intData"/>
        <Column datatype="string" key="false" name="charData"/>
        <Column datatype="double" key="false" name="floatData"/>
        <Column datatype="date" key="false" name="dateData"/>
    </SourceStream>

    <FilterStream id="filter" istream="filterInput" store="store">
        <FilterExpression>filterInput.intData = 10</FilterExpression>
    </FilterStream>

</Platform>
```

A data conversion model would be required to dynamically change the type of the column **a** from **int32** to **int64** while preserving the data. The modified data model would become:

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

<Platform>

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

    <SourceStream id="filterInput" store="store">
        <Column datatype="int64" key="true" name="a"/>
        <Column datatype="string" key="true" name="b"/>
        <Column datatype="double" key="true" name="c"/>
        <Column datatype="date" key="true" name="d"/>
        <Column datatype="int32" key="false" name="intData"/>
        <Column datatype="string" key="false" name="charData"/>
        <Column datatype="double" key="false" name="floatData"/>
        <Column datatype="date" key="false" name="dateData"/>
    </SourceStream>

    <FilterStream id="filter" istream="filterInput" store="store">
        <FilterExpression>filterInput.intData = 10</FilterExpression>
    </FilterStream>

</Platform>
```

To do the conversion, create another small model consisting of a source stream, which is required to get the data from the main model, and a ComputeStream that does the bulk of the work.

The conversion model is:

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

<Platform>

    <Store id="store" />

    <SourceStream convsrc="filterInput" id="input" store="store">
        <Column datatype="int32" key="true" name="a"/>
        <Column datatype="string" key="true" name="b"/>
        <Column datatype="double" key="true" name="c"/>
        <Column datatype="date" key="true" name="d"/>
        <Column datatype="int32" key="false" name="intData"/>
        <Column datatype="string" key="false" name="charData"/>
        <Column datatype="double" key="false" name="floatData"/>
        <Column datatype="date" key="false" name="dateData"/>
    </SourceStream>

    <ComputeStream convdst="filterInput" id="compute"
            permitKeyChange="true" istream="input" store="store">
        <ColumnExpression key="true" name="a">
            cast(int64, input.a)
        </ColumnExpression>
        <ColumnExpression key="true" name="b">
            input.b
        </ColumnExpression>
        <ColumnExpression key="true" name="c">
            input.c
        </ColumnExpression>
        <ColumnExpression key="true" name="d">
            input.d
        </ColumnExpression>
        <ColumnExpression key="false" name="intData">
            input.intData
        </ColumnExpression>
        <ColumnExpression key="false" name="charData">
            input.charData
        </ColumnExpression>
        <ColumnExpression key="false" name="floatData">
            input.floatData
        </ColumnExpression>
        <ColumnExpression key="false" name="dateData">
            input.dateData
        </ColumnExpression>
    </ComputeStream>

</Platform>
```

The source stream schema matches the original unmodified configuration of the stream **filterInput** (as specified in its `convsrc` attribute); the ComputeStream uses the new schema to match the new configuration of the stream **filterInput** (as specified in its convdst attribute).

Note that the value of the modified column in the ComputeStream's expression must be explicitly cast to the correct type, to make the column's type match the new model. Also, since the column modified is a key column, the attribute `permitKeyChange="true"` must be used on the ComputeStream.

The main data model and the conversion model have completely separate namespaces. The store named `store` used in the conversion model is completely separate from the store named `store` in the main data model. The same is true for any other object. The attributes convsrc and convdst are the only things that connect the two models.

The stores in the conversion model default to the memory type, no matter what is specified as the default for the main model. Log Stores are not supported for conversion models.

The conversion model can be loaded using **sp_cli**. It can be specified either as a file name in the `conv` option of the **load_config** or **load_config_inline** commands, or as an inline text argument for the **load_config_inline_conv** command.

If the conversion fails due to an error in the conversion model or because of a key conflict in the records it produces, the whole modification is rejected and rolled back.

The conversion model does not have to convert the data in all of the main data model's source streams. It may convert only some of them. Even if the source stream is unchanged, you can specify a conversion model to modify its data if necessary.

Note:

Even though this type of conversion is available, minimizing processing in the source streams themselves is a good principle of model design.

4.7. The Effect of Dynamic Service Modifications on Subscribers/Publishers

Dynamic modification of the Sybase Aleri Streaming Platform may affect existing subscribers and publishers. The only objects visible outside the Sybase Aleri Streaming Platform are streams; changes to the other objects (for example, Global) affect subscribers and publishers only through their effects on the streams.

The modification of a stream may affect publishers and subscribers in the following ways:

- If there is no change to the stream or to any of the streams upstream from it, there are no consequences for subscribers nor publishers.
- If a compatible change is made to a stream (see Section 4.4, “Compatible and Incompatible Modifications”), the stream's schema and handle stay the same but the stream's contents must be regenerated according to the new definition. In this case, subscribers are notified of regeneration by the message `WIPEOUT` to indicate that the old data must be discarded, then `START_SYNC`, the `INSERT`s with the new contents, and `END_SYNC`. See the *Guide to Programming Interfaces* for more information about these messages.
- If the dynamic modification does not change a particular stream but changes one “upstream”, that particular stream's schema and handle stay the same but the stream's contents must be regenerated according to the new definition. If this happens, the Sybase Aleri Streaming Platform notifies the subscribers of regeneration by sending the message `WIPEOUT` to indicate that the old data must be discarded, then `START_SYNC`, the `INSERT`s with the new contents, and `END_SYNC`. *Guide to Programming Interfaces*
- If a stream is deleted, an **EXIT** notification is sent to all the stream's subscribers, after which all subscriptions to this stream get closed. The deleted stream's handle cannot be referenced again, and cannot be assigned to another stream until the Streaming Processor restarts. Any attempt to subscribe or publish to this handle returns an error. All rows related to this stream in the Sybase Aleri Streaming Platform's metadata streams are deleted. The deletion of each stream is seen as a separate transaction in the metadata streams.
- If the dynamic modification creates a stream, the new stream is allocated a new handle, and the new stream's contents are generated. But this has no effect on subscribers (since the new stream has no subscribers yet). Metadata related to the new stream is inserted into the metadata streams. The creation of each stream is seen as a separate transaction in the metadata streams.

- Making an incompatible change is a logical equivalent of deleting a stream and then creating it from scratch. See Section 4.4, “Compatible and Incompatible Modifications” for details. This usually happens when a stream's schema changes, such that existing subscriptions and publishing won't be able to parse the new data correctly. If this happens, the stream **EXIT** notification is sent to all the stream's subscribers, after which all these subscriptions get closed. After the old incarnation of the stream is deleted, its handle cannot be assigned to a new stream until the Streaming Processor restarts. Any attempts to subscribe or publish to this handle returns an error. The new incarnation of the stream is created with a new handle. The stream's contents are regenerated (if possible) but no Subscribers are affected, since there are no subscribers to the new incarnation. The stream's old metadata is deleted from the metadata streams and the new metadata is inserted. The deletion and insertion of the metadata are processed as two separate transactions.

Any applications subscribing or publishing to the Streaming Processor must be prepared to adjust if the Streaming Processor's data model is modified dynamically. The applications may detect the dynamic changes by subscribing to the metadata streams (see the *Authoring Guide* for more detail). But for most practical purposes it's enough to monitor the usual subscription information and responses to the publishing of data.

An application publishing data to the Streaming Processor should watch for errors returned from the Streaming Processor. An error might indicate that the stream being published to was deleted or modified. If this happens, the application should:

1. Get the fresh list of streams from the Streaming Processor.
2. Locate its target stream in the list.
3. Obtain the new schema using the new stream handle.

After that, the application can continue publishing to the new incarnation of the stream. If the stream is not on the list any more, then it has been deleted, and the application should fail or otherwise handle the situation.

A subscriber application must also monitor the data it receives from the subscription. When it gets the `EXIT` notification, its reaction should be the same as that of the publisher described above: the subscriber should get the fresh list of streams, locate the stream and re-subscribe to it. The `WIPEOUT` notification means that the stream's contents are being regenerated. On receiving this notification, the application should wipe out its copy of the stream's contents and then refresh it from the regeneration data it receives between the `START_SYNC` and `END_SYNC` messages. The Streaming Processor sends this regeneration data whether the subscription was started with the `BASE` or `NOBASE` option. The application is free to ignore this regenerated data.

When an application establishes a connection to a stream, there is always a chance that the Sybase Aleri Streaming Platform configuration will be changed between the moment the subscriber receives the stream's metadata and the moment when the subscription is actually created. There are two ways to avoid this problem:

- A stream's handle (integer id) changes every time the stream's schema changes. The application can get the stream's schema via XML RPC using the stream's handle, and the subscription can be received using the handle as well. Doing this guarantees that the schema received by the subscriber matches the actual stream.
- The other way is simpler: get the information about the streams, do the subscriptions, and then get the information again. If the stream's definitions haven't changed in the second version, then the subscriptions are correct. Otherwise, close the subscriptions and start from scratch.

4.8. Idle Model

A Streaming Processor process cannot be started without a data model. If the current data model is dynamically modified to contain no Source Streams at all, the process will exit.

If you have a use for an idle Streaming Processor process, you can create a simple model that does essentially nothing. Here is an example:

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

<Platform>
    <Store id="Aleri_Idle_Store" kind="stateless"/>
    <SourceStream id="Aleri_Idle" insertOnly="true"
            store="Aleri_Idle_Store" >
        <Column name="placeholder" datatype="int32" key="true" />
    </SourceStream>
</Platform>
```

Or you can get the same effect by using the special file name `IDLE` (as is, uppercase, without any extension). The Sybase Aleri Streaming Platform recognizes it as a special name and supplies an idle model internally.

4.9. Modifications and their Effects

The following table summarizes the effects of the changes of individual attributes and subnodes on their parent and other nodes. The effects on the streams are listed as compatible (`compat`) or incompatible (`incompat`). See Section 4.4, “Compatible and Incompatible Modifications” for explanations of these terms.

Node	Modified attribute or subnode	Effect
Any Stream	no change	if any upstream stream is changed in any way, the data is regenerated (`WIPEOUT`, `SYNC_START`, inserts, `SYNC_END`)
	id, oldid	`compat`
	istream, refers to the same (renamed) streams by the new names	`compat`
	istream, refers to different streams	`incompat`
	Column or ColumnExpression or schema changes by inheritance, the keys, number and types of columns stay the same	`compat`
	Column or or ColumnExpression schema changes by inheritance, the keys or number or types of columns change	`incompat`
	store, refers to the same (renamed) store by a new name	`compat`
	store, refers to a different store	`incompat`
	ofile	`compat`
	type	`compat`
	restrictAccess	`compat`, the existing subscriptions aren't affected even if the new permissions don't allow the access for that user any more (unless the Stream receives another incompatible change)
	Input Window	`compat`, not allowed with option `noregen`
	Local	`compat`, not allowed with option `noregen`
	InConnection, or any change if the old stream had an InConnection	`incompat`
	OutConnection	`compat`
	any other change, except as listed below by stream types	`incompat`
AggregateStream	Group	`compat`, not allowed with option `noregen`
	GroupOrder	`compat`, not allowed with option `noregen`
	GroupFilter	`compat`, not allowed with option `noregen`
	ColumnExpression	`compat`, not allowed with option `noregen`

Node	Modified attribute or subnode	Effect
SourceStream	FilterExpression	`compat`
	insertOnly	`compat`
ComputeStream or ExtendStream	permitKeyChange	`compat`
FilterStream	FilterExpression	`compat`
FlexStream	Method	`compat`
	Timer	`compat`
JoinStream	Join	`compat`, not allowed with option `noregen`
PatternStream	Pattern	`compat`
Any Store	id, oldid	store gets renamed
	sweepamount	new value will be used for future cleaning of log stores, ignored for memory and stateless stores
	any other change, except as listed below by store types	all streams in it are marked `incompat`, new store created, data moved in Source Streams if possible, old stream discarded; the "file" attribute for Log Stores must refer to a not-yet-existing directory.
Store kind=*`memory`* or *`stateless`*	file	ignored
	fullsize	ignored
Global	any change	all the streams marked `compat`
StartUp	any change	all the streams marked `compat`

Some streams may also consider a change in the whitespace in the their definition as a compatible change.

When the new model has In/OutConnections defined in it, after performing the dynamic modification the connectors are restarted in the same way as during the normal start of Sybase Aleri Streaming Platform. Any connectors that have exited will be restarted. If some streams are not modified, their connectors will be left running. For the streams that are modified in any way, the connectors are stopped and restarted. The new connectors are started according to the new StartUp sequence (if any). Moreover, if a stream has any InConnections, any changes to that stream are treated as incompatible.

Some points worth reinforcing:

- When a stream's contents are regenerated, the stream loses its state and starts recollecting it during the regeneration. This includes the variables of computational streams and Input Windows.
- Some changes require a major recompilation of the stream's internals. This process may make these internals incompatible with the old contents. Such changes may not be done if the option `noregen` is specified. The `noregen` option is very dangerous and should only be used in very rare cases.
- The `noregen` and `nocompat` options are mutually exclusive. No incompatible changes may be done without regeneration.
- If the file attribute of a Log Store is changed, the new value must refer to a not-yet-existing directory. This is done to prevent the accidental data corruption.

- Never apply the dynamic changes in production without testing them first on a test instance of the Sybase Aleri Streaming Platform.

Appendix A. Administrative Best Practices

A.1. Log files/messages

In a production system, the Sybase Aleri Streaming Platform should always be run as a server (see the `-s` switch to **sp**). This forces log messages to be written to the UNIX® syslog facility. If you need to redirect the log messages to another location, a syslog replacement such as **slog** should be installed and configured to filter all Sybase Aleri Streaming Platform syslog messages and put them into its own log file (for example, `/var/log/Aleri`).

Running a production system in foreground mode (with logging to stderr), is not recommended, even if forced into the background in a job control shell (such as bash's `&`). This technique leaves the **sp** attached to a terminal, and subject to all terminal control signals, (such as **Ctrl**-**C**, **Ctrl**-**Z**, and **Ctrl**-**Y**); if the Streaming Processor writes to a terminal, even if redirected, it could be subject to blocking. Syslog (or a syslog replacement) can also typically be configured to take care of "rolling" the log file and limiting the file size.

Appendix B. Error Messages

The following is a list of error messages from the Sybase Aleri Streaming Platform, including those that provide status information.

100000 2(CRIT) Platform(): Out of memory at source file %s, line %d.

Action: Contact your system administrator to either increase the per-process memory usage limits or add physical memory to the machine.

100001 6(INFO) ----------

Action: This is a separator used to make the log easier to read.

100002 4(WARNING) Platform(): Error detected on file '%s': %s.

Action: The Aleri Streaming Platform has experienced an error when trying to read or write a file. Check whether the path exists and if the permissions are correct. If the path is relative, make sure that the Aleri Streaming Platform's current directory is correct.

100003 4(WARNING) Platform(): Requested operation on stream '%s', no such stream.

Action: An XMLRPC command (from sp_cli or from your custom binary) requested an operation on a non-existing stream. Make sure your command is correct.

100004 4(WARNING) Internal error: %s.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

100005 3(ERR) %s

Action: A general error message without an assigned error number. It should be self-explanatory.

100006 4(WARNING) Environment: %s is set to %s, causing SLOW strcoll() behavior.

Action: The check of LC_ALL, LC_COLLATE, and LANG yielded a non-posix locale collating sequence. this means the strcoll() function will run SLOW.

100007 7(DEBUG) Environment: %s is set to %s, causing FAST strcoll() behavior.

Action: The check of LC_ALL, LC_COLLATE, and LANG yielded a posix locale collating sequence. this means the strcoll() function will run FAST.

100008 7(DEBUG) LC_ALL, LC_COLLATE, and LANG are all unset, causing FAST strcoll() behavior.

Action: The values of LC_ALL, LC_COLLATE, and LANG are all NULL which is the same a a posix collating sequence, this means the strcoll() function will run FAST.

101000 2(CRIT) CommandControl(): could not get ephemeral port for xmlrpc server (abyss).

Action: An attempt to find an unused port number for XMLRPC service has failed. It is likely that there is a major issue in the machine's networking configuration. There is also the possibility of a rare failure if two instances of the Aleri Streaming Platform or a similar program are trying to allocate unused port numbers at the same time.

101002 6(INFO) CommandControl::execute(): starting xmlrpc server (abyss) with log file %s on port %d.

Action: An informational message about starting the XMLRPC service. Shows the port number that may have been specified with option -c or chosen automatically with the parameter -c 0. Lack of this message may mean that the option -c was not specified and the XMLRPC service was not started.

101003 6(INFO) CommandControl::execute(): server (abyss) exited.

Action: An informational message that the XMLRPC service has exited. It lets you track the stages when the Aleri Streaming Platform stops.

101004 3(ERR) CommandControl::authenticate(): invalid authentication token.

Action: A client tried to execute an XMLRPC request but presented an invalid authentication token. Possibly the server was restarted since this client has logged in, and it tries to present the old token to the new server. Restart the offending client.

101005 3(ERR) CommandControl()::<various>: could not access GatewayServer.

Action: An XMLRPC command (from sp_cli or from your custom binary) requested an operation that involves the gateway interface but the gateway is not available. This situation may happen if the Aleri Streaming Platform is being shut down.

101006 3(ERR) CommandControl::get_stream_definition(): Stream %s not found.

Action: An XMLRPC command (from sp_cli or from your custom binary) requested an operation on a non-existing stream. Make sure your command is correct.

101007 3(ERR) CommandControl::get_stream_definition(): a null stream name is invalid.

Action: A client has sent an XMLRPC request to execute an operation on a stream but provided no stream name. Probably a wrong stream name parameter was given to that client.

101008 7(DEBUG) CommandControl::<various>(): entering call (%s)

Action: An informational message about the start of processing of an XMLRPC call.

101009 7(DEBUG) CommandControl()::<various>: returning from call (%s).

Action: An informational message about the end of processing an XMLRPC call.

101010 6(INFO) CommandControl::execute(): wait on stream initialization before bringing up interfaces.

Action: Information about the internal stages of initialization.

101011 6(INFO) CommandControl::execute(): stream initialization complete, CnC & Gateway interfaces initializing.

Action: Information about the internal stages of initialization.

101012 2(CRIT) CommandServer()::load_config(): loading config from '%s', flags '%s'.

Action: Information about a received XMLRPC call for a dynamic modification.

101013 2(CRIT) CommandServer()::load_config(): missing arguments.

Action: An XMLRPC call for a dynamic modification that has some arguments missing.

101014 2(CRIT) CommandServer()::load_config(): new configuration contains incompatible changes.

Action: See the Administrator's Guide for the information on the Dynamic Services. You can either use the option "nocompat" to permit the application of incompatible changes or check your new model for the unintentionally introduced incompatible changes.

101015 2(CRIT) CommandServer()::load_config(): new configuration contains errors.

Action: The model that was requested to be loaded dynamically contains errors. Look at the preceding messages for the detailed information about the errors.

101016 2(CRIT) CommandServer()::load_config(): successfully loaded from '%s'.

Action: Informational message that the dynamic modification request has been successfully completed.

101017 2(CRIT) CommandServer()::load_config(): Dynamic configuration modifications are not supported in clustered mode.

Action: See the Administrator's Guide for the information on the Dynamic Services. The dynamic modifications may not be applied to an instance of Aleri Streaming Platform that is a part of a cluster.

101018 3(ERR) CommandControl::get_stream_handle_definition(): Stream handle %d not found.

Action: An XMLRPC command (from sp_cli or from your custom binary) requested an operation on a non-existing numeric stream id. This could happen due to an incorrect command or dynamic modification to the model. If a stream was dynamically modified in an incompatible way, the same stream name will become associated with a new numeric id. If a command has looked up the id by name before the dynamic modification and an operation was executed on this id after the modification has occurred, this error occurs. It should be treated as an intermittent error by the client program, with a retry of the lookup and execution of the operations.

101019 2(CRIT) CommandServer()::load_config(): Creating the data conversion platform.

Action: Information about stages of the dynamic service modifications.

101020 2(CRIT) CommandServer()::load_config(): Starting the data conversion platform.

