Administrators Guide

Sybase Event Stream Processor 5.0

Contents

CHAPTER 1: Basic Administrative Tasks 1

Provisioning Users 1

Logging 1

Cluster Node Log Configuration File 1

Project Logging 3

Logging Level 3

Data Backup and Restoration 4

Data Backup 4

Performing an Offline Back Up 5

Performing an Online Back Up 5

Viewing Backup Files 6

Data Restoration 6

Restoring Backup Files on Linux and Solaris Systems 6

Restoring Backup Files on Windows Systems 7

Sizing a Log Store 7

Log Stores 9

Memory Usage 12

Monitoring the Project Server 12

CHAPTER 2: Administering Clusters 15

Clustering Architecture 15

File and Directory Infrastructure 17

Clustering Guidelines 17

Multiple Project Support 18

High Availability 18

Project Deployment Options 19

Cluster Persistence and Caching 23

Centralized Security 23

Starting a Cluster **24**
Connecting to a Server from ESP Studio 24
Configuring Clusters **25**
Cluster Administrative Tool 28
Using Log Stores in a Cluster Environment 31
Sample Project Configuration File **33**

CHAPTER 3: Security in Event Stream Processor 37

Safeguarding Your Data **37**
Authentication **37**
Configuring the Server for No Security 38
RSA Authentication 38
Generating a RSA Key with the Java Keytool 38
Configuring the Server for RSA 39
Configuring the Server for Kerberos 39
LDAP Authentication 40
Configuring the Server for LDAP 40
Access Control 40
Secure Sockets Layer (SSL) Connections **46**
Generating the Java Keystore 46
Generating Pem Format Private Keys 48
Viewing the Security Log **48**

CHAPTER 4: Password Encryption on Configuration Files 49

Calling esp_cluster_admin **49**
Encrypting Passwords for Configuration Files **49**
Adapter Encryption and Decryption Scripts **51**
Encrypting Passwords for Java External Adapters **51**
Encrypting the RTView Adapter Password **53**

CHAPTER 5: External Database Access 55

Service Configuration File **55**

Sample Service Configuration File 56
Configuring Connections to External Databases 56

APPENDIX A: Metadata Streams 61

_ESP_Clients 61
_ESP_Clients_Monitor 62
_ESP_Clockupdates 64
_ESP_Columns 64
_ESP_Config 65
_ESP_Connectors 65
_ESP_Keycolumns 67
_ESP_RunUpdates 67
_ESP_Streams 68
_ESP_Streams_Monitor 69
_ESP_Streams_Topology 70
_ESP_Subscriptions 70
_ESP_Subscriptions_Ext 70
_ESP_Tables 71

APPENDIX B: Time Zones 73

Index 75

CHAPTER 1 Basic Administrative Tasks

Basic administrative tasks prepare a minimal Sybase® Event Stream Processor environment for users.

Although some administrative tasks, such as starting a server, are accessible from within Studio, it is designed as a development environment and not a monitoring tool. Therefore, most administration for Event Stream Processor requires use of the command line utilities or direct interaction with files.

Provisioning Users

Provisioning users is a typical administrative task. Event Stream Processor integrates with your existing security infrastructure to authenticate users and grant them access to projects.

Users are authenticated in Event Stream Processor only when clustering is configured. While projects can be run in standalone mode from the command-line, this is not recommended; when in standalone mode, there is no security.

An Event Stream Processor cluster supports the following authentication mechanisms:

- No security
- RSA
- Kerberos
- LDAP

See also

- *Chapter 3, Security in Event Stream Processor* on page 37

Logging

Configure log files at the cluster server level and project level. A cluster node may contain multiple projects, each with their own project log.

Cluster Node Log Configuration File

The configuration file for node logging is `cluster.log.properties`, a **log4j** property file.