Action: Information about stages of the dynamic service modifications.

101021 2(CRIT) CommandServer()::load_config(): The data conversion platform has completed successfully.

Action: Information about stages of the dynamic service modifications.

101022 2(CRIT) CommandServer()::load_config(): The data conversion platform has failed.

Action: The data conversion model specified for a dynamic modification is incorrect. See the preceding messages for the details of the errors.

101023 2(CRIT) CommandServer()::load_config(): The option 'conv' may not be used together with

an inlined conversion model.

Action: The data conversion model for a dynamic modification may be specified in one of two ways, inline or in a server-side file specified by the option 'conv'. But you cannot specify both at the same time so you must select one.

101024 6(INFO) CommandServer()::load_config(): Disposing of the data conversion platform.

Action: Information about stages of the dynamic service modifications.

101025 6(INFO) CommandServer()::load_config(): Disposing of the temporary platform instance.

Action: During the dynamic service modification, the new model is loaded into a temporary instance of the Aleri Streaming Platform, within the same Aleri Streaming Platform process. This instance is then analyzed for differences from the currently running model, and the modifications are applied to the currently running model. After modification is finished, the temporary instance is disposed.

101026 6(INFO) CommandServer()::kill_client(): Killed client %u on user request.

Action: The administrator has terminated a client connection through the XMLRPC interface command.

101027 6(INFO) CommandServer()::kill_client(): User request to kill the client %u, client not found.

Action: The administrator has requested to terminate a non-existing client connection. Possibly the client id argument was mistyped or the client had already disconnected by itself.

101028 6(INFO) CommandServer()::run_control(): Unknown subcommand '%s'.

Action: The XMLRPC command run_control has a number of subcommands identifying the exact action. The Pubsub API takes care of wrapping the subcommands into the individual calls. This error may mean that a new client program is trying to execute the new commands in an old version of the Aleri Streaming Platform or a general corruption in the client program.

101029 6(INFO) CommandServer()::run_control(): subcmd '%s' arg '%s'.

Action: Informational message about the subcommand of an XMLRPC call. The XMLRPC command run_control has a number of subcommands identifying the exact action.

101030 3(ERR) CommandServer(): Could not parse XMLRPC call args: %d: %s.

Action: An XMLRPC request with unparseable arguments was received. This error may mean that a new client program is trying to execute the incompatible commands in an old version of the platform, or a general corruption in the client program.

101031 2(CRIT) CommandServer(): IMMEDIATE EXIT REQUESTED, EXITING

Action: Informational message the the XMLRPC command "immediate exit" was received and executed. The Aleri Streaming Platform exits abruptly after this message.

101032 6(INFO) CommandControl::execute(): waiting for the GatewayServer to exit.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

101033 6(INFO) CommandControl::execute(): the GatewayServer exited.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

101034 3(ERR) CommandControl()::<various>: could not access Kerberos Server.

Action: You should either use a different authentication method or ask the system administrator to configure Kerberos. Contact your system administrator.

101035 3(ERR) CommandControl()::execute: could not initialize the SASL library.

Action: This message appears when either Kerberos is not configured, the service "sp" is not registered or the Kerberos library has experienced some other error.

101036 3(ERR) CommandControl()::checkCredentialsPAM, trying to login via PAM, but server not using PAM authentication.

Action: The client is trying to use a different kind of authentication than what is configured for the server. Check the server and client arguments and use the same authentication method for both.

101037 3(ERR) CommandControl()::checkCredentialsRSA, trying to login via RSA, but server not using RSA authentication.

Action: The client is trying to use a different kind of authentication than what is configured for the server. Check the server and client arguments and use the same authentication method for both.

101038 3(ERR) CommandControl()::checkCredentialsKERBV5, trying to login via SASL/Kerberos V5, but server not using SASL/Kerberos V5 authentication.

Action: The client is trying to use a different kind of authentication than what is configured for the server. Check the server and client arguments and use the same authentication method for both.

101039 3(ERR) CommandControl()::login[_key,_kerberos]()], Authentication rejected for user (%s).

Action: A client tried to log into XMLRPC service with a wrong password or unknown RSA key. If you believe that the password is correct and PAM authentication method is used, contact your system administrator to check the PAM configuration. If you believe that the RSA key is correct and the RSA authentication is used, check whether the matching public key is installed in the server's RSA public key directory. Make sure that the client uses the secret key file and the server has the public key file, not the other way around. See the Administrator's Guide for details.

101040 5(NOTICE) CommandControl()::login[_key,_kerberos]()], Authentication successful for user (%s).

Action: A user has logged in successfully.

101041 3(ERR) CommandControl::authenticate(): attempt at a control call from user without permissions.

Action: An user without administrative privileges has tried to execute an Aleri Streaming Platform control request. Check the user name, and either grant the control role or use a different name. The control role may be granted using the attribute restrictAccess of the <Platform> node.

102002 2(CRIT) RowInternal::pack() critical error.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

104000 3(ERR) Primop: Primitive operation '%s' not yet implemented.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

104001 4(WARNING) Function '%s': Calendar file %s not found.

Action: A non-existing calendar file was given as an argument to a calendar function. Check whether the path exists and whether the permissions are correct. If the path is relative, check that the Aleri Streaming Platform's current directory is correct.

104002 4(WARNING) Function 'business': Illegal offset of 0.

Action: Check your expressions. Probably an illegal argument was passed to a function.

104003 3(ERR) Error in function '%s': %s

Action: Check your expressions. Probably an illegal argument was passed to a function.

107000 3(ERR) %s(%s)::safeRead(): Read error %s.

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

107001 7(DEBUG) %s(%s)::processData(): Starting FileReader.

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

107002 2(CRIT) %s(%s)::processData(): Record has zero size.

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

107003 2(CRIT) %s(%s)::processData(): Short read on record.

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

107004 4(WARNING) %s(%s)::execute() record is for a different stream.

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

107005 2(CRIT) %s(%s)::execute() error in reading record.

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

107006 7(DEBUG) %s(%s)::execute(): Exiting FileReader.

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

107007 4(WARNING) %s(%s): out of memory allocating a record of size %d, skipping this record.

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

107008 2(CRIT) Filereader(fd=%d) %s(%s): got unknown stream handle %d, stopped reading

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

107009 2(CRIT) %s(%s)::processData(): Record of %u bytes is too large, probably a wrong byte order.

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

107010 2(CRIT) %s(%s)::processData(): A record is corrupted.

Action: Message from the File Reader that is used internally in the Aleri test infrastructure. Should not occur in the normal customer usage.

108000 7(DEBUG) GatewayClient(:%d)::safeIO::execute(): %s.

Action: Information about a client disconnecting from the gateway. The message describes the exact reason. It may be a normal close, an abruptly closed socket causing an exception or a protocol error where a subscriber client tries to send more commands to the gateway interface.

108001 4(WARNING) GatewayClient(%d:%d)::~GatewayClient() destroyed (auto).

Action: Information about a client being fully disconnected from the Aleri Streaming Platform gateway and all information about this client being disposed.

108002 5(NOTICE) GatewayClient(%d:%d)::execute() Client successfully authenticated: mode (%s), user (%s).

Action: Informational message about a successful login to the gateway service.

108003 3(ERR) GatewayClient(%d:%d)::execute() Client failed authentication: mode (%s), user (%s).

Action: A client tried to log in to the gateway service with a wrong password or unknown RSA key. If you believe that the password is correct, and PAM authentication method is used, contact your system administrator to check the PAM configuration. If you believe that the RSA key is correct, and the RSA authentication is used, check whether the matching public key is installed in the server's RSA public key directory. Make sure that the client uses the secret key file and the server has the public key file, not the other way around. See the Administrator's Guide for details.

108004 6(INFO) GatewayClient(%d:%d)::execute() Switching to secure socket.

Action: Informational message that the client has requested the SSL encryption on the gateway connection, and the Aleri Streaming Platform is configured to support it.

108005 6(INFO) GatewayClient(%d:%d)::execute() Server could not give client a secure connection.

Action: The client has requested the SSL encryption on the gateway connection but the Aleri Streaming Platform is not configured to support it. Either configure the Aleri Streaming Platform for the SSL support or change the client to not request it.

108006 3(ERR) GatewayClient(%d:%d)::execute() First command must be AUTHENTICATE_CLIENT.

Action: The client program hasn't followed the correct protocol. If it happens with an Aleri-provided program, contact the Aleri support. Otherwise check your program.

108007 6(INFO) GatewayClient(%d:%d)::execute() REQUEST_SSL_MODE must be the first request from client.

Action: The client program hasn't followed the correct protocol. If it happens with an Aleri-provided program, contact the Aleri support. Otherwise check your program.

108008 4(WARNING) GatewayClient(%d:%d)::execute() Client closed/dropped connection.

Action: Information about a client disconnecting from the gateway.

108009 4(WARNING) GatewayClient(%d:%d)::%s Caught exception: %s

Action: An informational message that may happen on certain socket errors. Probably, the client has dropped the connection at an unexpected time. It's also possible that the client program has crashed.

108010 7(DEBUG) GatewayClient(%d:%d)::execute() SUBSCRIBE_LOSSY

Action: Informational message that a client has requested the lossy subscription mode.

108011 7(DEBUG) GatewayClient(%d:%d)::execute() SUBSCRIBE_NOBASE

Action: Informational message that a client has requested the subscription without receiving the base contents of the stream.

108012 7(DEBUG) GatewayClient(%d:%d)::execute() Exiting.

Action: Informational message that a per-client gateway thread in the Aleri Streaming Platform has exited.

108013 7(DEBUG) GatewayClient(%d:%d)::S_processMessage() Error exporting record (transactional).

Action: The gateway has failed to write a transaction data to the socket, and it's likely that the client has dropped the connection in the meantime.

108014 7(DEBUG) GatewayClient(%d:%d)::S_processMessage() Error exporting record (base).

Action: The gateway has failed to write the stream base data record to the socket, and it's likely the client has dropped the connection in the meantime.

108015 4(WARNING) GatewayClient(%d:%d)::S_processMessage() Got an unknown message type.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

108016 7(DEBUG) GatewayClient(%d:%d)::S_:postWithQueueCheck() dropping data.

Action: Client has subscribed in a lossy mode, but is not receiving fast enough to keep up with the incoming data.

108017 4(WARNING) GatewayClient(%d:%d)::S_postWithQueueCheck() resuming, dropped since

last resume (%d), total dropped (%d).

Action: Client has subscribed in a lossy mode and caught up with receiving the data after some was dropped.

108018 7(DEBUG) GatewayClient(%d:%d)::S_postWithQueueCheck() fill %d percent.

Action: Informational message about the fill level of the client's output queue. The high level of fill shows that the client is not keeping up with the data rate and soon may either block the Aleri Streaming Platform (with the normal subscription mode) or have its data dropped (with the lossy subscription mode) or connection dropped (with the droppable subscription mode).

108019 4(WARNING) GatewayClient(%d:%d)::P_doRecord() received record for unknown stream. Index: %d.

Action: There is something wrong with either the client program or the data it's trying to upload. Possibly the binary data was converted for a different model or machine with a different byte order (such as Intel vs Sun SPARC) or the data file was simply corrupted.

108020 4(WARNING) GatewayClient(%d:%d)::G_getCommand() failed to malloc buffer during authentication or encryption, aborting.

Action: Contact your system administrator to either increase the per-process memory usage limits or add physical memory to the machine.

108021 4(WARNING) GatewayClient(%d:%d)::%s failed to malloc buffer during record command, size %d.

Action: Contact your system administrator to either increase the per-process memory usage limits or add physical memory to the machine.

108022 3(ERR) GatewayClient(%d:%d)::%s detected large, possibly bad message from client of size %d

Action: There is something wrong with either the client program or the data it's trying to upload. Possibly the binary data was converted for a different model or machine with a different byte order (such as Intel vs Sun SPARC) or the data file was simply corrupted.

108023 7(DEBUG) GatewayClient(%d:%d)::execute() SUBSCRIBE_DISTRIBUTED

Action: Informational message showing that this client is another instance of the Aleri Streaming Platform that is a part of the same cluster.

108024 4(WARNING) GatewayClient(%d:%d)::S_registerSqlStream() could not parse SQL statement {%s}.

Action: The SQL statement given in an SQL subscription is incorrect. Check its syntax and names of the streams and fields it refers to.

108025 4(WARNING) GatewayClient(%d:%d)::S_registerSqlStream() could not resolve input stream for SQL statement {%s}.

Action: The SQL statement given in an SQL subscription is incorrect. Check its syntax and names of the streams and fields it refers to.

108026 4(WARNING) GatewayClient(%d:%d)::S_registerSqlStream() could not compile the SQL

statement {%s}, error:{%s}.

Action: The SQL statement given in an SQL subscription is incorrect. Check its syntax and names of the streams and fields it refers to.

108027 4(WARNING) GatewayClient(%d:%d): %s access control denied, stream '%s'

Action: The client doesn't have permission to connect to the Aleri Streaming Platform. Check the user name and then either grant the connect role or use a different user name.

108028 6(INFO) GatewayClient(%d:%d)::execute() Secure socket handshake failed, see preceding messages for details.

Action: The SSL protocol error. Check the client program call. Possibly the option enabling SSL was missed so it is trying to connect with a plain-text protocol to a secure port.

108029 7(DEBUG) GatewayClient(%d:%d)::execute() SUBSCRIBE_DROPPABLE

Action: Informational message that a client has requested the subscription that may be dropped (disconnected) if the client doesn't keep up with the data rate.

108030 4(WARNING) GatewayClient(%d:%d)::S_postWithQueueCheck() client cannot keep up, queue full, dropping client connection.

Action: The client has subscribed with the droppable option and could not keep up with the flow of data.

108031 4(WARNING) GatewayClient(%d:%d)::P_doRecord() caught exception reading record of size: %d

Action: An informational message that may happen on certain socket errors. Probably, the client has dropped the connection at an unexpected time. It's also possible that the client program has crashed.

108032 4(WARNING) GatewayClient(%d:%d)::G_getCommand failed, returned error code: %d

Action: There is something wrong with either the client program or the data it's trying to upload. Possibly the binary data was converted for a different model or machine with a different byte order (such as Intel vs Sun SPARC) or the data file was simply corrupted.

108033 7(DEBUG) GatewayClient(%d:%d)::execute() SUBSCRIBE_PULSED

Action: Informational message that a client has requested the subscription in the pulse mode. In this mode, the data is sent periodically with the multiple modifications of the same record during the period collapsed into one.

108036 6(INFO) GatewayClient(%d:%d)::S_processSubscribeQueue() stream %d pulsing %d records to client.

Action: Informational message that the pulse update to this client is happening now.

108038 4(WARNING) GatewayClient(%d:%d)::~GatewayClient() lossy subscribe: dropped since last resume (%d), total dropped (%d).

Action: The statistics of the dropped data at the end of a lossy subscription.

108040 4(WARNING) GatewayClient(%d:%d)::P_doRecord() - max(size(block)) changed from %d to %d.

Action: The statistics about the highest block size received so far from this client.

108041 7(DEBUG) GatewayClient(%d:%d)::execute() SUBSCRIBE_PRESERVE_BLOCKS

Action: Informational message that a client has requested the subscription with preservation of the transaction boundaries.

108042 3(ERR) GatewayClient(%d:%d)::initFinalizer() could not parse finalizer statement '%s'; error %s

Action: The SQL statement given in a finalizer is incorrect. Check its syntax and names of the streams and fields referred to.

108043 3(ERR) GatewayClient(%d:%d)::initFinalizer() access control denied for %sstream '%s'

Action: The SQL statement given in a finalizer is incorrect. Check its syntax and names of the streams and fields referred to.

108044 3(ERR) GatewayClient(%d:%d)::initFinalizer() finalizer references non-source stream '%s'

Action: The SQL statement given in a finalizer is incorrect. Check its syntax and names of the streams and fields referred to.

108045 6(INFO) GatewayClient(%d:%d) Finalizer succeeded: %s

Action: Informational message that the finalizer registered by this client had successfully executed.

108046 3(ERR) GatewayClient(%d:%d) Finalizer() failed with run-time code %d: %s

Action: The SQL statement given in a finalizer is incorrect. Check its syntax and names of the streams and fields referred to. Another reason could be that the Aleri Streaming Platform was already shut down when the finalizer attempted to run.

108047 6(INFO) GatewayClient(%d:%d)::S_registerSqlStream() SQL statement {%s}.

Action: Informational message reporting the statement used by this client for an SQL subscription.

108048 6(INFO) GatewayClient(%d:%d)::G_tryAuthenticationKERBV5() Client successfully authenticated.

Action: Informational message that a client has successfully logged in using the Kerberos authentication.

108049 4(WARNING) GatewayClient(%d:%d)::G_tryAuthenticationKERBV5() generated a fault, error message: %s

Action: A client has failed to authenticate with Kerberos. The error message contains the more detailed reason. Probably Kerberos is not properly configured. Consult your system administrator.

108050 4(WARNING) GatewayClient(%d:%d)::G_tryAuthenticationKERBV5() failed, error message: %s

Action: A client failed to authenticate through Kerberos. Check that the client's Ker-

beros ticket has not expired.

108051 3(ERR) GatewayClient(%d:%d)::G_tryAuthenticationPAM(), trying to login via PAM, but server not using PAM authentication.

Action: The client is trying to use a different kind of authentication than what is configured for the server. Check the server and client arguments and use the same authentication method for both.

108052 3(ERR) GatewayClient(%d:%d)::G_tryAuthenticationRSA(), trying to login via RSA, but server not using PAM authentication.

Action: The client is trying to use a different kind of authentication than what is configured for the server. Check the server and client arguments and use the same authentication method for both.

108053 3(ERR) GatewayClient(%d:%d)::G_tryAuthenticationKERBV5(), trying to login via SASL/Kerberos V5, but server not using SASL/Kerberos V5 authentication.

Action: The client is trying to use a different kind of authentication than what is configured for the server. Check the server and client arguments and use the same authentication method for both.

108054 6(INFO) GatewayClient(%d:%d)::G_setTerminator(), will exit without heartbeat of %d ms.

Action: Informational message that a client has requested the Aleri Streaming Platform to exit if the client stops generating the heartbeat or the connection to this client ever gets dropped for any reason.

108055 6(INFO) GatewayClient(%d:%d)::S_setQueueMax(), setting queue depth to %d rows.

Action: Informational message that a client has requested its gateway queue size limit set to the reported value.

108056 6(INFO) GatewayClient(%d:%d)::S_setQueueMax(), argument of %d rows is less than minimum value of 1000.

Action: Informational message that a client has requested its gateway queue size limit set to a value lower than the lowest one supported.

108057 4(WARNING) GatewayClient(%d:%d)::execute(), ignoring request for shine through, a SQL subscription is incompatible with shine through.

Action: Informational message that a client has requested the shine-through option on an unsupported SQL subscription.

108058 5(NOTICE) GatewayClient(%d:%d)::execute(), setting shine through optimization on subscription.

Action: A Notice that a client has requested the shine-through option on subscription to reduce the volume of data.

108059 6(INFO) GatewayClient(%d:%d)::execute(), setting EXIT on a dropped connection.

Action: Informational message that a client has requested the Aleri Streaming Platform to exit if the connection to this client ever gets dropped by any reason.

108060 4(WARNING) GatewayClient::S_post(%d:%d) queue at 99%% of limit.

Action: Informational message about the fill level of the client's output queue. The high level of fill shows that the client is not keeping up well with the data rate and soon may either block the Aleri Streaming Platform (with the normal subscription mode), have its data dropped (with the lossy subscription mode) or have its connection dropped (with the droppable subscription mode). This message means that the queue has gotten very close to being full and the stoppage or dropping is imminent.

108061 4(WARNING) GatewayClient::S_post(%d:%d) queue dropped below 51%% of limit.

Action: Informational message about the fill level of the client's output queue. It shows that the client has restarted the consumption of data, and the queue has drained from completely filled to less than half-filled.

108062 4(WARNING) GatewayClient::GatewayClient(%d:%d) host:[%s] has initiated a connection.

Action: Informational message about the origin of a newly received gateway connection.

108063 2(CRIT) GatewayClient::postToSubscriberQueue(%d:%d) time to drain base data %d ms exceeded limit %d ms, dropping client connection.

Action: The client took too long to read the base contents of the stream. The Aleri Streaming Platform has stopped sending data to this client to avoid blocking the other clients. If the stream contents is large, you may want to increase the time limit that can be set individually by each client.

108064 6(INFO) GatewayClient(%d:%d)::S_setBaseDrainTime(), setting base drain time from %d ms to %d ms.

Action: Informational message that a client has set an alternative time limit for receiving the stream base contents.

108065 3(ERR) GatewayClient(%d:%d)::P_doRecord() received an envelope with a record offset beyond its size (record %u/%u).

Action: There is something wrong with either the client program or the data it's trying to upload. Possibly the binary data was converted for a different model or machine with a different byte order (such as Intel vs Sun SPARC) or the data file was simply corrupted.

108066 3(ERR) GatewayClient(%d:%d)::P_doRecord() received a transaction with a record offset beyond its size (record %u/%u).

Action: There is something wrong with either the client program or the data it's trying to upload. Possibly the binary data was converted for a different model or machine with a different byte order (such as Intel vs Sun SPARC) or the data file was simply corrupted.

108067 3(ERR) GatewayClient(%d:%d)::P_doRecord() received a record with offset beyond its size.

Action: There is something wrong with either the client program or the data it's trying to upload. Possibly the binary data was converted for a different model or machine with a different byte order (such as Intel vs Sun SPARC) or the data file was simply corrupted.

108068 2(CRIT) GatewayClient(%d:%d)::~GatewayClient() forcing system shutdown, exitOnDrop or exitNoBeat is set.

Action: Information about a client being fully disconnected from the Aleri Streaming Platform gateway and all information about this client being disposed.

109000 2(CRIT) GatewayServer(%s): Could not bind port %d.

Action: The explicit port specified with the command line option -g is already taken by another program or another copy of the server already running. Check with your system administrator.

109001 6(INFO) GatewayServer(%s): Started on port %d.

Action: An informational message showing the port where the gateway service has started.

109002 2(CRIT) GatewayServer(%s): Failed to spawn new connection, exception: %s.

Action: Probably the limit of threads or TCP/IP sockets per process is exhausted. Consult your system administrator. Check for runaway processes that may be creating the rogue connections.

109003 2(CRIT) GatewayServer(%s): canonical hostname(%s) empty.

Action: The symbolic host name is not set on the machine. Consult your system administrator.

109004 4(WARNING) GatewayServer::GatewayServer() ephemeral port chosen: %d

Action: An informational message showing the automatically selected port number chosen for the gateway service.

109005 4(WARNING) GatewayServer::GatewayServer() using passed port: %d

Action: An informational message showing the port number specified for the gateway service with the command-line option -g.

110000 7(DEBUG) %s(%s)::setDebugLevel(): Debug level %d.

Action: A debugging message, about changes in the debug messages level.

111000 2(CRIT) Store type "%s" not supported in this version of the platform.

Action: An unknown store kind has been requested. Check the model and correct the error.

111001 2(CRIT) Element "%s" is not valid in this version of the platform.

Action: An unknown XML element has been used in the model. Check the model and correct the error.

112000 2(CRIT) DOMParser()::DOMParser(): Fatal error initializing xerces: %s.

Action: The XML parser failed to initialize. It should normally never happen. Probably there is some issue with the shared libraries configuration.

112001 3(ERR) DOMParser()::handleError() -- %s at line %d, column %d: %s.

Action: A syntax error in an XML file, an AleriML model or other. Check the file and correct the error.

113000 6(INFO) Monitor() Lost connection to subscriber %s, error %s.

Action: Obsolete. It was used with the old method of sending the monitoring data.

113001 2(CRIT) Monitor() Could not start timer.

Action: Obsolete. It was used with the old method of sending the monitoring data.

113002 2(CRIT) Monitor() Could not initialize XML-RPC.

Action: Obsolete. It was used with the old method of sending the monitoring data.

113003 3(ERR) Monitor() Could not establish connection to subscriber %s, error %s.

Action: Obsolete. It was used with the old method of sending the monitoring data.

114000 6(INFO) Platform(%s)::Platform(): Created Platform.

Action: Information about the internal stages of initialization.

114001 6(INFO) Platform(%s)::pauseQueries(): Pausing all SQL queries...

Action: Obsolete.

114002 6(INFO) Platform(%s)::resumeQueries(): Resuming...

Action: Obsolete.

114003 6(INFO) Platform(%s)::pauseStreams(): Pausing all streams...

Action: Obsolete.

114004 6(INFO) Platform(%s)::resumeStreams(): Resuming...

Action: Obsolete.

114005 6(INFO) Platform(%s)::dumpStreams().

Action: The platform was requested to dump the contents of the streams to XML files.

114006 6(INFO) Platform(%s)::run() -- Starting the Platform.

Action: Information about the internal stages of initialization.

114007 6(INFO) Platform(%s)::run() -- Starting Stream threads.

Action: Information about the internal stages of initialization.

114008 6(INFO) Platform(%s)::run() -- Starting thread for %s(%s).

Action: Information about the internal stages of initialization.

114009 2(CRIT) Platform(%s)::run() -- Thread start failed for %s(%s).

Action: The limit of threads per process is probably exhausted. Consult your system administrator.

114010 6(INFO) Platform(%s)::run() -- Waiting to join remaining Stream threads.

Action: Obsolete.

114011 6(INFO) Platform(%s)::run() -- Joined thread for Stream(%s).

Action: The Aleri Streaming Platform has detected that the thread for this stream has completed. The Aleri Streaming Platform collects the stream threads in order so a thread that has not yet exited may prevent the following threads from being collected.

114012 6(INFO) Platform(%s)::run() -- cleaning up %s(%s).

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

114013 1(ALERT) Platform(%s)::run() -- Platform shutdown complete.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

114014 6(INFO) %s(%s)::initialize() -- Full model checked successfully.

Action: In a clustered configuration, the Aleri Streaming Platform checks that the full clustered model is correct before implementing its part of the cluster.

114015 6(INFO) %s(%s)::initialize() -- Disposing of the full model.

Action: In a clustered configuration, the Aleri Streaming Platform checks that the full clustered model is correct before implementing its part of the cluster. The full model is temporarily instantiated and then disposed of.

114016 6(INFO) %s(%s)::initialize() -- type checking the full model...

Action: In a clustered configuration, the Aleri Streaming Platform checks that the full clustered model is correct before implementing its part of the cluster.

114017 6(INFO) %s(%s)::initialize() -- initializing...

Action: Information about the internal stages of initialization.

114018 5(NOTICE) %s(%s)::initializeStreams() -- Stream %s subject to insert-only optimizations.

Action: The Aleri Streaming Platform has detected that this stream may only ever receive the INSERT messages and may use this fact for optimization of the processing.

114019 2(CRIT) Platform()::initialize---Two different persistent Stores have the same "file" attribute "%s"; on dynamic changes any incompatible change of a Store requires changing the file too.

Action: Each persistent store must use a different file for its data. If multiple stores were to use the same file, it would corrupt the data. Check the model to ensure that the files for all the stores are different. In case of a dynamic modification, any incompatible changes to a store require a switch to a new file, as the data is converted from the old file to the new one.

114020 6(INFO) %s(%s)::initialize() -- initialization complete.

Action: Information about the internal stages of initialization.

114021 6(INFO) Platform::syncSourceStreams() -- pausing all Source Stream input.

Action: To synchronize the data, the Aleri Streaming Platform pauses the input to all source streams and lets it fully process the existing input queues. Afterwards, it syncs all persistent stores to disk, and in case of a cluster, makes sure that the data is propagated to the backup nodes. Eventually it restarts the processing.

114022 6(INFO) Platform::syncSourceStreams() -- took: %lf secs to pause Source Streams.

Action: To synchronize the data, the Aleri Streaming Platform pauses the input to all source streams and lets it fully process the existing input queues. Afterwards, it syncs all persistent stores to disk, and in case of a cluster, makes sure that the data is propagated to the backup nodes. Eventually it restarts the processing.

114023 6(INFO) Platform::syncSourceStreams() -- took: %lf secs for Source Streams.to drain.

Action: To synchronize the data, the Aleri Streaming Platform pauses the input to all source streams and lets it fully process the existing input queues. Afterwards, it syncs all persistent stores to disk, and in case of a cluster, makes sure that the data is propagated to the backup nodes. Eventually it restarts the processing.

114024 6(INFO) Platform::syncSourceStreams() -- took: %lf secs to sync stores.

Action: To synchronize the data, the Aleri Streaming Platform pauses the input to all source streams and lets it fully process the existing input queues. Afterwards, it syncs all persistent stores to disk, and in case of a cluster, makes sure that the data is propagated to the backup nodes. Eventually it restarts the processing.

114025 6(INFO) Platform::syncSourceStreams() -- resumed all Source Stream input.

Action: To synchronize the data, the Aleri Streaming Platform pauses the input to all source streams and lets it fully process the existing input queues. Afterwards, it syncs all persistent stores to disk, and in case of a cluster, makes sure that the data is propagated to the backup nodes. Eventually it restarts the processing.

114026 6(INFO) Platform::quiesced() -- waiting on all queues to drain.

Action: The Aleri Streaming Platform has received a request to quiesce and started draining the queued up data.

114027 6(INFO) Platform::quiesced() -- all queues have drained.

Action: The queued up data has been fully processed and drained throughout Aleri Streaming Platform, the normal processing will resume.

114028 6(INFO) Platform::areQueuesEmpty() -- path rooted at stream %s still has data in it.

Action: On an XMLRPC call asking if the platform is quiesced, these message provide the information on which stream queues still contain data and are busy processing it.

114029 2(CRIT) Platform::initializeStreams() Inner join found in non-insert-only stream %s.

Action: The inner joins are only allowed on the insert-only streams. That is, from a source stream with attribute insertOnly="true" or a stream that is derived only from such source streams and doesn't use certain features like InputWindows.

114030 2(CRIT) Platform(%s)::initialize---Two different Streams have the same "ofile" attribute "%s".

Action: The "ofile" attribute may be used to dump the stream contents automatically on the Aleri Streaming Platform exit. If multiple streams were to dump the data into the same file, it would overwrite each other. Check the model and correct the file names.

114031 2(CRIT) Platform(%s)::initialize---the store associated with stream %s must be not be "stateless".

Action: Only some very limited streams may use the stateless store. Most of the streams can't. See the Authoring Reference for details.

114032 2(CRIT) Platform()::buildLogGraph() -- cyclic dependency in logstores, consistent recovery impossible, aborting.

Action: For the Log Stores to provide a correct persistence, the streams that use it must follow certain rules of interconnection. If stream S1 uses the persistent store A, and its derived stream S2 uses the persistent store B, none of the streams derived from S2 may use the store A. In general, it's best to either put only the source streams into a persistent store or put the whole "vertical slices" of a model, starting from a source stream and all the streams derived from it into the same persistent store.

114033 6(INFO) Platform()::buildLogGraph() -- logstore checkpoint order: %s

Action: For the Log Stores to provide a correct persistence, the streams that use it must follow a certain order for checkpointing the stores from those containing the most derived information to the ones containing the source information. This informational message shows the order.

114034 2(CRIT) Platform()::buildLogGraph() -- logstore %s: has children %s

Action: For the Log Stores to provide a correct persistence, the streams that use it must follow a certain order for checkpointing the stores, from those containing the most derived information to ones containing the source information. As this order is being created, these messages show the derivation.

114035 7(DEBUG) Platform()::lockExternal() -- locking

Action: Only one XMLRPC command may be executed at a time (with the exception of certain long-running commands that wait for a condition). This message shows that a command is acquiring the lock before starting the execution.

114036 7(DEBUG) Platform()::unlockExternal -- unlocking

Action: Only one XMLRPC command may be executed at a time (with the exception of certain long-running commands that wait for a condition). This message shows that a command has completed its execution, and other commands may run now.

114037 3(ERR) Platform()::syncSourceStreams -- redundancy is set, but sync of secondary failed!

Action: You should check if the slave node of the cluster has failed or was not yet started.

114038 3(ERR) Platform()::setStreamRedundancy -- could not establish XMLrpc connection to secondary!

Action: You should check if the slave node of the cluster has failed or was not yet started.

114039 6(INFO) Platform(%s)::XML Model Version: %s

Action: It has information about the AleriML version of the model being loaded.

114040 4(WARNING) Platform(%s)::Warning, no XML Model Version Specified.

Action: The information about the AleriML version of the model being loaded is not present in the model. If no syntax errors are found, the Aleri Streaming Platform at-

tempts to execute the model anyway.

114041 2(CRIT) Platform(%s)::Mismatched XML Model Version detected, expected: %s, found: %s

Action: Probably the model was developed for a different version of AleriML. Use the tool sp_upgrade for conversion of the models to the current version.

114042 2(CRIT) %s(%s)::Bad access control specification: '%s' is not a pair of strings separated by ':'.

Action: The attribute "restrictAccess" has been misformatted. It must contain a space-separated list of colon-separated pairs. Check the model and correct it.

114043 2(CRIT) %s(%s)::Bad access control specification: unknown access control type in pair '%s'.

Action: The attribute "restrictAccess" has been misformatted. Check the model and correct it.

114044 2(CRIT) Platform()::initialize -- cluster '%s' module '%s' is not in the config file.

Action: In a clustered model, the module specified with option -M of the cluster specified with the option -C of the command line is not present in the model. Check and correct the command line arguments and/or the model.

114045 2(CRIT) Platform()::changeRestriction -- Bad 'from' restriction value %d.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

114046 2(CRIT) Platform()::changeRestriction -- Bad 'to' restriction value %d.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

114047 2(CRIT) Platform()::changeRestriction -- Can't change restriction when a platform is down.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

114048 2(CRIT) Platform()::changeRestriction -- State consistency error, no restrictions of %d seen yet.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

114049 2(CRIT) Platform(): unknown platform modification option '%s'.

Action: The dynamic modification XMLRPC command may be given a number of options. See the Administrator's Guide for the information on the Dynamic Services.

114050 6(INFO) Platform()::modify(): modifying stream config %s(%s).

Action: The dynamic modification has requested a compatible modification of this stream. The stream is modified in place.

114051 6(INFO) Platform()::modify(): modification of %s(%s) succeeded.

Action: The dynamic modification has requested a compatible modification of this stream. The stream was successfully modified.

114052 2(CRIT) Platform()::modify(): modification of %s(%s) FAILED.

Action: The dynamic modification has requested a compatible modification of this stream. The stream could not be modified. See the preceding error messages for error details. The whole modification is rolled back. Check the new model and correct the error.

114053 6(INFO) Platform()::modify(): committing modification of %s(%s).

Action: The dynamic modification has requested a compatible modification of this stream. All the modifications have succeeded, so now it is committed.

114054 6(INFO) Platform()::modify(): rolling back modification of %s(%s).

Action: The dynamic modification has requested a compatible modification of this stream. An attempt to modify this or another stream has failed so all modifications are rolled back.

114055 2(CRIT) Platform()::modify(): Failed to add node %s(%s).

Action: The dynamic modification has requested to add a new XML element, but the addition has failed. The preceding error messages may contain more information about the reasons of the failure.

114056 6(INFO) Platform()::modify(): Initializing the newly added %s(%s).

Action: The dynamic modification has requested to add a new stream so it is now added.

114057 2(CRIT) Platform()::modify(): Dynamic configuration modifications are not supported in clustered mode.

Action: See the Administrator's Guide for the information on the Dynamic Services. The dynamic modifications may not be applied to an instance of Aleri Streaming Platform that is a part of a cluster.

114060 2(CRIT) Platform()::initialize: Non-metadata stream %s(%s) is not allowed to use AleriMetadataStore.

Action: The names starting with "Aleri" are reserved and cannot be used in a model.

114061 6(INFO) Platform()::modify(): Renaming node %s(%s) to %s(%s).

Action: The dynamic modification has requested to rename an XML element. It is now renamed.

114062 6(INFO) Platform()::modify(): Renaming node %s(%s) back to %s(%s).

Action: The dynamic modification has requested to rename an XML element. Some part of the modification has failed so the element is renamed back.

114063 6(INFO) Platform()::modify(): The old node %s(%s) will be deleted.

Action: The dynamic modification has requested to delete an XML element. It is now deleted.

114064 2(CRIT) Platform()::modify(): Requested modification on the Source Stream %s(%s) is potentially unsafe, use option "base" to override.

Action: See the Administrator's Guide for the information on the Dynamic Services. The source streams are different from the other streams in the way that its contents can not be regenerated. Any incompatible changes would cause a data loss. Therefore, the option "base" must be specified to allow any incompatible changes to prevent the accidental data loss. If it was not intended, check the model and correct it. The data in the changed source streams may be transferred from the old model to the new one through use of the conversion model.

114065 2(CRIT) Platform()::modify(): The data directory "%s" for dynamically added store %s(%s) already exists.

Action: Each persistent store must use a different file for its data. If multiple stores were to use the same file, it would corrupt the data. Check the model to ensure that the files for all the stores are different. In case of a dynamic modification, any incompatible changes to a store require a switch to a new file, as the data is converted from the old file to the new one.

114066 2(CRIT) Platform()::modify(): Invalid renaming attempt of node %s(%s) to %s(%s).

Action: The renaming must be direct. The new name must not be the same as the name of any other node in the new model. Also an old node can not be renamed to more than one new name. Check the new model and correct the error.

114067 6(INFO) Platform()::modify(): modifying store config %s(%s).

Action: The dynamic modification has requested a compatible modification of this store. The store is modified in place.

114068 6(INFO) Platform()::modify(): Stream %s(%s) will be deleted and re-created, because its store %s(%s) gets re-created.

Action: In case if a store is dynamically modified in an incompatible way, this is processed by deleting the store and creating a new one with the same name but different configuration. Since the streams are tied to a store, this means that any streams using this store must also be deleted and reincarnated even if these streams have no incompatible changes.

114069 2(CRIT) Platform()::modify(): options '%s' and '%s' are mutually exclusive.

Action: See the Administrator's Guide for the information on the Dynamic Services. The requested dynamic modification options are incompatible. Check and correct your request.

114068 6(INFO) Platform()::modify(): Node %s(%s) is marked as modified because it depends on a modified node %s(%s).

Action: Certain XML nodes in the model are used by other nodes. For example, the <Globals> node is used by the streams. Any change to such underlying node requires all the dependent nodes to be recompiled, even though they might not be changed themselves. The dynamic modification does such recompilation automatically. This message displays such dependencies found by the dynamic services.

114069 6(INFO) Platform()::modify(): Deleting old %s(%s).

Action: A node was present in the old model but not in the new one. It will be deleted during the modification.

114070 6(INFO) Platform()::modify(): modifying node %s(%s).

Action: The dynamic modification has requested a compatible modification of this node. It is modified in place.

114071 2(CRIT) Platform()::modify(): An incompatible change requested for %s(%s), possibly as a propagation of other changes. Not allowed with option 'noregen'.

Action: The option "noregen" bypasses the regeneration after the dynamic modification. It expects that the user knows this would not introduce a major discrepancy in the data or accept such a discrepancy, such as a tradeoff for a faster modification. The incompatible changes are not allowed with the option "noregen" since this combination would cause grossly misformatted data that may cause the Aleri Streaming Platform to crash. Remove the option "noregen" from the dynamic modification request to resolve this issue.

114072 2(CRIT) Platform()::initialize(): A conversion model may not use log stores, found %s(%s).

Action: The conversion model is temporary by definition, and it takes the data from the old model and creates the data to be put into the new model. The persistent stores may not be used in the conversion model. Change the conversion model to use the memory store.

114073 2(CRIT) Platform()::initialize(): Stream %s(%s) in a normal platform may not use attributes 'convsrc' and 'convdst'.

Action: The attributes "convsrc" and "convdst" are reserved for use in the conversion model, to describe how it is connected to the streams in the old and new model. Change your model to not use these attributes.

114074 2(CRIT) Platform()::initialize(): Stream %s(%s) in a conversion must have the 'convsrc' attribute.