Create one cluster node log configuration file for each node in the cluster. Create the file in the directory from which the node is typically started. For example, the node base directory

`%ESP_HOME%\cluster\projects\<cluster-name>` (Windows) or `$ESP_HOME/cluster/projects/<cluster-name>` (Linux/Solaris).

If you configured a cluster during installation, by default, the cluster node log configuration file is placed in `<ESP_HOME>/cluster/nodes/<node-name>`. Once you run a project on that cluster, the base directory changes to `<ESP_HOME>/cluster/projects/<node-name>`. As you configure additional clusters, you may change the base directory in which the cluster nodes are saved. Ensure that the cluster node log configuration file is located in the same folder as that node's configuration file.

A sample `cluster.log.properties` file:

```
com.sybase.esp.cluster.logfile=cluster.log

log4j.rootLogger=info, A

log4j.appender.A=org.apache.log4j.RollingFileAppender
log4j.appender.A.File=${com.sybase.esp.cluster.logfile}
log4j.appender.A.MaxFileSize=1MB
log4j.appender.A.MaxBackupIndex=5
log4j.appender.A.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.A.layout.ConversionPattern=%d{MMM dd yyyy
HH:mm:ss.SSS} %p %t %c - %m%n


log4j.logger.com.sybase.esp=info
log4j.logger.com.sybase.esp.cluster.applications=info

log4j.additivity.com.sybase.esp.cluster.applications=false


.level=INFO
handlers=com.sybase.esp.cluster.impl.Log4JHandler
com.sybase.esp.cluster.impl.Log4JHandler.level=FINEST
```

The cluster node log is written to the file `node_one.log`. In this sample configuration, the log file is configured to back up its contents once the file reaches 1MB in size. The `MaxBackUpIndex` option specifies how many backup files to create.

You can set the `rootLogger` and `logger.com.sybase.esp` options to `error` or `info`. The `info` option produces minimum log information. Under normal circumstances, keep the `rootLogger` option set to the default value `info`, or the log becomes almost unreadable because of its size. You can use `logger.com.sybase.esp` to debug a node without using third-party debugging components. Do not modify the `log4j.logger.com.sybase.esp.cluster.applications` property; the `info` value is required in this instance.

Consult **log4j** documentation for more information on supported properties and configuration instructions.

Project Logging

Configure project logs to capture errors in running projects. You can configure logs for single or multiple projects in a cluster.

In Event Stream Processor, projects are run on local and remote clusters. Project logs are stored in different directories depending on the type of cluster you deploy.

The local cluster is used by Studio as a testing environment. The files generated by projects in the local cluster, including the project log file (`esp_server.log`), are placed in `<ESP_HOME>\SybaseESP\5.0\workspace\<workspace-name>.<project-name>.<instance-number>`.

Remote cluster nodes must have their own node-specific base directories. The node base directory is `<ESP_HOME>/cluster/projects/<node-name>`, but this path can be modified. This is the parent directory for the project working directories, in which you can find the project log file (`esp_server.log`) specific to each project. All relative paths specified in CCL are relative to the project working directory.

Modify logging levels for projects in their project configuration files (`.ccr`), or using the Project Configuration Editor in Studio. For more information, see the *Studio Users Guide*.

See also

- *Configuring Clusters* on page 25

Logging Level

Logging levels range from 0 to 7, and represent a decreasing order of severity. The default logging level for projects is 4.

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 */

Data Backup and Restoration

Perform an offline or online backup of data, view and restore data based on your system.

An offline backup is the preferred method of data backup. You can, however, perform a backup while the project server is running.

Use the **tar -tvf backup.tar** or **pkunzip -v backup.zip** command to view the contents of your backup files.

Restore data from the backup files for Linux, Solaris, and Windows systems.

See also

- *Sizing a Log Store* on page 7

Data Backup

Manage and protect data by performing a backup. Perform a regular backup of the data managed by Sybase Event Stream Processor. You can perform backups while Event Stream Processor is offline (preferred) or online.