Action: Each source stream in the conversion model must take its data from a stream in the old model. The attribute "convsrc" sets up this connection. Correct your conversion model.

114075 2(CRIT) Platform()::initialize(): Attribute 'convsrc' of %s(%s) refers to an unknown stream '%s'.

Action: Each source stream in the conversion model must take its data from a stream in the old model. The attribute "convsrc" names the stream in the old model. No such stream exists in the old model. Correct your conversion model.

114076 2(CRIT) Platform()::initialize(): Attribute 'convdst' of %s(%s) refers to an unknown or unsuitable stream '%s', which must be a Source Stream.

Action: The results of the conversion model are to be placed into the source streams of the new model. The attribute "convdst" names the stream in the new model. No such source stream exists in the new model. Correct your conversion model.

114077 6(INFO) Platform()::run() -- All stream threads exited.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

114078 4(WARNING) Platform()::setMtDebugString() -- Unknown suboption '%c'.

Action: The option -D may be used to specify a set of debugging features. Only a limited subset of these features is published, the rest are used internally by Aleri. The suboptions are specified as the letter(s) following -D, such as -DD. Check and correct your

command line.

114079 1(ALERT) Platform()::run -- Platform Server alive & ready for services

Action: Information that the initialization is completed.

114080 6(INFO) Platform::quiesceMeta() -- waiting on metadata queues to drain.

Action: The shutdown procedure, dynamic modifications and certain other requests require that all the streams in the Aleri Streaming Platform be quiesced, including the metadata streams. This indicates the start of the quiescence of the metadata streams.

114081 6(INFO) Platform::quiesceMeta() -- metadata queues have drained.

Action: The shutdown procedure, dynamic modifications and certain other requests require that all streams in the Aleri Streaming Platform be quiesced, including the metadata streams. This indicates the completion of the quiescence of the metadata streams.

114082 2(CRIT) PlatformControl()::quiesceMeta -- called at wrong restriction level %d, ignored

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

114083 6(INFO) Platform()::enableTrace() -- %s the trace mode.

Action: The request to enable or disable trace mode was received and processed.

114084 4(WARNING) Platform()::modify() -- can not be modified while in trace mode.

Action: While the Aleri Streaming Platform is in the trace mode, the dynamic modifications may not be applied. Disable the trace mode and retry the dynamic modification.

114085 6(INFO) Platform()::traceRequestPause() -- Platform is being paused.

Action: The pause request was received and processed from XMLRPC or on a breakpoint. The state of the Aleri Streaming Platform may now be examined.

114086 6(INFO) Platform()::traceRun() -- Platform has continued running.

Action: The Aleri Streaming Platform has continued running after being paused.

114087 2(CRIT) Platform()::loadConnections(): Connections initialization failure: %s.

Action: A connection type shared library could not be loaded. The message contains the detailed reason. This connection type then may not be used in the models, and attempts to use it are reported as a syntax error. Check that the shared library is present, has correct permissions, was built for the correct architecture and this exact version of the Aleri Streaming Platform.

114088 2(CRIT) Platform(%s)::initialize -- Two different Connectors have the same "name" attribute "%s".

Action: Each Connection must have an unique name. Check and correct the model.

114089 6(INFO) Platform()::run() -- Starting the Connectors.

Action: Information about the internal stages of initialization.

114090 6(INFO) Platform()::run() -- Waiting for Connectors to complete the initial loading.

Action: Some connection types are "one-shot", in that it loads a set of data when started and exited. The Aleri Streaming Platform lets these connections load data to initialize the model before continuing with the normal operations. See the Authoring Guide for more information on the connection states. An incorrectly programmed connection type (especially the user-defined types) may never report that it has completed the initial loading and would cause the Aleri Streaming Platform to be stuck forever. If such a situation occurs, check your connections.

114091 6(INFO) Platform()::run() -- Connectors completed the initial loading.

Action: Information about the internal stages of initialization.

114092 2(CRIT) Platform()::initialize -- A connector on stream '%s' has no name.

Action: Each Connection must have an unique name. Check and correct the model.

114093 2(CRIT) Platform()::initialize -- The 'connection' attribute is missing in ConnectionRef node %s.

Action: Each ConnectionRef in the start-up sequence must refer to an existing Connection node. Check and correct the model.

114094 2(CRIT) Platform()::initialize -- The StartUp has multiple references to Connection '%s'.

Action: If the start-up sequence is specified explicitly in the StartUp node, it must refer exactly once to each of the Connection nodes. Check and correct the model.

114095 2(CRIT) Platform()::initialize -- ConnectionRef refers to a non-existing Connection name '%s'.

Action: Each ConnectionRef in the start-up sequence must refer to an existing Connection node. Check and correct the model.

114096 2(CRIT) Platform()::initialize -- The StartUp has no references to Connection '%s', list it under <ConnectionGroup type="nostart"> if it doesn't need to be started automatically.

Action: If the start-up sequence is specified explicitly in the StartUp node, it must refer exactly once to each of the Connection nodes. Check and correct the model.

114097 4(WARNING) Platform()::run() -- Connector '%s' start failure: %s.

Action: A connection has failed to start, with the specified reason. Check and correct the connection parameters if necessary. Check the external environment for possible issues (such as sockets being already used by other processes, incorrect paths etc).

114098 4(WARNING) Platform()::run() -- Internal error, can not find the stream for Connector '%s'.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

114099 2(CRIT) Platform()::initialize -- Only one StartUp node is supported per module.

Action: The model may have at most one StartUp node for each module of the cluster, or one StartUp node for the whole non-clustered model. Check and correct the model.

114100 2(CRIT) Platform()::initialize -- Only one Global node is supported per module.

Action: The model may have at most one Global node for each module of the cluster, or one Global node for the whole non-clustered model. Check and correct the model.

114101 2(CRIT) Platform()::initialize -- Error found in Global declarations: %s.

Action: The global declaration contains a syntax error. Check and correct the model.

114102 2(CRIT) Platform(): dynamic modification of Global is not permitted in 'noregen' mode.

Action: The option "noregen" bypasses the regeneration after the dynamic modification. It expects that the user knows this would not introduce a major discrepancy in the data or accepts such a discrepancy, as a tradeoff for a faster modification. The changes to the Globals are not allowed with the option "noregen" since this combination may cause the stream types to change, which would cause grossly misformatted data that may cause the Aleri Streaming Platform to crash. Remove the option "noregen" from the dynamic modification request to resolve this issue.

114103 2(CRIT) Platform(): Error in compiling Global initializer expression: %s.

Action: The global declaration contains a syntax error. Check and correct the model.

114104 6(INFO) Platform()::run -- Platform Command Server is started.

Action: Information about the internal stages of initialization.

114105 1(ALERT) Platform()::notifyRequestExit() -- Platform shutdown sequence initiated.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

114108 6(INFO) Platform::quiesceSource -- waiting for SourceStreams input queues to drain.

Action: Committing the input to the store requires that the source streams fully process their current input queue. No new data will be put into their input queue until the existing requests are processed and committed to the store.

114109 6(INFO) Platform::quiesceSource -- SourceStreams input queues have drained.

Action: The source streams have fully processed their input.

114110 3(ERR) Platform()::daemonize(): Switching to daemon mode.

Action: All the log messages after this one are written to syslog.

114111 3(ERR) Platform()::daemonize(): Daemon process current directory set to '%s'.

Action: During daemonization the Aleri Streaming Platform may change its current directory to avoid holding down the directory where it was started. It expects all file pathnames to be specified either as absolute or relative to the new current directory.

114112 3(ERR) Platform()::validateLicense(): Could not allocate license control object.

Action: Contact your system administrator to either increase the per-process memory usage limits or add physical memory to the machine.

114113 3(ERR) Platform()::processSignal(): fatal error in thread id %d, thread name '%s'; aborting.

Action: Some major crash has happened in the Aleri Streaming Platform. Ideally, this should never happen. If it happens, collect the log file, the core dump and the Aleri Streaming Platform binaries and contact the Aleri support. If you can provide a way to

reproduce this crash, preferably with a smaller model and data set, it would be very helpful and lead to a quicker fix.

114113 3(ERR) Platform()::initialize(): Unable to set the time rate %lf.

Action: The time rate specified at command line with option -R is incorrect. Check and correct your command line.

114114 3(ERR) Platform():daemonize(): could not fork the daemonized process: %s (%d)

Action: The fork() system call that creates the daemonized process has failed. Possibly the system has run out of resources. Check with your system administrator.

114115 2(CRIT) Platform()::lockExternal() -- timed out at %d sec., the current owner of the lock is '%s'.

Action: Only one XMLRPC command may be executed at a time (with the exception of certain long-running commands that wait for a condition). This message shows that a command could not get this exclusive access for a relatively long period of time and gave up. The message shows the information about the other outstanding command that is currently hogging the exclusive access. For some commands, such as those that perform the Dynamic Service modifications, taking a long time is normal.

115000 2(CRIT) Authenticate(): Could not initialize PAM layer.

Action: Probably PAM is not configured in the operating system. Check with your system administrator.

115001 2(CRIT) Authenticate(): Could not shutdown PAM layer.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

115002 2(CRIT) Authenticate()::setDirectDatabase(): Syntax error in user name at '%-.20s...'.

Action: The testing user database specified with option -A is misformatted. Check and correct your command line.

115003 2(CRIT) Authenticate()::setDirectDatabase(): Syntax error after user name '%s'.

Action: The testing user database specified with option -A is misformatted. Check and correct your command line.

115004 2(CRIT) Authenticate()::setDirectDatabase(): Syntax error after password for user '%s'.

Action: The testing user database specified with option -A is misformatted. Check and correct your command line.

115005 2(CRIT) Authenticate()::setDirectDatabase(): Syntax error after role '%s' for user '%s'.

Action: The testing user database specified with option -A is misformatted. Check and correct your command line.

116000 5(NOTICE) SqlControl(%d): Authentication successful for user %s, database %s.

Action: Informational message about a successful login to the ODBC service.

116001 3(ERR) SqlControl(%d): Authentication rejected for user %s, database %s.

Action: A client tried to log in to the ODBC service with a wrong password or unknown RSA key. If you believe that the password is correct and PAM authentication method is used, contact your system administrator to check the PAM configuration. If you believe that the RSA key is correct, and the RSA authentication is used, check whether the matching public key is installed in the server's RSA public key directory. Make sure that the client uses the secret key file and the server has the public key file, not the other way around. See the Administrator's Guide for details.

116002 5(NOTICE) SqlControl(%d): exited control thread on port %d

Action: The Aleri Streaming Platform has completed servicing queries from an ODBC client.

116003 5(NOTICE) SqlControl(%d): Skipping message from client of size %u because it's too large

Action: There is something wrong with either the client program or the data it's trying to upload. Possibly the binary data was converted for a different model or machine with a different byte order (such as Intel vs Sun SPARC) or the data file was simply corrupted.

116004 5(NOTICE) SqlControl(%d): Detected shutdown of client socket

Action: An ODBC client has disconnected from the socket.

116005 5(NOTICE) SqlControl(%d): Detected large, possibly bad message from client of size %d

Action: There is something wrong with either the client program or the data it's trying to upload. Possibly the binary data was converted for a different model or machine with a different byte order (such as Intel vs Sun SPARC) or the data file was simply corrupted.

116006 5(NOTICE) SqlControl(%d): started control thread

Action: A new ODBC service thread was created for a new ODBC connection.

116007 7(DEBUG) SqlControl(%d): Received SQL statement %s

Action: An SQL query was received from an ODBC client.

116008 7(DEBUG) SqlControl(%d): started SQL statement execution.

Action: The SQL query was parsed successfully, and its execution has started.

116009 7(DEBUG) SqlControl(%d): ended SQL statement execution.

Action: The SQL query execution has completed. If any errors were detected, it was reported to the ODBC client.

116010 7(DEBUG) SqlControl(%d): started result set send.

Action: The SQL query has completed successfully, and the results are about to be sent to the ODBC client.

116011 7(DEBUG) SqlControl(%d): ended result set send.

Action: The SQL query results have been fully sent to the client.

117000 2(CRIT) SqlQuery(): could not get ephemeral port.

Action: An attempt to find an unused port number for ODBC service has failed. Probably there is some major issue in the machine's networking configuration. There is also a possibility of a rare failure if two instances of Aleri Streaming Platform or a similar program try to allocate unused port numbers at the same time.

117001 2(CRIT) SqlQuery(%d): could not bind port %d.

Action: The explicit port specified with the command line option -q is already taken by another program, or possibly by another copy of the server already running. Check with your system administrator.

117002 6(INFO) SqlQuery(%d): listening on port %d.

Action: An informational message about starting the ODBC service. Shows the port number that may have been specified with option -q or chosen automatically with the parameter -q 0. Lack of this message may mean that the option -q was not specified and the ODBC service was not started.

117003 5(NOTICE) SqlQuery(%d): spawned control thread on port %d

Action: A connection has been received from an ODBC client and a server thread has been created to service it.

118000 2(CRIT) LogFile(%s)::LogFile() -- file exists, but is truncated, cannot continue.

Action: The Log Store data file is probably corrupted in some way.

118001 6(INFO) LogFile(%s)::LogFile() -- opening existing log file of size %ld.

Action: An informational message about the file used for LogStore.

118002 2(CRIT) LogFile(%s)::LogFile() -- could not open file: %s.

Action: The Aleri Streaming Platform has experienced an error when trying to open a LogStore file. Check whether the path exists and whether the permissions are correct. If the path is relative, check that the Aleri Streaming Platform's current directory is correct.

118003 2(CRIT) LogFile(%s)::LogFile() -- could not create file: %s

Action: The Aleri Streaming Platform has experienced an error when trying to open a LogStore file. Check whether the path exists and whether the permissions are correct. If the path is relative, check that the Aleri Streaming Platform's current directory is correct.

118004 6(INFO) LogFile(%s)::LogFile() -- extending new log file to %ld bytes.

Action: The LogStore file already existed but was shorter than specified in the model. It was extended to the new specification.

118005 2(CRIT) LogFile(%s)::LogFile() -- could not map file: %s.

Action: The Log Store data file is probably corrupted in some way.

118006 2(CRIT) LogFile(%s)::~LogFile( ) -- could not unmap file.

Action: The Log Store data file is probably corrupted in some way.

118007 2(CRIT) %s(%s)::LogFile() -- bad magic number.

Action: The Log Store data file is probably corrupted in some way.

118008 2(CRIT) %s(%s)::LogFile() -- bad checksum.

Action: The Log Store data file is probably corrupted in some way.

118009 2(CRIT) %s(%s)::LogFile() -- incompatible version number %d.%d, current %d.%d.

Action: An attempt was made to open a LogStore file in a newer format, created with a newer version of the Aleri Streaming Platform. It may also happen the other way around, when opening a much older file with a new version of the platform, that does not know any more, how to upgrade such a file. When the Aleri Streaming Platform opens an older supported LogStore file, it upgrades its format to the new version, and the older version of the Aleri Streaming Platform won't be able to open this file any more. The workaround is to dump the data with a compatible version of the Aleri Streaming Platform and reload it from scratch.

118010 7(DEBUG) LogFile(%s)::~LogFile() -- logfile closed.

Action: The LogStore file has been unmapped and closed.

118011 7(DEBUG) LogFile(%s)::alloc(%ld) -- at address %ld

Action: The details of the internal space allocation in the LogStore.

118012 7(DEBUG) LogFile(%s)::append(%ld, %ld) -- written to address %ld

Action: The details of the internal space allocation in the LogStore.

118013 7(DEBUG) LogFile(%s)::write(%ld, %ld, %ld)

Action: The details of the internal space allocation in the LogStore.

118014 2(CRIT) LogFile(%s)::LogFile() -- could not extend file: %s.

Action: The Log Store data file is probably corrupted in some way.

118015 2(CRIT) LogFile(%s)::LogFile() -- could not mark file with EOF mark: %s.

Action: The Log Store data file is probably corrupted in some way.

118016 6(INFO) LogFile(%s)::alloc() -- wrapping logfile(len), tail=%ld, size=%ld, requested length=%ld

Action: The last write position in the LogStore has reached the end of file and continued writing circularly from the start of the file. The log cleaning makes sure that the valid data does not get overwritten. The frequency of log wraps shows the intensity of the data modifications in the store.

118016 6(INFO) LogFile(%s)::alloc() -- wrapping logfile(align), tail=%ld, size=%ld, requested length=%ld

Action: The last write position in the LogStore has reached the end of file and continued writing circularly from the start of the file. The log cleaning makes sure that the valid data does not get overwritten. The frequency of log wraps shows the intensity of the data modifications in the store.

118017 6(INFO) LogFile(%s)::append() -- wrapping logfile(len), tail=%ld, size=%ld, requested length=%ld

Action: The last write position in the LogStore has reached the end of file and continued writing circularly from the start of the file. The log cleaning makes sure that the valid data does not get overwritten. The frequency of log wraps shows the intensity of the data modifications in the store.

118017 6(INFO) LogFile(%s)::append() -- wrapping logfile(align), tail=%ld, size=%ld, requested length=%ld

Action: The last write position in the LogStore has reached the end of file and continued writing circularly from the start of the file. The log cleaning makes sure that the valid data does not get overwritten. The frequency of log wraps shows the intensity of the data modifications in the store.

118018 6(INFO) %s(%s)::LogFile() -- upgraded file version from %d.%d to %d.%d.

Action: When the Aleri Streaming Platform opens an existing LogStore of an older version, it transparently upgrades the format of the store to the newer version. The LogStore version numbers are not directly related to the Aleri Streaming Platform version numbers. Such an upgraded LogStore can not be opened by the old version of the Aleri Streaming Platform any more. To preserve the compatibility with the older versions, either make a backup copy of the LogStore in advance or dump the contents of the store into XML with the new Aleri Streaming Platform version and load the data back into the old platform version.

118019 2(CRIT) LogFile(%s) -- overfilled, tried to overwrite its head, stopping.

Action: The specified Log Store size was too small and ran out of space. Monitor the Log Store usage and increase it in advance to prevent this fatal error. The store reserve is intended to guarantee against this situation, but in exceptional circumstances it still may happen. If the log store comes to this condition, it can not be resized any more. You must start with a new clean store (and increase its size too).

118020 2(CRIT) LogFile(%s)::LogFile() -- probably another platform instance is already running, could not lock file: %s.

Action: To prevent multiple instances of Aleri Streaming Platform from trying to write to the same file and corrupting it, the Aleri Streaming Platform locks its LogStore files. If the file is already locked by some other process, it can not be used.

118021 6(INFO) LogFile(%s)::LogFile() mapped at address, 0x%lx, size, %ld bytes.

Action: Informational message that the LogStore file has been successfully mapped into memory.

118022 6(INFO) LogFile(%s)::LogFile() The log file size can not be reduced, running with the old size %ld MB.

Action: The log file size can not be safely reduced in the model before restarting it. The model will continue running with the old file size. The only way to reduce the log file is to delete the store and have it re-created from scratch. To preserve the data, save it and reload afterwards.

119000 2(CRIT) LogFile(%s)::sync(%ld, %ld): msync() failed, errno: %d, errmsg: %s.

Action: Possibly the underlying disk drive or filesystem has failed. Consult the system administrator. If no disk error, contact the Aleri support.

120000 6(INFO) %s(%s)::checkpoint() -- time: %lf, wrote log checkpoint at %ld for %d indices.

Action: Information about the location of important data structures in the LogStore file.

120001 6(INFO) %s(%s)::checkpoint() -- time: %lf, wrote log header (tail=%ld).

Action: Information about the location of important data structures in the LogStore file.

120002 7(DEBUG) LogStoreIndex()::reload() -- reloading index at offset %ld of store '%s'.

Action: When the Aleri Streaming Platform is restarted with an existing LogStore file, it reads the contents of this file and uses it to populate the initial state of the streams.

120003 7(DEBUG) LogStoreIndex()::reload() -- recreating index '%s'.

Action: The index reported in the previous message is used to populate the named stream.

120004 6(INFO) %s(%s)::reload() -- recovered empty log.

Action: An existing LogStore file has been found, but it is empty.

120005 6(INFO) %s(%s)::reload() -- reloading %ld indices from last checkpoint.