Back up this data:

Data	Description
Projects	Back up `.ccl` and `.ccr` files; `.ccx` and `.cclnotation` files can be regenerated.
Log stores	Configure streams to write their content to a log store. Backing up restores data after system failure.

Sybase recommends that you use offline backups. You can, however, back up projects and log stores while the project server is running. This is called an online backup. An offline backup is preferred because it performs the backup on all machines and is quicker than an online backup. You do not need to individually rename files and file extensions when restoring data, as you would with an online backup.

Note: Ensure all files are added to the backup set. Generally, these files have the same name, but different extensions, and are all stored in the same directory. Ensure all projects and associated log stores are backed up.

See also

- *Data Restoration* on page 6
- *Sizing a Log Store* on page 7

Performing an Offline Back Up

Perform a log store backup while the project server is not running.

Prerequisites

Before shutting down Event Stream Processor, verify the locations of the project files and the type of store defined for each stream. Ensure there are no data streams publishing or subscribing to the project.

Task

1. Use the following command to stop the project:

   ```
   $ESP_HOME/bin/esp_cluster_admin --uri=esp://cluster_server:19011
   --username=me --password=sybase --stop project <workspace-name>/
   <project-name> [<instance-index>]
   ```

   For *`port`*, enter the port number used by your Event Stream Processor installation. For *`espuser`* and *`password`*, enter your user credentials.
2. Back up the `.ccl` and `.ccr` files for each project and its associated log stores. On Linux and Solaris systems, use the **tar** system utility. On Windows systems, use the **pkzip** freeware utility or equivalent. Use the path `<base-directory>/<workspace-name>.<project-name>.<instance-number>` to back up individual log stores. Use the `<base-directory>/<workspace-name>.<project-name>.<instance-number>` folder to back up all log stores in a project.

See also

- *Data Restoration* on page 6
- *Sizing a Log Store* on page 7

Performing an Online Back Up

Perform a log store backup while the project server is running. Deployed projects are not included in an online backup.

Note: You do not need to stop the project server during an online backup, but operation suspends while the backup files are being created, which may cause a short disruption. The length of this suspension depends on the amount of data accumulated in the log stores. Perform an online backup only when short disruptions are acceptable.

Use the **esp_client** utility in the command prompt to create a backup copy of log store files. For example:

```
$ESP_HOME/bin/esp_client -p <host>:<port>/<workspace-name>/<project-
name> -c espuser[:password] backup
```

For *`<host>:<port>`*, enter the host name and port number used by your Event Stream Processor installation. For *`espuser`* and *`password`*, enter your user credentials.

This creates a set of backup files in the log store directories, each with the extension `.bak`. Only the current contents of the stores are copied over.

See also

- *Data Restoration* on page 6
- *Sizing a Log Store* on page 7

Viewing Backup Files

Use **tar -tvf backup.tar** or **pkunzip -v backup.zip** to view your backup files.

- For Linux and Solaris systems, use the **tar -tvf backup.tar** command.
- For Windows systems, use the **pkunzip -v backup.zip** command.

See also

- *Data Restoration* on page 6
- *Sizing a Log Store* on page 7

Data Restoration

Extract the log store contents from backup files to restore data.

To restore files created during an online backup, you will need to rename the `.bak` files. If you did an offline backup, you do not need to rename the files you wish to restore.

For Linux and Solaris systems, extract the back up files using the **tar -xvf** command. For Windows systems, use either the **WinZip** or **pkunzip** command.

See also

- *Data Backup* on page 4
- *Sizing a Log Store* on page 7

Restoring Backup Files on Linux and Solaris Systems

Use the **tar -xvf** command to restore files from the backup file on Linux and Solaris systems.

1. Extract the files to restore from the backup file.
   For example, to extract files from `backup.tar`, use:
   ```
   cd \
   tar -xvf backup.tar
   ```
2. If the files you want to restore were created using the online backup process, rename all `.bak` files to `.log` files. For example:
   ```
   for i in fastraid1/algorithmic1/*.bak; do mv $i ${i%%.bak}.log;
   done
   ```

This example assumes you are using a `ksh` or `bash` shell, and using the same location for all files.

3. Restart the project server.

See also

- *Data Backup* on page 4
- *Sizing a Log Store* on page 7

Restoring Backup Files on Windows Systems

Use the **WinZip** or **pkunzip** command to restore files from the backup file on Windows systems.

Prerequisites

Stop Event Stream Processor.

Task

1. Extract the files to restore from the backup file.
   For example, to extract files from `backup.zip` use:
   ```
   c:
   cd \
   pkunzip backup.zip
   ```
2. Restart Event Stream Processor.

See also

- *Data Backup* on page 4
- *Sizing a Log Store* on page 7

Sizing a Log Store

Calculate the size of log store you require. Correctly sizing your log store is important, as stores that are too small or large can lead to performance issues.

1. Estimate the maximum amount of data, in bytes, that you collect in the log store, as both record count and volume. If you are certain about both the number of records arriving in the source streams and the size of the records, simply perform the calculation. If not, you can use the Playback feature to record and play back real data to get a better idea of the amount of data you need to store.

   The log store reports "`liveSize`" in the server log when the project exits (with log level three or higher) and after every compaction (with log level six or higher).

Note: Skip step 2 if the messages in the server log report "`liveSize`," with the indexing overhead already included.

2. To calculate the basic indexing overhead, multiply the record count by 96 bytes. Add the result to the volume.
3. Choose the value of the **reservePct** parameter. The required store size (in bytes), including the reserve, is calculated as:

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

   Round this calculation up to the next megabyte.
4. Ensure the reserve cannot be overrun by the uncheckpointed data.

   Estimate the maximum amount of uncheckpointed data that is produced when the input queues of all the streams (except 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 project. Include the number of output records that are produced by the stream for each of its input records.

   This example shows the 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 are produced by each stream as they consume the contents of its queue:
      - 1024 records may end up in `derived1`'s input queue. Assuming the queue produces one output record for one input record, it produces 1024 records.
      - 2048 records may end up in `derived2`'s input queue (1024 that are already collected on its own queue, and 1024 more from `derived1`). Assuming that `derived2` is a join and generates on average 2 output records for each input record, it produces 4096 records ([1024 + 1024] * 2).
      - 5120 records may end up in `derived3` (1024 from its own queue and 4096 from `derived2`). Assuming a passthrough ratio of 1, `derived3` produces 5120 records.

      When the project's topology is not linear, you must take all branches into account. The passthrough ratio may be different for data coming from the different parent streams. The data from all the input paths must be added up. Each stream has only one input queue, so its depth is fixed, regardless of how many parent streams it is connected to. However, the mix of records in each queue may vary. Assume the entire queue is composed from the records that produce that highest amount of output. Some input streams may contain static data that is loaded once and never changes during normal work. You do not need to count these inputs. In the example, `derived2` is a join stream, and has static data as its second input.

   b) Calculate the space required by multiplying the total number of records by the average record size of that stream.

For example, if the records in `derived1` average 100 bytes; `derived2`, 200 bytes; and `derived3`, 150 bytes, the calculation is:

(1024 * 100) + (4096 * 200) + (5120 * 150) = 1,689,600

Trace the record count through the entire project, starting from the source streams and down to at least all the streams in the log store. Sum only the data sized from the streams located in the log store.

c) Multiply the record count by 96 bytes to calculate the indexing overhead and add the result to the volume in bytes:

(1024 + 4096 + 5120) * 96 = 983,040

1,689,600 + 983,040 = 2,672,640

Verify that this result is no larger than one quarter of the reserve size:

uncheckpointedBytes < storeBytes * (**reservePct** / 4) / 100

If the result is larger than one quarter of the reserve size, increase the reserve percent and repeat the store size calculation. Uncheckpointed data is mainly a concern for smaller stores. 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.