Action: When the Aleri Streaming Platform is restarted with an existing LogStore file, it reads the contents of this file and uses it to populate the initial state of the streams.

120006 6(INFO) %s(%s)::reload() -- attempting roll-forward recovery from offset %ld.

Action: After the last checkpointed state was read from the LogStore and used to populate the streams, the source stream records that have not been checkpointed yet but remembered in the store are played back again to fully restore the state of the model.

120007 6(INFO) %s(%s)::reload() -- %s(addr=%ld, size=%ld) .

Action: Stages of the data reload.

120008 6(INFO) %s(%s)::reload() -- skipping record (client=%ld, addr=%ld, length=%ld).

Action: The non-checkpointed records from the derived stores are not played back because it is not consistent. The playback of the source stream records regenerates the state of the derived streams in a consistent way.

120009 6(INFO) %s(%s)::readHeader() -- reading log header.

Action: Stages of the data reload.

120010 2(CRIT) %s(%s)::compact() freeSpace:%ld kB (%d%%) after full scrubbing is still under %d%% of required reserve, liveSize: %ld kB of totalSize: %ld kB, increase the store size and restart, stopping.

Action: The LogStore needs a padding of free space to run. If the free space drops under the value of the reservePct attribute (20% by default), it does not leave enough service space for cleaning, and the store can not continue running. Increase the size of the store and restart the model. Monitor the space used and resize the store preventively to avoid such failures. The liveSize includes the estimation of the indexing overhead.

120011 6(INFO) %s(%s)::compact() START, tail: %ld, cleanerTail: %ld, freeSpace:%ld kB (%d%%).

Action: When the LogStore runs out of free space, it tries to restore it by compacting the data and removing the records that are not used any more.

120012 6(INFO) %s(%s)::compact() END, time: %lf, tail: %ld, cleanerTail: %ld, freeSpace:%ld kB (%d%%), liveDataCopied: %ld kB, full liveSize: %ld kB.

Action: Information about the results of a compaction run. The liveSize includes the estimation of the indexing overhead. See the Administration Guide for further explanation.

120013 2(CRIT) %s(%s)::compact() compactification cause log tail to enter redzone.

Action: Obsolete.

120014 2(CRIT) %s(%s)::readHeader() -- header checksum invalid.

Action: The LogStore file has an incorrect header checksum. Either it has been corrupted or it is a wrong file. The Aleri Streaming Platform can not start. Try using a different file name or removing the corrupted file.

120015 6(INFO) %s(%s)::checkpoint() -- logfile not dirty, skipping checkpoint.

Action: The Aleri Streaming Platform checkpoints all the stores at the same time. If some store has seen no changes since the last checkpoint, it does not need to checkpoint again.

120016 2(CRIT) %s(%s)::readHeader() -- incompatible version %d.%d in the log file, current %d.%d.

Action: An attempt was made to open a LogStore file in a newer format, created with a newer version of the Aleri Streaming Platform. It may also happen the other way around, when opening a much older file with a new version of the platform, that does not know any more, how to upgrade such a file. When the Aleri Streaming Platform opens an older supported LogStore file, it upgrades its format to the new version, and the older version of the Aleri Streaming Platform won't be able to open this file any more. The workaround is to dump the data with a compatible version of the Aleri Streaming Platform and reload it from scratch.

120018 7(DEBUG) %s(%s)::reload() -- successful restart.

Action: The last state of the store has been successfully reloaded.

120019 6(INFO) %s(%s)::initialize() -- initializing, fullSize: %ld mb.

Action: Information about stages of the LogStore initialization.

120020 2(CRIT) %s(%s)::initialize() -- the log-structured store size is limited to 2GB on 32-bit platforms.

Action: On the 32-bit machines the sum of LogStore sizes is limited to 2GB. In practice even smaller because the OS will limit the memory usage.

120021 2(CRIT) %s(%s)::initialize() -- '%s' is not a directory.

Action: The "file" attribute of the LogStore must actually refer to a directory where the store files are created. A name of non-existing directory is OK, as long as its parent

directory exists. But if there is already a plain file with the same name, a directory can not be created in its place.

120022 6(INFO) %s(%s)::initialize() -- creating directory '%s'.

Action: The "file" attribute has referred to a non-existing directory. It is automatically created.

120023 2(CRIT) %s(%s)::initialize() -- could not create directory '%s': %s.

Action: The "file" attribute has referred to a non-existing directory. The Aleri Streaming Platform tried to create it and failed. Check that the parent directory exists and that permissions are correct.

120026 2(CRIT) %s(%s)::initialize() -- Missing name of file for log store; exiting.

Action: The Log Store must have the attribute "file" set. Check the model.

120029 6(INFO) %s(%s)::~LogStore( ) -- syncing log.

Action: Information about stages of closing the LogStore.

120030 7(DEBUG) %s(%s)::~LogStore( ) -- done.

Action: Information about stages of closing the LogStore.

120035 7(DEBUG) LogStore(%s)::getIndex() -- request for index '%s'.

Action: Information about stages of the LogStore initialization.

120036 6(INFO) %s(%s)::getIndex(%s) -- creating index.

Action: A stream has been found that previously had no matching index in the LogStore file. The index is created.

120037 2(CRIT) %s(%s)::checkSignature(%s) -- could not find stream in configuration.

Action: Probably the Log Store file is used with a mismatched configuration file. Make sure that it matches. If any LogStore stream has been changed in the model, the old LogStore file can not be used with this model any more. Start with a fresh file.

120038 2(CRIT) %s(%s)::checkSignature(%s) -- bad signature, configuration changed.

Action: Probably the Log Store file is used with a mismatched configuration file. Make sure that it matches. If any LogStore stream has been changed in the model, the old LogStore file can not be used with this model any more. Start with a fresh file.

120039 7(DEBUG) %s(%s)::checkSignature(%s) -- good signature, configuration unchanged.

Action: Information about stages of the LogStore initialization.

120040 2(CRIT) %s(%s)::() -- cannot open file '%s': %s.

Action: One of the LogStore files can not be opened or created. Check the permissions in the filesystem.

120041 2(CRIT) %s(%s)::reload() -- fullsize of %d, exceeds the 32 bit limit (2047), aborting'.

Action: On the 32-bit machines the sum of LogStore sizes is limited to 2GB. In practice

even smaller because the OS will limit the memory usage.

120042 7(DEBUG) %s(%s)::reload() -- Found BEGINTRANS during roll-forward recovery.

Action: Information about stages of the LogStore initialization.

120043 7(DEBUG) %s(%s)::reload() -- Found ENDTRANS during roll-forward recovery.

Action: Information about stages of the LogStore initialization.

120044 7(DEBUG) %s(%s)::reload() -- Found a NORMAL record during roll-forward recovery.

Action: Information about stages of the LogStore initialization.

120045 7(DEBUG) %s(%s)::reload() -- Found BEGINCOMPACT record during roll-forward recovery.

Action: Information about stages of the LogStore initialization.

120046 2(CRIT) %s(%s)::reload() -- Found an unknown record type %d during roll-forward recovery, fatal error.

Action: An unknown record type found in the LogStore file means that the file is probably corrupted. The model cannot be started with such a file.

120047 2(CRIT) %s(%s)::dumpDebugData() -- last logstore write region [%ld,%ld), overlaps cleaner protected region: [%ld,%ld). The store is wedged, stopping.

Action: The specified Log Store size was too small and ran out of space. Monitor the Log Store usage and increase it in advance to prevent this fatal error. The store reserve is intended to guarantee against this situation, but in exceptional circumstances it still may happen. If the log store comes to this condition, it can not be resized any more. You must start with a new clean store (and increase its size too).

120048 2(CRIT) %s(%s)::getIndex(%s) -- failed, may have no more than %d indexes per store.

Action: A single LogStore may contain only a limited number of streams in it. Even though the limit is higher than would normally be seen, very large models may hit it. In such a case, split a LogStore in two. Also, if the store is used continuously with the changing models, where old streams are deleted and new streams created, the store still remembers the discarded streams. In this case, either start with a fresh file or use the online backup to move the active streams to a new file. If the streams are deleted during a Dynamic Service Modification, it does not leave the data behind.

120049 6(INFO) %s(%s)::backup -- successful, used %lu bytes.

Action: The backup request has succeeded.

120050 2(CRIT) %s(%s)::backup -- index '%s' has no stream associated to it.

Action: The backup has found data from a discarded stream in the store. (Such as if the model has changed and does not contain some stream any more). This data will be skipped during backup. The previous versions of the Aleri Streaming Platform refused to do the backup if such data was found.

120051 2(CRIT) %s(%s)::backup -- failed to copy a record in index '%s'.

Action: The backup procedure failed to copy the data, which means the whole backup

has failed. Possibly the process has run out of memory, as the backup temporarily needs to double its memory size.

120052 2(CRIT) %s(%s)::getIndex(%s) -- failed, out of UINT32_MAX ids; do a backup and restart from backed-up stores to reset the ids.

Action: In addition to the limited number of active stream indexes in the LogStore, each stream gets assigned a unique id. These ids are never reused even if the stream is properly disposed of using a dynamic modification. The limit on the numbed of ids is very large but frequent modifications on a very large model may still exhaust it. Use the online backup to reset the ids in the backed-up data.

120053 6(INFO) %s(%s)::PrepareMod() -- changing SWEEPAMOUNT to %ld bytes.

Action: The sweep amount is specified in the attribute "sweepamount", in percent of the file size. It's the size of data that gets processed during one pass of LogStore cleaning.

120054 4(WARNING) %s(%s)::reload() -- during roll-forward recovery, encountered a bad record, error: %s.

Action: The Log Store file was probably corrupted.

120055 3(ERR) %s(%s)::compact() -- nearing capacity, at %d%% free, %d%% live data; performance is being degraded.

Action: The volume of data is nearing the size of the Log Store. The cleaning is performed most often to conserve space. Stop the server and increase the store size in the model as soon as practical before the store has overflowed. This message may also appear after resizing, while the Platform is relocating the data to assimilate the new free space.

120056 4(WARNING) %s(%s)::readHeader() -- upgraded log version from %d.%d to %d.%d.

Action: When the Aleri Streaming Platform opens an existing LogStore of an older version, it transparently upgrades the format of the store to the newer version. The LogStore version numbers are not directly related to the Aleri Streaming Platform version numbers. Such an upgraded LogStore can not be opened by the old version of the Aleri Streaming Platform any more. To preserve the compatibility with the older versions, either make a backup copy of the LogStore in advance or dump the contents of the store into XML with the new Aleri Streaming Platform version and load the data back into the old platform version.

120057 4(WARNING) %s(%s)::initialize() -- specified sweepamount(%ld kb) exceeds 20%% of fullsize, reducing to 20%% of fullsize.

Action: The sweep amount is specified in the attribute "sweepamount", in percent of the file size. It's the size of data that gets processed during one pass of LogStore cleaning.

120058 4(WARNING) %s(%s)::initialize() -- specified sweepamount(%ld kb) below 5%% of fullsize, increasing to 5%% of fullsize.

Action: The sweep amount is specified in the attribute "sweepamount", in percent of the file size. It's the size of data that gets processed during one pass of LogStore cleaning.

120059 6(INFO) %s(%s)::initialize() -- using sweepAmount = %ld kb

Action: The sweep amount is specified in the attribute "sweepamount", in percent of the file size. It's the size of data that gets processed during one pass of LogStore cleaning.

120060 5(NOTICE) %s(%s)::initialize() -- setting metadata checpointing at %d records

Action: The checkpoint count specified in the attribute "ckcount" controls, how often the intermediate Log Store index cache is checkpointed to the disk (not to be confused with the full store checkpoints). The value is the count of modified index nodes collected in the cache, roughly equal to the number of records modified since the last such checkpoint. Setting it to 0 is equivalent to the old Log Store logic that flushed this cache after every transaction. The higher values improve the efficience of space use in the store and reduce the amount of cleaning. However the very large values have diminshed returns and will increase the memory use. The highest theoretically supported value is 2G but the values over 100000 are probably impractical.

120061 5(NOTICE) %s(%s)::initialize() -- setting the reserve size at %d%%.

Action: The Log Store needs a certain amount of its capacity reserved to maintain the efficiency of its operations. The attribute "reservePct" specifies the reserve capacity in percentage in comparison to the full store size. It can be set to a value between 10 and 40 and is 20 percent by default. If after all the possible cleaning the unused space in the log store falls below this amount, the Aleri Streaming Platform aborts. If that happens, increase the store size and restart, the Log Store will automatically grow.

120062 4(WARNING) %s(%s)::initialize() -- the reserve size specified too low at %d%%, changed to %d%%.

Action: The Log Store needs a certain amount of its capacity reserved to maintain the efficiency of its operations. The attribute "reservePct" specifies the reserve capacity in percentage in comparison to the full store size. It can be set to a value between 10 and 40 and is 20 percent by default. If after all the possible cleaning the unused space in the log store falls below this amount, the Aleri Streaming Platform aborts. If that happens, increase the store size and restart, the Log Store will automatically grow.

120063 4(WARNING) %s(%s)::initialize() -- the reserve size specified too high at %d%%, changed to %d%%.

Action: The Log Store needs a certain amount of its capacity reserved to maintain the efficiency of its operations. The attribute "reservePct" specifies the reserve capacity in percentage in comparison to the full store size. It can be set to a value between 10 and 40 and is 20 percent by default. If after all the possible cleaning the unused space in the log store falls below this amount, the Aleri Streaming Platform aborts. If that happens, increase the store size and restart, the Log Store will automatically grow.

120064 2(CRIT) %s(%s)::initialize() -- the reserve of %d%% would be required for the reliable work of this store, higher than the maximum of %d%%; increase the size of the store.

Action: The small stores are particularly sensitive to the size of reserve. An automatic check is done to ensure that the reserve is sufficient, and increased if it is found too low. If the automatic calculation results in the required reserve over 40 percent, the model won't start. Increase the size of the store and restart.

120065 4(WARNING) %s(%s)::initialize() -- the reserve of %d%% is too small for the reliable work of this store, increased to %d%%.

Action: The small stores are particularly sensitive to the size of reserve. An automatic check is done to ensure that the reserve is sufficient, and increased if it is found too low.

120066 2(CRIT) %s(%s)::compact() -- cleaning made no progress, the store is wedged, stopping.

Action: The specified Log Store size was too small and ran out of space. Monitor the Log Store usage and increase it in advance to prevent this fatal error. The store reserve is intended to guarantee against this situation, but in exceptional circumstances it still may happen. If the log store comes to this condition, it can not be resized any more. You must start with a new clean store (and increase its size too).

120067 4(WARNING) %s(%s)::compact() -- free space dipped into reserve by %d%%, scrubbing in attempt to reclaim more space.

Action: When the free space becomes very low, the Log Store logic attempts to reclaim more free space by repeating the compaction. If it scrubs through the whole store without finding more space, the Platform will stop. Increase the size of the store as soon as possible. This message may also appear after resizing, while the Platform is relocating the data to assimilate the new free space.

120068 4(WARNING) %s(%s)::initialize() -- the index '%s' has no matching stream in the configuration file, discarded.

Action: If the model was edited to remove some streams, on the next restart the data of these streams will be automatically discarded from the Log Store.

120069 4(WARNING) %s(%s)::~LogStore( ) -- at exit liveSize: %ld kB, %d%% of fullsize.

Action: At the exit time the log store reports its usage statistics.

121000 7(DEBUG) LogIndexNodeCache(%s)::add(k=%ld, l=%ld, r=%ld) -- addr=%ld

Action: The debugging information about the internals of a LogStore index.

121001 7(DEBUG) LogIndexNodeCache(%s)::commit() -- cache unchanged, returning root == %ld

Action: The debugging information about the internals of a LogStore index.

121002 7(DEBUG) LogIndexNodeCache(%s)::commit() -- writing %ld of %ld nodes.

Action: The debugging information about the internals of a LogStore index.

121003 7(DEBUG) LogStoreIndex(%s)::LogStoreIndex() -- created and attached to store '%s'.

Action: The debugging information about the internals of a LogStore index.

121004 7(DEBUG) LogStoreIndex(%s)::LogStoreIndex() -- reload (%ld records, %ld bytes, root=%ld, seq=%ld).

Action: The debugging information about the internals of a LogStore index.

121005 7(DEBUG) LogStoreIndex(%s)::~LogStoreIndex() -- disposing of LogStoreIndex, index storage: %ld Bytes, record storage: %ld Bytes, liveSize: %ld kB

Action: The debugging information about the internals of a LogStore index.

121018 7(DEBUG) LogStoreIndex(%s)::removeRoot(%ld) -- complex remove, rotating %ld.

Action: The debugging information about the internals of a LogStore index.

121019 7(DEBUG) LogStoreIndex(%s)::putDelete( ) -- failed, empty tree.

Action: The debugging information about the internals of a LogStore index.

121024 7(DEBUG) LogStoreIndex(%s)::get(%s) -- v. %s, %d.

Action: The debugging information about the internals of a LogStore index.

121025 7(DEBUG) LogStoreIndex(%s)::get(%s) -- key match '%s'.

Action: The debugging information about the internals of a LogStore index.

121026 7(DEBUG) LogStoreIndex(%s)::get(%s) -- no matching record found.

Action: The debugging information about the internals of a LogStore index.

121027 7(DEBUG) LogStoreIndex(%s)::checkpoint() -- checkpointing (%ld records, %ld bytes).

Action: The debugging information about the internals of a LogStore index.

121030 7(DEBUG) LogStoreIndex(%s)::beginTransaction()

Action: The debugging information about the internals of a LogStore index.

121031 7(DEBUG) LogStoreIndex(%s)::commitTransaction()

Action: The debugging information about the internals of a LogStore index.

121032 7(DEBUG) LogStoreIndex(%s)::rollbackTransaction()

Action: The debugging information about the internals of a LogStore index.

121033 7(DEBUG) LogStoreAccessor() -- attached to index '%s' with root '%ld'.

Action: The debugging information about the internals of a LogStore index.

121034 7(DEBUG) LogStoreAccessor() -- destroying accessor attached to index '%s'

Action: The debugging information about the internals of a LogStore index.

121035 7(DEBUG) LogStoreAccessor(%s)::hasNext() -- %s

Action: The debugging information about the internals of a LogStore index.

121036 7(DEBUG) LogStoreAccessor(%s)::getNext() -- no more records.

Action: The debugging information about the internals of a LogStore index.

121037 7(DEBUG) LogStoreAccessor(%s)::getNext() -- addr=%ld, node[k=%ld, l=%ld, r=%ld]

Action: The debugging information about the internals of a LogStore index.

121038 7(DEBUG) LogStoreAccessor(%s)::get(%s) -- record md5 '%s'.

Action: The debugging information about the internals of a LogStore index.

121039 7(DEBUG) LogStoreAccessor(%s)::get(%s) -- no matching record found.

Action: The debugging information about the internals of a LogStore index.

122000 7(DEBUG) %s(%s)::initialize() -- initializing.

Action: The debugging information about the internals of a memory store.

122001 7(DEBUG) %s(%s)::~%s() -- disposing.

Action: The debugging information about the internals of a memory store.

122002 6(INFO) %s(%s)::getIndex() -- setting up index '%s'.

Action: Information about stages of the memory store initialization.

122004 7(DEBUG) MemoryStoreIndex(%s)::MemoryStoreIndex() -- attached to store '%s'.

Action: The debugging information about the internals of a memory store.

122005 7(DEBUG) MemoryStoreIndex(%s)::~MemoryStoreIndex() -- disposing of MemoryStoreIndex attached to store '%s'.

Action: The debugging information about the internals of a memory store.

122006 7(DEBUG) MemoryStoreIndex(%s): dumping.

Action: The debugging information about the internals of a memory store.

122007 7(DEBUG) MemoryStoreIndex(%s)::dump -- key md5 '%s', record '0x%x'.

Action: The debugging information about the internals of a memory store.

122010 7(DEBUG) MemoryStoreIndex(%s)::getAccessor() -- making new accessor.

Action: The debugging information about the internals of a memory store.

122011 2(CRIT) StoreIndex()::commitTransaction() -- got unexpected operation %d.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

122012 7(DEBUG) MemoryStoreList(%s)::MemoryStoreList() -- attached to store '%s'.