See also

- *Data Backup* on page 4
- *Data Restoration* on page 6

Log Stores

Specify log store size in a project's XML file.

Unlike memory stores, log stores do not extend automatically. Sizing the log stores correctly is important. A store that is too small requires more frequent cleaning cycles, which severely degrades performance. In the worst case, the log store can overflow and cause the processing to stop. A store that is too large also causes performance issues due to the larger memory and disk footprint; however, these issues are not as severe as those caused by log stores that are too small.

reservePct *Parameter*

The reserve is kept as intermediate space that is used during periodic cleaning of the store, and to perform the correct resize of the store.

Note: If the reserve space is too small and the project runs until the store fills with data, a resize attempt may cause the store to become wedged. This means that it cannot be resized, and the data can be extracted from it only by Sybase Technical Support. It is safer to have too much reserve than too little. The default of 20 percent is adequate in most situations. Multigigabyte stores may use a reduced value as low as 10 percent. Small stores (under 30MB), especially those with multiple streams, may require a higher reserve (up to 40 percent). If you find that 40 percent is still not enough, increase the size of the store.

Event Stream Processor automatically estimates the required reserve size and increases the reserve if it is too small. This usually affects only small stores.

Note: Increasing the reserve reduces the amount of space left for data. Monitor server log messages for automatic adjustments when you start a new project. You may need to increase the store size if these messages appear.

As the store runs, more records are written into it until the free space falls below the reserve. At this point, the source streams are temporarily stopped, the streams quiesced, and the checkpoint and cleaning cycle are performed. Streams do not quiesce immediately: they must first process any data collected in their input queues. Any data produced during quiescence is added to the store, meaning that the reserve must be large enough to accommodate this data and still have enough space left to perform the cleaning cycle. If this data overruns the reserve, the store becomes wedged, because it cannot perform the cleaning cycle. The automatic reserve calculation does not account for uncheckpointed data.

Log Store Size Warnings

As the amount of data in the store grows, and the free space falls below 10 percent (excluding the reserve), Event Stream Processor starts reporting "`log store is nearing capacity`" in the server log. If the data is deleted from the store in bursts, (for example, if data is collected during the day, and data older than a week is discarded at the end of the day), these messages may appear intermittently even after the old data has been flushed. As the cleaning cycle rolls over the data that has been deleted, the messages disappear.

Unless your log store is very small, these warnings appear before the store runs out of space. If you see them, stop Event Stream Processor when convenient, and increase the store size. Otherwise, Event Stream Processor aborts when the free space in the project falls below the reserve size.

If a store is sized incorrectly, the entire reserve may be used up, or “wedged”, and cannot be resized or preserve the content. Delete the store files and restart Event Stream Processor with a clean store. If you make a backup of the store files before deleting them Sybase Technical Support may be able to extract content. Change the store size in the project, and it is resized on restart. You cannot decrease the store size. When you restart a project after resizing the store, it will likely produce server log messages about the free space being below the reserve until the cleaning cycle assimilates the newly added free space.

Streams and Log Stores

If a stream (such as a flex stream) uses the context of local or global variables in its logic, it generally uses a memory store. Otherwise, when Event Stream Processor is restarted, the stream's store is preserved, but values of variables are reset. If these variables are used to create unique keys, they are not unique.

In general, Sybase recommends that you either place only the source streams into the log stores, or place a source stream where 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 on restart. With local or global variables, a restart may cause even bigger inconsistencies.

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 must process large amounts of data from the static stream. Keeping streams separate optimizes cleaning cycles. While this contradicts keeping the source stream and all the streams derived from it in the same log store, it is better to keep only the source streams in the log stores and the derived streams in the memory stores.

ckcount *Parameter*

The **ckcount** (checkpointing count) parameter affects the size of uncheckpointed data. This count shows the number of records that 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 96 bytes per record. Setting it to a small value increases the overhead. With the count set to zero, index data is written after each transaction, and for the single-transaction records the overhead becomes:

96 + 32 * ceiling (log$_2$(number_of_records_in_the_stream))

If a stream is small (for example, fewer than 1000 records), the overhead for each record is:

96 + 32 * ceiling (log$_2$(1000)) = 96 + 32 * 10 = 416

In many cases, the record itself is smaller than its overhead of 416 bytes. Since the effect is logarithmic, large streams are not badly affected. A stream with a million records has a logarithm of 20 and incurs an overhead of 736 bytes per record. The increased overhead affects performance by writing extra data and increasing the frequency of store cleaning.

sweepamount *Parameter*

The **sweepamount** parameter determines how much of the log file is “swept through” during each cleaning pass. It must be between 5 percent to 20 percent 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 performs 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 stops if the relocated live data will fill up the reserve (the swept newly cleaned area becomes the new reserve for the next cycle). Unless other factors override, Sybase recommends that you keep the sweep and the reserve sizes close to each other. **reservePct** is specified in percent while **sweepamount** is specified in megabytes.

Log Store Size and File Locations

Ensure the total size of all log store files does not exceed the size of the machine's available RAM. If this occurs, the machine takes longer to process the data, causing 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.

For storing data locally using log stores, Sybase recommends that you use a high-speed storage device (for example, a raid array or SAN, preferably with a large dynamic RAM cache). For a moderately low throughput, place backing files for log stores on single disk drives (whether SAS, SCSI, IDE, or SATA).

See also

- *Data Backup* on page 4
- *Data Restoration* on page 6

Memory Usage

To maximize performance, the project server ensures that only one copy of a record can exist. To further maximize performance, limit the amount of available memory to each shell process.

There are no configuration settings in the project server that directly set up or control RAM usage on the machine. However, the project server does count records in the system, to ensure that only one copy of a record exists in different streams. Memory usage is directly proportional to the number of records stored in the project.

Set the Java heap size for the Java virtual machine (VM) in the project configuration file (`.ccr`).

Monitoring the Project Server

Use the command-line interface to monitor the performance of a running instance of the project server.

The **esp_monitor** tool reads performance data from a running instance of the project server and displays it on standard output. Monitoring data is only available if the time granularity option is set in the project configuration (CCR) file or using Studio.

The time granularity option specifies, in seconds, how often the set of performance records —one per stream and one per gateway connection— is obtained from the running Event Stream Processor. By default, time granularity for all projects is disabled. Users may choose to leave the time granularity project option disabled when monitoring is not required, to increase performance. The `.ccr` file and Studio set this project option. For more information on configuring projects in Studio, see the *Studio Users Guide*.

Note: The **esp_clients_monitor** stream contains basic information about the connected clients but performance-related fields are populated only with the monitoring option.

To monitor a project running in a cluster that has a manager mode on the host "myhost.sybase.com" with an RPC port of 31415, use the following command:

```
esp_monitor -p myhost.sybase.com:31415/workspace-name/project-name
```

This is a simple example only. For detailed information on the **esp_monitor** tool, see the *Utilities Guide*.

See also

- *Project Deployment Options* on page 19
- *Sample Project Configuration File* on page 33
- *Configuring Clusters* on page 25

CHAPTER 2 Administering Clusters

Clustering provides scale-out support in Event Stream Processor, allowing you to add more nodes to a cluster as needed. Clustering lets users run multiple projects simultaneously, provides high availability and failover, and lets you apply centralized security and support for managing cluster connections.