Action: The debugging information about the internals of a memory store.

122013 7(DEBUG) MemoryStoreList(%s)::~MemoryStoreList() -- disposing of MemoryStoreList attached to store '%s'.

Action: The debugging information about the internals of a memory store.

122014 7(DEBUG) MemoryStoreList(%s): dumping.

Action: The debugging information about the internals of a memory store.

122015 7(DEBUG) MemoryStoreList(%s)::dump -- key md5 '%s', record '0x%x'.

Action: The debugging information about the internals of a memory store.

122016 2(CRIT) MemoryStoreList(%s)::putUpdate() -- bad call

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

122017 2(CRIT) MemoryStoreList(%s)::putDelete() -- bad call

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which

should never happen. Contact Aleri support if this message appears.

122018 7(DEBUG) MemoryStoreList(%s)::getAccessor() -- making new accessor.

Action: The debugging information about the internals of a memory store.

122019 2(CRIT) MemoryStoreListAccessor(%s)::get() -- calling get.

Action: The debugging information about the internals of a memory store.

123000 2(CRIT) StoreIndex(%s)::collapseTransaction() -- got unexpected operation %d.

Action: This error may happen if the SPLASH code in a FlexStream sets an invalid value for the operation code. Check your SPLASH code.

123001 4(WARNING) StoreIndex(%s)::put() bad insert, tid=%d.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123002 4(WARNING) StoreIndex(%s)::put() bad update, tid=%d.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123003 4(WARNING) StoreIndex(%s)::put() bad upsert, tid=%d.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123004 4(WARNING) StoreIndex(%s)::put() bad delete, tid=%d.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123005 2(CRIT) StoreIndex(%s)::put() -- got unexpected operation %d.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123006 4(WARNING) StoreIndex(%s)::put() -- roll back transaction of size %d.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123007 4(WARNING) Bad insert writing to store.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123008 4(WARNING) Bad update writing to store.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123009 4(WARNING) Bad upsert writing to store.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123010 4(WARNING) Bad delete.writing to store.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123011 4(WARNING) StoreIndex(%s)::collapse() Error collapsing transaction: op=%d, oldop=%d, tid=%d.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123012 4(WARNING) Bad collapse.

Action: An invalid data sequence was encountered. Check the input data and the model for correctness. If enabled, more detail about the offending records has been written to the bad records file. To enable the bad records file, use the option -B.

123013 6(INFO) %s(%s)::initialize() indexSizeHint set to %d.

Action: Information about the index size hint set in the model. The hint lets the stream index preallocate space and require less resizing as the model works. It improves the speed and latency.

124001 6(INFO) %s(%s)::Memory usage: %lu bytes in aggregation index.

Action: Statistics about memory usage in the aggregation index. Each AggregateStream keeps a copy or reference to all input data in its aggregation index.

124002 2(CRIT) %s(%s)::Bad expression '%s' encountered: %s

Action: A syntax error in the aggregation expression. Check and correct the model.

124004 2(CRIT) %s(%s)::init() number of Group clauses does not match number of key columns.

Action: The aggregation creates a new primary index on the data. The grouping guarantees the uniqueness on this index, so the group clauses must match the index counts. Check and correct the model.

124007 5(NOTICE) %s(%s)::init() optimizing for additive case.

Action: A notice that the additive optimization is to be used on the data. The additive optimization is possible if the aggregation expressions can be recalculated by looking strictly at one added, updated or deleted record, without iterating through all the records in the group.

124008 5(NOTICE) %s(%s)::init() CANNOT optimize, aggrgation is not additive.

Action: An notice that the additive optimization is not to be used on the data. The additive optimization is possible if the aggregation expressions can be recalculated by looking strictly at one added, updated or deleted record, without iterating through all the records in the group.

124010 2(CRIT) AggregateStream(%s): Encountered fatal error in update during delete phase.

Action: An illegal calculation has happened in the model. Check the model logic and received data.

124011 4(WARNING) AggregateStream(%s): Discarding UPDATE---not valid for AggregateStream.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

124012 4(WARNING) AggregateStream(%s): Discarding UPSERT---not valid for AggregateStream.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

124013 2(CRIT) %s(%s)::init() The rule for Group %d has an aggregation operation (group-by clauses must not use aggregation).

Action: Since the grouping is done before aggregation, the grouping conditions may not use the aggregation operations. Check and correct the model.

124014 2(CRIT) %s(%s)::init() The rule for GroupFilter %d has an aggregation operation (group-by clauses must not use aggregation).

Action: Since the grouping is done before aggregation, the grouping conditions may not use the aggregation operations. Check and correct the model.

124015 2(CRIT) %s(%s)::init() The rule for GroupOrder %d has an aggregation operation (group-by clauses must not use aggregation).

Action: Since the grouping is done before aggregation, the grouping conditions may not use the aggregation operations. Check and correct the model.

124017 2(CRIT) %s(%s)::error in compilation of GroupFilter %d: %s

Action: A syntax error in the expression. Check and correct the model.

124018 2(CRIT) %s(%s)::error in compilation of GroupOrder %d: %s

Action: A syntax error in the expression. Check and correct the model.

124019 4(WARNING) AggregateStream(%s): Discarding UPDATE; record no longer present (tid=%d).

Action: This error may happen if an InputWindow is defined on this stream. For correct usage of InputWindows, the input data must be insert-only. Check and correct the model or the input data.

124020 4(WARNING) AggregateStream discarding UPDATE; record no longer present.

Action: This error may happen if an InputWindow is defined on this stream. For cor-

rect usage of InputWindows, the input data must be insert-only. Check and correct the model or the input data.

124021 2(CRIT) %s(%s)::init() no key columns specified; need at least one key column.

Action: Each stream must have at least one key column defined. Check and correct the model.

125003 4(WARNING) SourceStream(%s): Discarding UPDATE---not valid for SourceStream with insertOnly (tid=%d).

Action: A stream defined as insert-only may not receive the updates or deletes. Check and correct the input data.

125004 4(WARNING) SourceStream(%s): Discarding UPSERT---not valid for SourceStream with insertOnly (tid=%d).

Action: A stream defined as insert-only may not receive the updates or deletes. Check and correct the input data.

125005 4(WARNING) SourceStream(%s): Discarding DELETE---not valid for SourceStream with insertOnly (tid=%d).

Action: A stream defined as insert-only may not receive the updates or deletes. Check and correct the input data.

125007 4(WARNING) SourceStream discarding UPDATE---not valid for SourceStream with insertOnly .

Action: A stream defined as insert-only may not receive the updates or deletes. Check and correct the input data.

125008 4(WARNING) SourceStream discarding UPSERT---not valid for SourceStream with insertOnly .

Action: A stream defined as insert-only may not receive the updates or deletes. Check and correct the input data.

125009 4(WARNING) SourceStream discarding DELETE---not valid for SourceStream with insertOnly .

Action: A stream defined as insert-only may not receive the updates or deletes. Check and correct the input data.

126000 2(CRIT) SourceStream(%s)::initInputs() error in reading file %s.

Action: Obsolete.

126001 6(INFO) %s(%s)::run() -- starting event queue Stream.

Action: Information about the internal stages of initialization.

126002 4(WARNING) SourceStream(%s): Discarding INSERT; record has null key (tid=%d).

Action: The key fields may not be NULL. Check and correct the input data.

126003 4(WARNING) SourceStream(%s): Discarding UPDATE; record has null key (tid=%d).

Action: The key fields may not be NULL. Check and correct the input data.

126004 4(WARNING) SourceStream(%s): Discarding UPSERT; record has null key (tid=%d).

Action: The key fields may not be NULL. Check and correct the input data.

126005 4(WARNING) SourceStream(%s): Discarding DELETE; record has null key (tid=%d).

Action: The key fields may not be NULL. Check and correct the input data.

126006 4(WARNING) SourceStream discarding INSERT; record has null key.

Action: The key fields may not be NULL. Check and correct the input data.

126007 4(WARNING) SourceStream discarding UPDATE; record has null key.

Action: The key fields may not be NULL. Check and correct the input data.

126008 4(WARNING) SourceStream discarding UPSERT; record has null key.

Action: The key fields may not be NULL. Check and correct the input data.

126009 4(WARNING) SourceStream discarding DELETE; record has null key.

Action: The key fields may not be NULL. Check and correct the input data.

126010 2(CRIT) %s(%s): The autogen column '%s' does not have type 'int64' in the row definition.

Action: The auto-generated sequence numbers are always 64-bit. Check and correct the model.

126011 6(INFO) %s(%s): The autogen column will start at %lld.

Action: Informational message. The logic checks the existing data in the stream and chooses the next sequence number to be larger.

127001 2(CRIT) %s(%s)::error in compilation of ColumnExpressions: %s

Action: A syntax error in the expression. Check and correct the model.

127002 2(CRIT) %s(%s)::ColumnExpression %d has bad expression '%s': %s.

Action: A syntax error in the expression. Check and correct the model.

127003 2(CRIT) %s(%s)::must have exactly one input stream

Action: The ComputeStream processes the data from one stream. Check and correct the model.

127004 2(CRIT) %s(%s)::the number of keys in the input stream be <= the number of keys of the output

Action: The ComputeStream cannot change the key of the data. It may add new fields to the key but no fields may be removed. Check and correct the model.

127005 2(CRIT) %s(%s)::all keys of the input must be copied into the keys of the output.

Action: The ComputeStream cannot change the key of the data. It may add new fields to the key but no fields may be removed. Check and correct the model.

127006 2(CRIT) %s(%s) ColumnExpression for a key column does not refer to a valid input table.

Action: The ComputeStream cannot change the key of the data. It may add new fields to the key but no fields may be removed. Check and correct the model.

127007 2(CRIT) %s(%s) ColumnExpression for a key column does not refer to a valid column in the input table.

Action: The ComputeStream cannot change the key of the data. It may add new fields to the key but no fields may be removed. Check and correct the model.

127008 2(CRIT) %s(%s) ColumnExpression for a key column refers to the same column as another key column rule.

Action: The ComputeStream cannot change the key of the data. It may add new fields to the key but no fields may be removed. Check and correct the model.

127009 2(CRIT) %s(%s)::ColumnExpression %d has aggregate operation, which is not valid in ComputeStream.

Action: The aggregate operations may be used only in the AggregateStreams. Check and correct the model.

127010 4(WARNING) ComputeStream(%s): Discarding UPSERT---not valid for ComputeStream.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

127015 2(CRIT) %s(%s): %s syntax error in the type '%s'.

Action: A syntax error in the expression. Check and correct the model.

127017 2(CRIT) %s(%s): %s type '%s' can not be resolved: %s.

Action: A syntax error in the expression. Check and correct the model.

127019 2(CRIT) %s(%s): ColumnExpression %d is not of base type.

Action: A syntax error in the expression. Check and correct the model.

128000 2(CRIT) %s(%s)::FilterStream requires exactly one input stream

Action: The FilterStream processes the data from one stream. Check and correct the model.

128004 2(CRIT) %s(%s)::error in compilation of FilterExpression: %s

Action: A syntax error in the expression. Check and correct the model.

128005 2(CRIT) %s(%s)::Bad expression '%s' encountered: %s

Action: A syntax error in the expression. Check and correct the model.

128006 4(WARNING) FilterStream(%s): Discarding UPSERT---not valid for FilterStream.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

129017 2(CRIT) %s(%s)::init() found no Join definitions

Action: A JoinStream must have a join condition defined. Check and correct the model.

129018 4(WARNING) JoinStream(%s): Discarding UPSERT---not valid for JoinStream

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

129019 2(CRIT) %s(%s): Input stream %s to Join expression not found

Action: A syntax error in the expression. Check and correct the model.

129020 2(CRIT) %s(%s): Input field %s in Join expression not found in stream %s

Action: A syntax error in the expression. Check and correct the model.

129021 2(CRIT) %s(%s): Types of the fields %s, %s in Join expression do not match

Action: The join expression matches together two columns from different streams. These columns must have the same type.

129024 2(CRIT) %s(%s): Expression for a key column does not refer to a valid column in the input table

Action: A syntax error in the expression. Check and correct the model.

129025 2(CRIT) %s(%s): Expression for a key refers to a non-key of an input table

Action: The key of the resulting join must follow certain rules. It may include key columns from more than one original stream. No column may be used more than once. All the columns from at least one original stream must be used to guarantee its uniqueness. Each individual field of the result key may not refer more than one field of an original stream's key, also to guarantee the uniqueness. Check and correct the model.

129026 2(CRIT) %s(%s): Expression for a key refers to an input key column already seen

Action: The key of the resulting join must follow certain rules. It may include key columns from more than one original stream. No column may be used more than once. All the columns from at least one original stream must be used to guarantee its uniqueness. Each individual field of the result key may not refer more than one field of an original stream's key, also to guarantee the uniqueness. Check and correct the model.

129027 2(CRIT) %s(%s): At least one input table must have all keys copied into key rules

Action: The key of the resulting join must follow certain rules. It may include key columns from more than one original stream. No column may be used more than once. All the columns from at least one original stream must be used to guarantee its uniqueness. Each individual field of the result key may not refer more than one field of an original stream's key, also to guarantee the uniqueness. Check and correct the model.

129028 2(CRIT) %s(%s): Expression for key refers to the same input table more than once

Action: The key of the resulting join must follow certain rules. It may include key columns from more than one original stream. No column may be used more than once. All the columns from at least one original stream must be used to guarantee its uniqueness. Each individual field of the result key may not refer more than one field of an original stream's key, also to guarantee the uniqueness. Check and correct the model.

129029 2(CRIT) %s(%s): Could not determine a join strategy; try reordering the input streams.

Action: When joining more than two streams, determining the order of joining may become complicated. Try to change the order of streams in the join, or split one join into

multiple sequential joins, with fewer streams joined in each of them.

130001 2(CRIT) RowDefinition(%s): empty column name

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

130002 2(CRIT) RowDefinition(%s): unknown type '%s'

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

130003 2(CRIT) RowDefinition(%s): unexpected '%s' in place of literal KEY

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

130004 2(CRIT) RowDefinition(%s): unexpected character '%c'

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

130005 2(CRIT) RowDefinition(%s)::pack(): internal error, %d values passed, %d expected.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

131000 2(CRIT) %s(%s)::initInputs() cannot find input stream %s.

Action: Probably the name of an input stream is mistyped. Check and correct the model.

131004 2(CRIT) %s(%s)::init() Configuration file contains a cycle of streams involving this stream.

Action: In AleriML the streams may not be connected in a loop. Check and correct the model.

131005 2(CRIT) Stream(%s)::initStream() -- thread start failed when starting data retention

Action: The limit of threads per process is probably exhausted. Consult your system administrator.

131006 7(DEBUG) %s(%s)::handleEvent() -- exit requested, no more feeders.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

131007 7(DEBUG) %s(%s)::handleEvent() -- exit requested %d feeders still running.

Action: The exit request is propagated through the model from the source streams to the derived streams. Each stream must collect the exit requests from all inputs before shutting down. This message provides the information on the progress of shutdown.

131008 6(INFO) %s(%s)::run() -- starting event queue Stream.

Action: Information about the internal stages of initialization.

131009 6(INFO) %s(%s)::run() -- starting thread with thread id %d

Action: Information about the internal stages of initialization.

131010 7(DEBUG) %s(%s)::run() -- sending completion notification to %s.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

131011 7(DEBUG) %s(%s)::dumpXML() -- writing out records.

Action: If the stream has tile attribute "ofile", its contents gets automatically dumped into an XML file when the model exits.

131012 3(ERR) %s(%s): Divide-by-zero.

Action: An illegal calculation has happened in the model. Check the model logic and received data.

131013 3(ERR) %s(%s): Floating-point exception.

Action: An illegal calculation has happened in the model. Check the model logic and received data.

131014 2(CRIT) %s(%s): Illegal record operation %d in 0x%x.

Action: An illegal calculation has happened in the model. Check the model logic and received data.

131015 6(INFO) %s(%s)::regenerate() -- regenerating stream.

Action: When the model starts, it regenerates the contents of the streams to a consistent state, based on the contents of the persistent stores. After a dynamic modification, the affected streams are also regenerated.

131016 6(INFO) %s(%s)::regenerate() -- done regenerating stream.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

131017 6(INFO) %s(%s)::stats: msec %d qsize %d rtrans %d ttrans %d rops %d tops %d streamsize %d cpu %4.1f%%.

Action: The periodic statistics about the stream, a subset of those provided by sp_monitor. The Aleri Streaming Platform must be started with option -t to enable the monitoring of statistics. The values: msec - time since the last report, in milliseconds; qsize - current queue size; rtrans - number of transactions processed since the last update; ttrans - total number of transactions processed; rops - number of rows processed since the last update; ttrans - tops number of rows processed; streamsize - current number of rows in stream's store; cpu - CPU usage since the last report, percent.

131018 6(INFO) %s(%s)::signature is %s.

Action: It is an informational message. The signature is like a checksum of the stream's schema. The signature is checked when loading data from the persistent stores to make sure that the schema did not change.

131019 7(DEBUG) %s(%s)::Inject orphaned records, opCode: %d, record: %s

Action: The orphaned records are the source stream records that have been written into the LogStore but not checkpointed yet. On the Aleri Streaming Platform restart, it is played back to restore the state.

131020 7(DEBUG) %s(%s)::Registered a cluster export stream, total export streams: %d

Action: Another node of the cluster has subscribed to this stream.

131021 7(DEBUG) %s(%s)::Unregistered a cluster export stream, total export streams: %d

Action: Another node of the cluster has unsubscribed from this stream.

131022 2(CRIT) %s(%s):: stream references a store <%s> that is not in the running module.

Action: A cluster node must be self-contained, with all streams and stores wrapped inside this node's Module. Check and correct the model.

131022 7(DEBUG) %s(%s):: waiting for distributed subscribers to disconnect before exiting.

Action: In a clustered model, all the nodes wait for the other nodes to process the exit instructions and disconnect. This makes sure that all data has been processed by the whole model.

131023 7(DEBUG) %s(%s):: exit posted to all subscribers.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

131024 2(CRIT) %s(%s)::%s received invalid opcode: %s on record, exiting.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

131025 2(CRIT) %s(%s)::initMetadata() both expiryTime and expiryField must be specified in config file.

Action: The expiry-related attributes must be either both present or absent. Check and correct the model.

131026 2(CRIT) %s(%s)::initMetadata() expiryField must be valid column name

Action: The attribute "expiryField" must identify an int32 column in the data. Check and correct the model.

131027 2(CRIT) %s(%s)::initMetadata() expiryField must be an int32 column

Action: The attribute "expiryField" must identify an int32 column in the data. Check and correct the model.

131030 2(CRIT) %s(%s)::initMetadata() expiryTimeField must be valid column name

Action: The attribute "expiryTimeField" must identify a date column in the data. Check and correct the model.

131031 2(CRIT) %s(%s)::initMetadata() expiryTimeField's type must be "date"

Action: The attribute "expiryTimeField" must identify a date column in the data. Check and correct the model.

131032 6(INFO) %s(%s)::pushToHotSpare() _gatewayClient is NULL, creating new object.

Action: In a Hot Spare cluster configuration, the connection to the secondary platform is created when the first data record comes in. The connection is also re-established if it was lost.

131033 4(WARNING) %s(%s)::pushToHotSpare() _gatewayClient unable to open connection, dis-

abling redundancy.

Action: If a Hot Spare cluster configuration can not establish connection to the secondary platform, it reverts to running in the non-redundant configuration. Check whether the secondary platform is running.

131034 4(WARNING) %s(%s)::pushToHotSpare() _gatewayClient authentication failed, disabling redundancy.

Action: If a Hot Spare cluster configuration can not establish connection to the secondary platform, it reverts to running in the non-redundant configuration. Check the credentials specified with the command-line option -a or -K.

131035 4(WARNING) %s(%s)::pushToHotSpare() _gatewayClient send message failed, disabling redundancy.

Action: If a Hot Spare cluster configuration can not establish connection to the secondary platform, it reverts to running in the non-redundant configuration. Check whether the secondary platform is running.