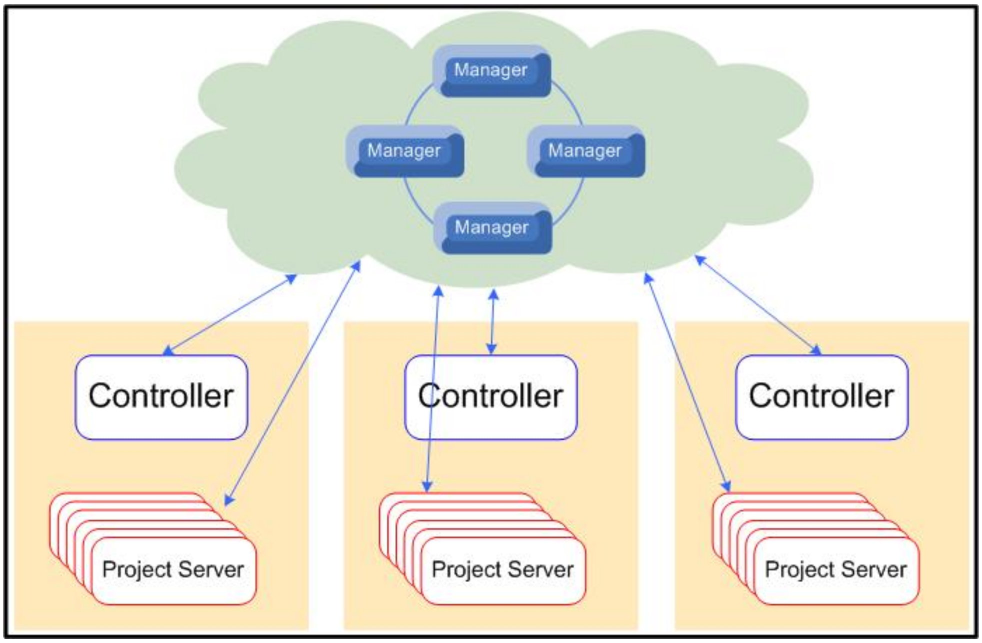
Clustering Architecture

Event Stream Processor clusters are designed for simplicity and minimal need for interaction from administrators once started.

A cluster consists of a group of nodes, which are processes that run on hosts. A cluster can have a single node or multiple nodes. Single-node clusters provide a convenient starting point from which to build and refine multinode clusters.

Clusters consist of manager-only nodes, controller-only nodes, or manager and controller nodes. The cluster launches project servers on demand and manages the project life cycle. In this diagram, containers represent the projects running in a cluster.

Figure 1: Cluster Architecture



A single-node cluster refers to a cluster with a single manager node (the node functions as both a manager and a controller). In development and test environments, a single node cluster may be sufficient. You can deploy several projects to a single-node cluster that monitors project status and, if the project deployed had failover configured, restarts failed projects. However, as you develop and refine your Event Stream Processor environment, the demands on your cluster grow. You can therefore expand your cluster to include additional nodes and, if necessary, additional clusters.

When you have multiple manager nodes in a cluster, it is called a multinode cluster. In a multinode cluster, all manager nodes are considered primary, so there is no single point of failure in the cluster. However, if you configure only one controller for multiple managers, the controller can become a single point of failure.

When a project is deployed to a cluster, it maintains a heartbeat with one of the managers in the node. If the manager node detects three consecutive missed heartbeats from a project, it assumes project failure and issues a **STOP** command and, if the project deployed had failover figured, restarts the project. If your CPU utilization is operating at 100 percent, the project server may not be able to send heartbeats to the cluster manager, stopping the project. In multinode clusters, a different manager may be responsible for monitoring the project than the manager through which it is deployed.

Manager nodes are paired with other managers through a shared cache. If a manager node starts a project and subsequently fails, any other manager with a shared cache can take over management of the projects previously being monitored by the failed manager node.

File and Directory Infrastructure

Manage cluster configuration, project deployment, and persistence using cluster and project configuration files.