131036 2(CRIT) %s(%s)::Bad access control specification: '%s' is not a pair of strings separated by ':'.

Action: The attribute "restrictAccess" has been misformatted. It must contain a space-separated list of colon-separated pairs. Check the model and correct it.

131037 2(CRIT) %s(%s)::Bad access control specification: unknown access control type in pair '%s'.

Action: The attribute "restrictAccess" has been misformatted. It must contain a space-separated list of colon-separated pairs. Check the model and correct it.

131038 4(WARNING) Stream(): error occurred in computation of row.

Action: An illegal calculation has happened in the model. Check the model logic and received data.

131039 6(INFO) %s(%s): Collecting statistics (this could take awhile).

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

131040 6(INFO) %s(%s): Memory usage: %lu bytes in %lu records.

Action: Statistics about the amount of memory used by the stream. This does not include any internal indexing overhead. This information can be used to plan the LogStore sizes for the future runs of the Aleri Streaming Platform.

131041 2(CRIT) %s(%s):: can not find store %s.

Action: The store name specified for this stream is not present in the model. Check and correct the model.

131042 2(CRIT) %s(%s)::initMetadata() expiryTime value '%s' is not a valid integer >=0.

Action: The attribute "expiryTime" must specify the expiry step in seconds. Check and correct the model.

131043 2(CRIT) %s(%s)::initMetadata() expiryMaxValue '%s' is not a valid integer >=0.

Action: The attribute "expiryMaxValue" specifies the maximum number of expiry

steps that can be done on a record. Check and correct the model.

131044 2(CRIT) %s(%s)::wipeout() -- started wiping out the old contents.

Action: After a dynamic modification, if the data in the stream is about to be regenerated, the old data must be wiped out first.

131045 2(CRIT) %s(%s)::wipeout() -- completed.

Action: After a dynamic modification, if the data in the stream is about to be regenerated, the old data must be wiped out first.

131046 2(CRIT) %s(%s): received an unexpected input regeneration report.

Action: The regeneration may happen only at certain moments in the life of the stream: either when the model gets started or after a dynamic modification. A regeneration request at any other time means that something went wrong in the Aleri Streaming Platform. Contact the Aleri support.

131047 2(CRIT) %s(%s): regeneration completed

Action: Information about the internal stages of initialization.

131048 2(CRIT) %s(%s): Join/Flex do not currently support Input Windows.

Action: JoinStreams and FlexStreams do not support InputWindows. A FlexStream may use eventCaches instead, achieving higher efficiency as well. A ComputeStream with an InputWindow may be added before either FlexStream or JoinStream. Check and correct the model.

131049 2(CRIT) %s(%s): specifying an input stream for an Input Window of a source stream is not allowed.

Action: Since the Source Stream receives data directly from the outside sources, its InputWindow may act only on this data. Check and correct the model.

131050 2(CRIT) %s(%s): a Source Stream can only have one Input Window.

Action: Since the Source Stream receives data directly from the outside sources, its InputWindow may act only on this data. Check and correct the model.

131051 2(CRIT) %s(%s): an Input Window on a derived stream must have a stream attribute set to one of its input streams.

Action: An InputWindow is associated with one input stream, no more than one InputWindow per input stream. The "stream" attribute of the InputWindow must identify this stream. Check and correct the model.

131052 2(CRIT) %s(%s): multiple Input Windows on a derived stream for the same input stream is illegal.

Action: An InputWindow is associated with one input stream, no more than one InputWindow per input stream. The "stream" attribute of the InputWindow must identify this stream. Check and correct the model.

131053 2(CRIT) %s(%s): the stream attribute of an Input Window does not match any input streams.

Action: An InputWindow is associated with one input stream, no more than one In-

putWindow per input stream. The "stream" attribute of the InputWindow must identify this stream. Check and correct the model.

131054 2(CRIT) %s(%s): for logstores, only SourceStreams and CopyStreams may specify Input Windows.

Action: The LogStore consistency relies on the input being injected only in the source streams or copy streams of a cluster. Since the InputWindows produce the deletion records in place, it may not be used on the streams with data in LogStores, unless these streams mare one of these cases. Check and correct the model.

131055 2(CRIT) %s(%s): dynamic modification of Streams with Input Window is not permitted in 'noregen' mode.

Action: The option "noregen" bypasses the regeneration after the dynamic modification. It expects that the user knows that this would not introduce a major discrepancy in the data or accept such a discrepancy, as a tradeoff for a faster modification. The InputWindows have major issues if the data is not properly regenerated, so any streams with InputWindows may not be modified in the "noregen" mode. Retry the modification without the option "noregen" (or with the explicit option "regen").

131056 2(CRIT) %s(%s): Insert-only streams (SourceStream) may not specify an Input Window.

Action: Since the InputWindows generate the delete records, any stream with them is no longer insert-only. However for the InputWindows to work correctly, the data coming into them must be insert-only. Check and correct the model.

131057 6(INFO) %s(%s): will be recompiled, because InsertOnly state change has propagated.

Action: Informational message. On a dynamic modification, changing the InsertOnly state of one stream may propagate to the streams derived from it. The streams that have InsertOnly state changed must be recompiled, even though they had no explicit changes in them. This is detected and handled automatically.

131058 2(CRIT) %s(%s): InsertOnly state change has propagated, not allowed without 'regen' option.

Action: The option "noregen" bypasses the regeneration after the dynamic modification. It expects that the user knows that this would not introduce a major discrepancy in the data or accepts such a discrepancy, as a tradeoff for a faster modification. The InsertOnly mode change without regeneration may cause major issues in the stream's logic, so this combination is not allowed. Retry the modification without the option "noregen" (or with the explicit option "regen").

131059 2(CRIT) %s(%s): attribute "%s" value is incompatible in the old and new configuration.

Action: See the Administrator's Guide for the information on the Dynamic Services. The incompatible modifications require the whole stream ot be destroyed and recreated with the same name but possibly a different schema. Since it may be too disruptive, by default (or with option "compat") such dynamic modifications are not allowed, to prevent an accidental disruption. This message identifies the incompatible changes in the model. If the resulting modification gets refused, either correct the new model to prevent the incompatible changes or do the modification with the option "nocompat".

131060 2(CRIT) %s(%s): input stream count can not be changed, is: %d, requested: %d.

Action: Obsolete.

131061 4(WARNING) %s(%s): Previous incarnation %s(%s) had an incompatible rowdef, data will be discarded.

Action: A source stream, unlike the derived streams, can not have its data regenerated. If a source stream is modified while keeping its schema the same, or if a conversion model is provided to convert the data, the data may be moved from the old model. But otherwise on an incompatible modification, the data can not be moved and is discarded.

131062 6(INFO) %s(%s): moving the data from the previous incarnation.

Action: If a source stream is modified in an incompatible way while keeping its schema the same, the data will be moved from the old model.

131063 2(CRIT) %s(%s): the data move has failed, discarding data.

Action: If a source stream is modified in an incompatible way while keeping its schema the same, the data will be moved from the old model. However the new model may specify a different primary key. If the old data do not fit the new unique key, the move is abandoned and data discarded.

131064 2(CRIT) %s(%s): can not set %s(%s) as data source, row definitions or keys differ.

Action: When using a conversion model, at the points where the data enters and leaves the conversion model, the stream schema must be the same to transfer the data between the models. Check and correct the conversion model.

131065 2(CRIT) %s(%s): a non-source stream can not source data from %s(%s).

Action: A conversion model may receive its data only in its source streams. It may provide data only for the source streams of the new model. Check and correct the conversion model.

131066 6(INFO) %s(%s): sourcing the data from %s(%s).

Action: A conversion model has been specified, and this message informs of the connection points between the models.

131067 6(INFO) %s(%s): added breakpoint %d.

Action: Informational message about a breakpoint being added.

131068 6(INFO) %s(%s): deleted breakpoint %d.

Action: It is informational message about a breakpoint being deleted.

131069 6(INFO) %s(%s): triggered breakpoint %d.

Action: It is informational message about a breakpoint being triggered. The model is paused.

131070 6(INFO) %s(%s): triggered a break on exception.

Action: It is informational message about an exception in a stream triggering the model pause.

131071 4(WARNING) %s(%s)::expiryTimeField is null; defaulting to the current time for this record.

Action: The attribute "expiryTimeField" may be used to specify the column in a row

that carries the information about the logical last modification time of the row. The expiry is counted started from that time instead of the real time when the row was last updated in the stream. If the value in this column is NULL, the logic reverts to using the real time. If this is not intended, check the input data and the model for the reasons why that NULL appeared.

131072 2(CRIT) %s(%s): %s %s is invalid: %s.

Action: The Connection could not be created. The most likely reason is the shared library for this connection type can not be loaded. Examine the preceding messages for the possible reasons that led to it. Check your configuration.

131073 6(INFO) %s(%s)::processService(): received START_SYNC when not empty

Action: It is an informational message. A connector or other source started the resynchronization of the stream contents, replacing the whole old contents with the new ones.

131074 6(INFO) %s(%s)::processService(): END_SYNC %d records of %d differ, posting diffs to sync stream.

Action: It is an informational message. A connector or other source completed the resynchronization of the stream contents, replacing the whole old contents with the new one. The stream compares the new contents with the old one, and passes through only the differences. The message provides the summary of the differences.

131075 7(DEBUG) %s(%s)::processService(): record diff: opCode: %d, record: %s

Action: Debugging message. Provides information about the ongoing stream resynchronization.

131076 3(ERR) %s(%s): Error in running initializers; see next message for details

Action: The initialization of the global or local variables has failed. The model does not start.

131077 7(DEBUG) %s(%s):: waiting for connectors to finish before exiting.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

131078 2(CRIT) %s(%s): illegal stream; no keys defined.

Action: A stream must have at least one key field defined. Check and correct the model.

131079 6(INFO) %s(%s)::dynamic changes with InConnection running are treated as incompatible

Action: If a source stream has a running InConnection, changing it in place is unsafe. Instead the stream will be re-created from scratch and the input connection will be restarted.

131080 6(INFO) %s(%s)::dynamic changes in 'noregen' mode with local variables are treated as incompatible

Action: The option "noregen" bypasses the regeneration after the dynamic modification. It expects that the user knows this would not introduce a major discrepancy in the data or accepts such a discrepancy, as a tradeoff for a faster modification. This cannot be safely applied to the streams with local variables, so any changes to these streams are treated as incompatible, effectively preventing them from being done in the "noregen" mode.

131081 2(CRIT) %s(%s): illegal stream; duplicate column '%s'.\n

Action: The column names in a stream must be unique. Check and correct the model.

132000 3(ERR) StreamExport(%s):: dropped a subscriber (via processed message).

Action: Informational message that a client subscribed to this stream has disconnected.

132001 4(WARNING) StreamExport::Got a {START,END,RECORD} or BASE data for an unknown client.

Action: A self-testing assertion in the platform has failed. However this is a minor point.

132002 4(WARNING) StreamExport::Got an unknown message type.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

132003 6(INFO) StreamExport(%s)::put() message queue at 99%% of limit.

Action: If any of this stream's subscribers is slow, that subscriber's gateway queue will fill and then the export queue of this stream will start filling, until it becomes full. Check the clients and sp_monitor data to locate the slow client.

132004 6(INFO) StreamExport(%s)::put() max(size(all seen blocks)) now: %d.

Action: Information about the biggest transaction size produced by this stream.

132005 6(INFO) StreamExport(%s)::put() message queue dropped below 51%% of limit.

Action: If any of this stream's subscribers is slow, that subscriber's gateway queue will fill and then the export queue of this stream will start filling, until it becomes full. This message means that the client continued consuming the data, and the operation is returning to normal.

134000 2(CRIT) %s(%s)::UnionStream requires at least one input stream.

Action: The UnionStream needs one or more input streams. Check and correct the model.

134001 2(CRIT) %s(%s)::All inputs must have matching fields and field types.

Action: A UnionStream merges the records with the same schema from multiple streams, including the exact same primary key. It can not handle the records with different schemas. Check and correct the model.

134002 2(CRIT) %s(%s)::All inputs must have the same number of key columns as the UnionStream.

Action: A UnionStream merges the records with the same schema from multiple streams, including the exact same primary key. It can not handle the records with different schemas. Check and correct the model.

134003 2(CRIT) %s(%s)::The key columns of the inputs must match that of the UnionStream.

Action: A UnionStream merges the records with the same schema from multiple streams, including the exact same primary key. It can not handle the records with different schemas. Check and correct the model.

135000 2(CRIT) Platform()::main() -- fatal error in thread %d; aborting.

Action: Some major crash has happened in the Aleri Streaming Platform. Ideally, this should never happen. If it happens, collect the log file, the core dump and the Aleri Streaming Platform binaries and contact the Aleri support. If you can provide a way to reproduce this crash, preferably with a smaller model and data set, it would be very helpful and lead to a quicker fix.

135001 2(CRIT) Platform::main() Could not initialize SSL layer.

Action: The SSL can not be initialized. Probably there is some issue with the SSL key files specified in the option -e. Check whether the path exists and whether the permissions are correct. If the path is relative, check that the Aleri Streaming Platform's current directory is what you think it is.

135002 1(ALERT) Platform::main() Could not find configuration file.

Action: The configuration file specified in the option -f can not be read. Check whether the path exists and whether the permissions are correct. If the path is relative, check that the Aleri Streaming Platform's current directory is what you think it is.

135003 2(CRIT) Platform::main() Configuration file is not a regular file.

Action: The configuration file specified in the option -f must be a regular file, not a pipe or device or directory.

135004 6(INFO) Platform()::main() -- loading configuration file '%s'.

Action: Information about the internal stages of initialization.

135005 4(WARNING) Platform()::asap_badrec_init() -- could not open bad records file'%s' for writing.

Action: The bad records file specified in the option -B can not be opened for writing. Check whether the path exists and whether the permissions are correct. If the path is relative, check that the Aleri Streaming Platform's current directory is what you think it is.

135006 2(CRIT) Platform::main() -- port %d for SQL interface is outside valid range 0<= p < 2^16.

Action: The TCP protocol allows 16 bits for the port number. Check and correct the argument to the option -q.

135007 2(CRIT) Platform::main() -- port %d for Command and Control interface is outside valid range 0<= p < 2^16.

Action: The TCP protocol allows 16 bits for the port number. Check and correct the argument to the option -c.

135008 6(INFO) Platform::main() -- now running as daemon, old pid: %d, current pid: %d

Action: Informational message that the daemonization succeeded.

135009 2(CRIT) Platform::main() -- Error parsing configuration file '%s': the configuration file does not match the schema. Please see the preceding messages.

Action: The XML parser has found a syntax error in the AleriML configuration file. The detailed information about the kind and location of the error should be reported in the preceding messages. Check and correct the model.

135010 2(CRIT) Platform::main() -- Configuration file '%s' is invalid, see the preceding messages for details.

Action: The exact error should be described in the preceding messages.

135011 2(CRIT) Platform::main() -- -a <user>:<pass> specified, but %s is NULL.

Action: The user name and password for the intra-cluster authentication mus not be empty.

135012 2(CRIT) Platform::main() -- -K <user>:<privateKeyFile> specified, but %s is NULL.

Action: The user name and key file name for the intra-cluster authentication must not be empty.

135013 2(CRIT) Platform::main() -- spcrypt::getPrivateRSALoginMsg() failed, status=%d, errmsg=%s

Action: Obsolete.

135014 2(CRIT) Platform::main() -- -H <host>:<port> (-H %s) is invalid, check format.

Action: The format of the clustering option argument is invalid.

135015 2(CRIT) Platform::main() -- -H <host>:<port> specified, but instance not HAC enabled, must specify -a, or -k/-K

Action: The option -H is used to enable the secondary platform in the Hot Spare cluster configuration to connect to the primary platform. However the user id and the password or key is also required to log in to the primary platform. Add the login information to the command line.

135016 2(CRIT) Platform::main() -- HA/Cluster authentication options -a, -K and -G are mutually exclusive.

Action: Only one intra-cluster authentication method may be specified. It must match the method used by the other node(s) of the cluster.

135017 2(CRIT) Platform::main() -- %s

Action: The license file can not be read. Check that the PLATFORM_HOME environment variable is properly set and exported, and the the file $PLATFORM_HOME/etc/license.key is installed.

135018 2(CRIT) Platform::main() -- %s

Action: Information about the license expiration time.

135019 2(CRIT) Platform::main() -- %s

Action: The trial license has expired. Purchase a commercial license.

135020 2(CRIT) Platform::main() -- the directory <%s> must exist and have read/execute permissions.

Action: The RSA key directory can not be read. Check whether the path exists and whether the permissions are correct. If the path is relative, check that the Aleri Streaming Platform's current directory is what you think it is.

135021 1(ALERT) Platform::main() -- REVISION <%s>, start timestamp (gmt): %s

Action: Information about the Aleri Streaming Platform build id.

135022 2(CRIT) Platform::main() -- file <%s> does not exist or is not readable.

Action: The Aleri Streaming Platform has experienced an error when trying to read or write a file. Check whether the path exists and if the permissions are correct. If the path is relative, make sure that the Aleri Streaming Platform's current directory is correct.

135023 2(CRIT) Platform::main() -- file <%s> does not exist or is not executable.

Action: The Aleri Streaming Platform has experienced an error when trying to read or write a file. Check whether the path exists and if the permissions are correct. If the path is relative, make sure that the Aleri Streaming Platform's current directory is correct.

135024 2(CRIT) Platform::main() -- environment variable <%s> is not set.

Action: Check that this environment variable is set and exported.

135025 2(CRIT) Platform::main() -- general problem starting process <%s>

Action: An auxiliary process can not be started. There may be multiple reasons for this. Check that the PLATFORM_HOME environment variable is properly set and exported, and the the binary is present and has correct permissions. Also the system limit for the number of processes may be reached, consult your systems administrator.

135026 2(CRIT) Platform::main() -- problem parsing argument list for sslwrap: <%s>

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

135027 2(CRIT) Platform::main() -- Badly formatted authentication database argument.

Action: The testing user database specified with option -A is misformatted. Check and correct your command line.

135028 6(INFO) Platform::main() -- Authentication database from the command line will be used.

Action: The testing user database was specified specified on the command line with option -A. It will be used instead of the normal PAM authentication.

135029 6(INFO) Platform::main() -- Stopping the Command Server, ignore the Socket Error messages.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

135030 6(INFO) Platform::main() -- Waiting for the Command Server completion signal.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

135031 6(INFO) Platform::main() -- Waiting for the Command Server thread to exit.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

135032 6(INFO) Platform::main() -- Destroying the Command Server.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

135033 2(CRIT) Platform::main() -- Authentication mechanism '-V %s' is invalid.

Action: The option -V takes one of the arguments: none, pam, rsa, gssapi. Check your command line.

135034 2(CRIT) Platform::main() -- Warning: invalid gateway port specified with '-g %s'.

Action: The gateway port number must be a positive 16-bit number.

135035 2(CRIT) Platform::main() -- Could not initialize Java VM: %s.

Action: The Java VM could not be initialized, the platform will try to proceed as is. Any functions using Java will be disabled or may crash. Java could be disabled by the option "-DJ".

135036 1(ALERT) Platform::main() -- Starting initialization sequence.

Action: Information about the internal stages of initialization.

135037 6(INFO) Platform::main() -- Stopping the Platform Monitor.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

135038 2(CRIT) Platform::main() -- authentication [%s] mismatch with HA/Cluster authentication [%s]

Action: The same authentication method must be used both between the cluster nodes and for the client connections. Check the command line options related to authentication.

135039 2(CRIT) Platform::main() -- for rsa authentication both "-V rsa" and "-k <path to rsa keys>" must be specified

Action: The RSA authentication method requires the authentication keys.

135040 2(CRIT) Platform::main() -- "-k <path to rsa keys>" may only be used when "-V rsa" is specified

Action: The RSA authentication keys can not be used with any other authentication method.

135041 2(CRIT) Platform::main() -- the file <%s> must exist and have read permissions.

Action: The Aleri Streaming Platform has experienced an error when trying to read or write a file. Check whether the path exists and if the permissions are correct. If the path is relative, make sure that the Aleri Streaming Platform's current directory is correct.

135042 2(CRIT) Platform::main() -- limiting virtual memory to %d megabytes.

Action: Information that the command line option -m was used to set the limit on the memory usage.

135043 2(CRIT) Platform::main() -- port %d for Gateway interface is outside valid range 0<= p < 2^16.

Action: The TCP protocol allows 16 bits for the port number. Check and correct the argument to the option -g.

135044 2(CRIT) Platform::main() -- port %d may not be used for both Command and for Gateway interface.

Action: The XMLRPC and gateway services may not share the same port. Check that the options -c and -g specify different values.

135045 2(CRIT) Platform::main() -- port %d may not be used for both Command and for SQL interface.

Action: The XMLRPC and ODBC services may not share the same port. Check that the options -c and -q specify different values.

135046 2(CRIT) Platform::main() -- port %d may not be used for both Gateway and for SQL interface.

Action: The gateway and ODBC services may not share the same port. Check that the options -g and -q specify different values.

135047 2(CRIT) Platform::main() -- Platform authentication is via <%s>.

Action: Information about the selected authentication mode.

135048 2(CRIT) Platform::main() -- Platform is running without any authentication.

Action: Information about the selected no-authentication mode.

136000 7(DEBUG) %s(%s)::initialize() -- initializing.

Action: Information about the internal stages of initialization.

136001 7(DEBUG) %s(%s)::~%s() -- disposing.

Action: Informational message about the stages of Aleri Streaming Platform shutdown.

136002 6(INFO) %s(%s)::getIndex() -- setting up index '%s'.

Action: Information about the internal stages of initialization.

137000 3(ERR) Platform(): Attempt to assign to parameter %s with value "%s" failed: %s.

Action: Obsolete.

138000 4(WARNING) CleanerMgr(schedule): cleaning scheduled for log: %s

Action: An informational message that the specified LogStore was detected to be running short on free space and was scheduled for cleaning.

138001 6(INFO) CleanerMgr(run): START cleaning dirty logstores (%f secs - since last cleaning).

Action: Information that the store cleaning cycle has started.

138002 6(INFO) CleanerMgr(run): END cleaning dirty logstores (%f secs - to clean).

Action: Information that the store cleaning cycle has completed.

137000 6(INFO) ExternalFeed(execute): [copystream:%s] attempting connection to stream %s(%d), from host %s on port %d.

Action: Information about the intra-cluster connection to be established.

137002 4(WARNING) ExternalFeed(loginXMLRPC): [copystream:%s] fault authenticating, exception: %s

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

137003 2(CRIT) ExternalFeed(loginXMLRPC): [copystream:%s] failed to authenticate, errno: %d

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

137004 2(CRIT) ExternalFeed(getGatewayPortXMLRPC): [copystream:%s] fault obtaining gateway port, exception: %s

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

137005 2(CRIT) ExternalFeed(getGatewayPortXMLRPC): [copystream:%s] failed to obtain gateway port, errno: %d

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

137006 2(CRIT) ExternalFeed(getStreamHandleXMLRPC): [copystream:%s] fault obtaining stream handle(%s), exception: %s

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

137007 2(CRIT) ExternalFeed(getStreamHandleXMLRPC): [copystream:%s] failed obtain stream handle(%s), errno: %d

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

137008 6(INFO) ExternalFeed(getData): [copystream:%s] received an exit from feeding stream %s

Action: Information the the feeding node of the cluster is exiting and sent an exit notification.

137011 2(CRIT) ExternalFeed::(%s): [copystream:%s] failure to authenticate, errno: %d

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

137012 2(CRIT) ExternalFeed::(%s): [copystream:%s] general gateway exception: %s

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

137013 2(CRIT) ExternalFeed(%s): [copystream:%s] failure to subscribe, errno: %d.

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

137014 6(INFO) ExternalFeed(subscribeAndForward): [copystream:%s] closing subscribe socket and exiting.

Action: Information that this cluster node is exiting and closing the connection to the other parts of the cluster.

137015 6(INFO) ExternalFeed(execute): [copystream:%s] from stream: %s, main thread exiting.

Action: Information that this cluster node is exiting and closing the connection to the other parts of the cluster.

138000 2(CRIT) Node(module:%s)::matches() actual host/port <%s/%d>, and node specified host/port <%s/%d> differ.

Action: The port specified for this cluster node in the model (as identified by the options -C and -M) and the port specified on the command line with option -c on the command line differ. They must be the same.

138001 2(CRIT) Node(module:%s)::initialize(): module could not be found.

Action: The module specified with option -M is not present in the model.

138002 2(CRIT) Cluster(%s)::verify(): could not find module <%s>

Action: The module specified with option -M is not present in the model.

138003 2(CRIT) Node(module:%s)::Node(): bad canonical hostname<%s> for node.

Action: The symbolic host name set on the machine does not match the host name set for this module in the model. Make sure that they are the same.

138004 2(CRIT) Node(module:%s)::matches(): canonical hostname of localhost resolved to <%s>

Action: An attempt to resolve the host name of this machine to the canonical name has failed. Check your DNS and the host name setting.

138005 6(INFO) Node(module:%s):: host:(%s:%d) matches our running instance.

Action: Information that the host name has been found to match the cluster configuration.

138006 6(INFO) Node::initialize() host:(%s:%d) has no module, assuming cold spare.

Action: The cluster configuration for this node does not specify a module. The platform will be started in cold spare mode, waiting for the cluster manager command to replace

a failed node.

138007 6(INFO) Node(module:%s)::matches(): found matching module names, useConfigTopology false, skipping host/port verification.

Action: The command line option -T specifies that the cluster topology from the model file should not be used. Instead the cluster manager will provide the actual topology.

138008 2(CRIT) Node(module:%s)::matches(): can not write ephemeral port to file '%s': %s.

Action: The clustered model was configured with the test feature that allows to run all the cluster modules on the same physical machine, with a dynamic allocation of the control port numbers.

139000 6(INFO) %s(%s)::run() -- starting event queue Stream.

Action: Information about the internal stages of initialization.

139001 6(INFO) %s(%s)::initExternalStream() istream<%s> found in module<%s>, running on host<%s:%d(%s)>.

Action: Information about the internal stages of initialization.

139002 2(CRIT) %s(%s)::initExternalStream() istream<%s> could not be found in any module, exiting.

Action: The source for the CopyStream is not present in the model. Check and correct the model.

139003 6(INFO) %s(%s)::initExternalStream() istream<%s> in module<%s> awaiting cluster topology from c&c call.

Action: The platform was started with option -T and there is no information on the location of the other nodes in the cluster. Need to wait for the cluster manager to provide this information.

139004 6(INFO) %s(%s)::migrateModuleLocation() istream<%s> in module<%s> moved to (%s:%d)

Action: The cluster manager has provided the information about the location of a cluster module.

140000 7(DEBUG) SqlParse(): optimizing for key searching.

Action: Debugging information that the SQL parser has found a way to optimize the key search.

141000 2(CRIT) %s(%s)::FlexStream requires at least one input stream

Action: The FlexStream needs one or more input streams. Check and correct the model.

141001 2(CRIT) %s(%s)::FlexStream Number of methods must match number of input streams

Action: The FlexStream must have exactly one Method for each input stream.

141002 2(CRIT) %s(%s)::FlexStream Method refers to unknown input stream %s

Action: The FlexStream must have exactly one Method for each input stream.

141003 2(CRIT) %s(%s)::FlexStream More than one method for input stream %s

Action: The FlexStream must have exactly one Method for each input stream.

141004 2(CRIT) %s(%s)::FlexStream Parse error in method %s

Action: A syntax error in the expression. Check and correct the model.

141005 2(CRIT) %s(%s)::compile() Error in compiling expression: %s.

Action: A syntax error in the expression. Check and correct the model.

141006 2(CRIT) %s(%s)::FlexStream Parse error in locals %s

Action: A syntax error in the expression. Check and correct the model.

141009 2(CRIT) %s(%s)::FlexStream Parse error in Timer block %s

Action: A syntax error in the expression. Check and correct the model.

141010 4(WARNING) FlexStream(%s): Discarding record with null key (tid=%d).

Action: The FlexStream calculation has produced a record with a NULL key field. Such records are illegal. Check and correct the model and/or the input data.

142000 6(INFO) Calendars(): Refreshing calendar data from file %s

Action: Informational message about loading the calendar database.

143000 7(DEBUG) %s: [stream :%s] record diff: opCode: %d, record: %s

Action: The secondary node of the HotSpare cluster connects to the primary node and starts by synchronizing the contents of its SourceStreams to the ones in the primary node. This is an informational message about the found differences.

143001 6(INFO) %s: [stream :%s] %d records of %d differ, posting diffs to sync stream.

Action: The secondary node of the HotSpare cluster connects to the primary node and starts by synchronizing the contents of its SourceStreams to the ones in the primary node. This is an informational message about the found differences.

143002 6(INFO) %s: [stream :%s] stream empty, diffs not required, pure insert synchronization.

Action: The secondary node of the HotSpare cluster connects to the primary node and starts by synchronizing the contents of its SourceStreams to the ones in the primary node. This is an informational message about the found differences.

143003 2(CRIT) %s: [stream: %s] general gateway exception: %s

Action: The connection of the secondary node of the Hot Spare cluster to the primary node has failed. Check that the primary node is running and that its address is specified correctly.

143004 6(INFO) %s: [stream: %s] received EXP_START_SYNC

Action: The secondary node of the HotSpare cluster connects to the primary node and starts by synchronizing the contents of its SourceStreams to the ones in the primary node. This is an informational message about the found differences.

143005 6(INFO) %s: [stream: %s] received EXP_END_SYNC

Action: The secondary node of the HotSpare cluster connects to the primary node and starts by synchronizing the contents of its SourceStreams to the ones in the primary node. This is an informational message about the found differences.

143006 6(INFO) %s: [stream: %s] received an exit from feeding stream %s

Action: The primary node of the Hot Spare cluster is exiting.

143007 2(CRIT) %s: [stream: %s] received an invalid op code (EXP_{START,STOP}_SYNC) while maintaining stream

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

143008 2(CRIT) %s: [stream: %s] received an invalid op code (EXP_STREAM_EXIT) while syncing stream

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

143009 2(CRIT) %s: [stream: %s] received a EXP_TRANS outside of [EXP_STREAM_START, EXP_STREAM_END] while syncing stream.

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

144002 2(CRIT) HotSpare(initConnections): fault authenticating, exception: %s

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

144003 2(CRIT) HotSpare(initConnections): failed to authenticate, errno: %d

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

144004 2(CRIT) HotSpare(initConnections): fault obtaining gateway port, exception: %s

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

144005 2(CRIT) HotSpare(initConnections): failed to obtain gateway port, errno: %d

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

144006 2(CRIT) HotSpare(CheckSourceStreams): failed to find Source Stream %s in running model.

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restric-

tAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

144007 2(CRIT) HotSpare(CheckSourceStreams): failed to obtain Source Streams, errno: %d, errmsg: %s

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

144008 6(INFO) HotSpare(execute): start SYNC with primary -- static streams

Action: The secondary node of the HotSpare cluster connects to the primary node and starts by synchronizing the contents of its SourceStreams to the ones in the primary node. This is an informational message about the synchronization progress.

144009 6(INFO) HotSpare(execute): end SYNC with primary -- static streams

Action: The secondary node of the HotSpare cluster connects to the primary node and starts by synchronizing the contents of its SourceStreams to the ones in the primary node. This is an informational message about the synchronization progress.

144010 6(INFO) HotSpare(execute): start SYNC with primary -- dynamic streams

Action: The secondary node of the HotSpare cluster connects to the primary node and starts by synchronizing the contents of its SourceStreams to the ones in the primary node. This is an informational message about the synchronization progress.

144011 6(INFO) HotSpare(execute): end SYNC with primary -- dynamic streams

Action: The secondary node of the HotSpare cluster connects to the primary node and starts by synchronizing the contents of its SourceStreams to the ones in the primary node. This is an informational message about the synchronization progress.

144012 2(CRIT) HotSpare(execute): failed to set secondary host&port on primary.

Action: An intra-cluster communication failed. Check that the primary node is still running.

144013 2(CRIT) HotSpare(execute): failed to set redundancy mode on primary.

Action: An intra-cluster communication failed. Check that the primary node is still running.

144014 3(ERR) HotSpare(execute): detected failed primary, promoting secondary to primary.

Action: The failure of the primary node in a Hot Spare cluster is detected. The secondary node will take over.

144015 5(NOTICE) HotSpare(execute): detected exiting primary, secondary exiting also.

Action: The shutdown of the primary node in a Hot Spare cluster is detected. The secondary node will follow with the shutdown.

144016 6(INFO) HotSpare(execute): heartbeat: %s

Action: Information about the heartbeat received from the primary node.

144017 2(CRIT) HotSpare()::HotSpare(): could not resolve a canonicalize hostname for <%s>

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

144018 2(CRIT) HotSpare()::verify(): fatal error -- XMLrpc->verify failed, status=%d, errorMsg=%s

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

144019 2(CRIT) HotSpare()::verify(): fatal error -- primary authenticates with: %s, secondary with: %s

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

144020 2(CRIT) HotSpare()::verify(): primary is not HA capable, exiting.

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

145001 2(CRIT) SyncStream::(%s): for stream (%s), failure to authenticate, errno: %d

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

145002 2(CRIT) SyncStream::(%s): for stream (%s), general gateway exception: %s

Action: The connection attempt to another node of the cluster has failed. Check the authentication information provided on the command line. If the model uses the "restrictAccess" attribute, check that it sets the correct permissions for the intra-cluster user id.

146001 2(CRIT) %s::(%s): an input window only supports "records" and "time" for the type attribute.

Action: The attribute "type" of am InputWindow must contain either "records" or "time". Check and correct the model.

146002 6(INFO) InputWindow::InputWindow() built an Input Window of type: %s, istream: %s, value: %d, slack: %d.

Action: Informational message about a defined InputWindow.

146003 6(INFO) InputWindow(%s)::PurgeList(time) -- deleting %d rows as %d transactions.

Action: Informational message about InputWindow operation.

146004 6(INFO) InputWindow(%s)::PurgeList(records) -- deleting %d rows as %d transactions.

Action: Informational message about InputWindow operation.

146005 2(CRIT) InputWindow(%s)::InsertRowTime() -- createTimeQueue empty, but record queue is not (error).

Action: Some self-testing assertion in the Aleri Streaming Platform has failed, which should never happen. Contact Aleri support if this message appears.

147000 4(WARNING) Breakpoint:: ANY INPUT type requires an empty expression, received: '%s'

Action: A breakpoint on any input stream may not have an expression associated with it, since an expression must be tied to a concrete input stream. You may define a separate breakpoint on each input window with its own expression instead.

147001 4(WARNING) Breakpoint:: Parsing error: %s

Action: A syntax error in the expression. Check and correct the argument.

147002 4(WARNING) Breakpoint:: Compilation error: %s, in expression '%s'

Action: A syntax error in the expression. Check and correct the argument.

148000 4(WARNING) ConnectionReader(%s) exiting, %lu records consumed, %lu read, %lu bad, %lu good.

Action: Summary information about a connection.

148001 4(WARNING) ConnectionReader(%s) connection reset error: %s

Action: A connection failed to reset. Probably some parameters are specified wrong. Check and correct the model.

148002 4(WARNING) ConnectionReader(%s) connection conversion error: %s

Action: The data received by connection can not be converted to the internal format. This indicates a probably bug in the connector.

148003 4(WARNING) ConnectionReader(%s) exiting.

Action: Informational message that the connection is exiting.

148004 4(WARNING) ConnectionReader(%s) connection error: %s

Action: A connection has experienced an error. See the connector description.

148000 4(WARNING) ConnectionWriter(%s) exiting, %lu records posted, %lu processed, %lu bad, %lu good.

Action: Summary information about a connection.

148001 4(WARNING) ConnectionWriter(%s) connection reset error: %s

Action: A connection failed to reset. Probably some parameters are specified wrong. Check and correct the model.

148002 4(WARNING) ConnectionWriter(%s) connection conversion error: %s

Action: The data to be sent to the connection can not be converted from the internal

format. This indicates a probably bug in the connector.

148003 4(WARNING) ConnectionWriter(%s) exiting.

Action: Informational message that the connection is exiting.

148004 4(WARNING) ConnectionWriter(%s) connection error: %s

Action: A connection has experienced an error. See the connector description.

149000 7(DEBUG) %s(%s)::get() -- key md5 '%s', record '0x%x'.

Action: Obsolete.

150000 6(INFO) Connection(%s/%s):: state changed to: %s.

Action: Informational message about the connection state change.

150001 0(EMERG) Connection(%s/%s):: %s.

Action: An error message from the connector.

150002 1(ALERT) Connection(%s/%s):: %s.

Action: An error message from the connector.

150003 2(CRIT) Connection(%s/%s):: %s.

Action: An error message from the connector.

150004 3(ERR) Connection(%s/%s):: %s.

Action: An error message from the connector.

150005 4(WARNING) Connection(%s/%s):: %s.

Action: An error message from the connector.

150006 5(NOTICE) Connection(%s/%s):: %s.

Action: An informational message from the connector.

150007 6(INFO) Connection(%s/%s):: %s.

Action: An informational message from the connector.

150008 7(DEBUG) Connection(%s/%s):: %s.

Action: An informational message from the connector.

151000 2(CRIT) %s(%s): Cannot compile ON expression: %s.

Action: A syntax error in the expression. Check and correct the model.

151001 2(CRIT) %s(%s): Unknown input stream '%s'

Action: A syntax error in the expression. Check and correct the model.

151002 2(CRIT) %s(%s): Unknown field '%s' in clause for event '%s'.

Action: A syntax error in the expression. Check and correct the model.

151003 2(CRIT) %s(%s): Field '%s' used at wrong type in FROM block.

Action: A syntax error in the expression. Check and correct the model.

151004 2(CRIT) %s(%s): Error in compiling actions for rules: %s

Action: A syntax error in the expression. Check and correct the model.

151005 2(CRIT) %s(%s): Stream requires at least on input stream.

Action: A syntax error in the expression. Check and correct the model.

151006 2(CRIT) %s(%s): Could not parse pattern expression: %s.

Action: A syntax error in the expression. Check and correct the model.

151007 2(CRIT) %s(%s): Could not create pattern matcher for expression: %s.

Action: A syntax error in the expression. Check and correct the model.

151008 2(CRIT) %s(%s): Event '%s' cannot occur twice in the FROM block.

Action: A syntax error in the expression. Check and correct the model.

151009 2(CRIT) %s(%s): Event '%s' mentioned in ON expression is not in FROM block.

Action: A syntax error in the expression. Check and correct the model.

151010 2(CRIT) %s(%s): Field variable '%s' mentioned more than once in pattern.

Action: Each pattern may bind a pattern variable name to only one field in the incoming record. The pattern bindings cannot be used to compare fields in the same record for equality. Use the "on" expression of the SPLASH block for such comparisons.

151011 2(CRIT) %s(%s): Bad constant for field variable '%s': %s.

Action: A syntax error in the expression. Check and correct the model.

151012 4(WARNING) %s(%s): Discarding record with null key (tid=%d).

Action: The pattern calculation has produced a record with a NULL key field. Such records are illegal. Check and correct the model and/or the input data.

152000 2(CRIT) %s(%s): Bad <Local> declaration: %s.

Action: A syntax error in the expression. Check and correct the model.

153000 2(CRIT) KerberosServer()::execute() new connection # %d.

Action: The Kerberos authentication service has accepted a new client connection.

153001 2(CRIT) KerberosServer()::execute() failed to spawn new connection, exception: %s.

Action: The limit of threads per process is probably exhausted. Consult your system administrator.

153002 2(CRIT) KerberosServer()::execute() client # %d successfully authenticated.

Action: The client has successfully authenticated through Kerberos.

153003 2(CRIT) KerberosServer()::execute() SASL fault while negotiating with client # %d, error message: %s

Action: The client has failed to authenticate through Kerberos due to the Kerberos library error.

153004 2(CRIT) KerberosServer()::execute() client # %d failed to authenticate.

Action: A client failed to authenticate through Kerberos. Check that the client's Kerberos ticket has not expired.