Cluster management uses these files and directories:

- `<nodename>.xml` – the cluster configuration file is created automatically for clusters configured at installation, and manually for clusters created after installation. The cluster configuration file contains all the parameters needed for defining nodes as either managers or controllers, setting security, enabling port communication, caching, high-availability projects, and persistence.
- `<projectname>.ccr` – the project configuration file contains parameters for deploying the project in high-availability mode (active-active), defining controller and instance affinities, and failover intervals. You can enable projects for Active-Active deployment only if high availability is enabled in the cluster configuration file for the node on which the project is running.
- `<base-directory>` – (optional) the cluster configuration file allows you to set the base directory where workspace and project instances are stored.
- `<persistence-directory>` – the cluster configuration file allows you to enable persistence and define the directory that persists application and workspace settings across starts and stops of the nodes of the cluster. If one manager in a cluster is enabled for persistence, all managers in the cluster must be enabled for persistence, as well as share the same directory. The directory name varies.
- `<security>` – the cluster configuration file references the Event Stream Processor security directory. This directory stores Event Stream Processor security files, such as the Java keystore file for RSA-protected server instances, LDAP policy files for access control, and the `csi_no_security.xml` file, which provides open security to ESP Server in the absence of other authentication modes. All components in a cluster are subject to the security rules defined in the cluster configuration file.

Clustering Guidelines

Best practices for multiple node configuration.

When configuring multiple nodes in a cluster:

- All nodes in a cluster require the same cache name and password to access the cache.
- All nodes must have unique names specified in their cluster configuration files. Names are case-insensitive.
- Define no more than one manager for every host, in every cluster.
- Set a common base directory for projects. This allows all project log store files to save to a common location.
- Set a common persistence directory across all managers in a cluster. If one manager node in a cluster is enabled for persistence, all managers must be enabled for persistence.

- Reference common security files. All nodes must have the same security configuration. All nodes require a keystore, regardless of authentication mode. The keystore file must be shared among all nodes in the cluster.

  Note: When deploying a new user key, the node to which the deploy command is sent updates the keystore, and the other nodes then reload that file. To test if the deploy key is working properly, log in to the cluster with the new key, but through a different node.

Multiple Project Support

Clustering lets users run multiple projects simultaneously using local and remote clusters.

A local cluster is created automatically by Studio when a user starts a project. A local cluster resides on a single user's machine and is inaccessible to others. Local clusters require separate licenses. Local clusters help users develop and test their own projects, but are not designed to support a production environment.

Administrators can create remote clusters on network servers, and control which users have access to those remote clusters. All clusters started with **esp_server** are considered to be remote, even in the Studio perspective, including those running on the same host as Studio. You can create clusters during the Event Stream Processor installation process, or after the installation is complete.

Users can run multiple projects on a single cluster node or across multiple nodes on separate host machines. Running projects in clusters provides for recovery in the event of project failure.

High Availability

In Event Stream Processor, server clusters promote failure recovery and data redundancy. Event Stream Processor provides an added level of high availability at the project level called Active-Active mode.

A single-node cluster provides project-level failure recovery, meaning it detects when a project stops running and automatically restarts it. However, a single-node cluster does not protect against server failure.

A multiple node cluster can protect against server failure. Create cluster managers on different machines and group them into a single cluster. When deploying a multiple node cluster, you can define project affinities for each node.

You can also deploy projects in Active-Active mode. This means two instances of the same project run in the cluster, preferably on separate machines. One version of the project is designated as the primary instance, and the other is designated as the secondary instance. All connections from outside the cluster (adapters, clients, Studio) are directed to the primary project server. If the primary instance fails, all connections are automatically directed to the secondary instance.

Data between primary and secondary instances is continuously synchronized. The primary instance receives each message first. To maintain redundancy, the secondary instance must also acknowledge receipt of the message before the primary instance begins processing.

See also

- *Project Deployment Options* on page 19

Project Deployment Options

Project deployment options determine how your project is deployed in a cluster and how it functions at runtime. These parameters, including project options, active-active instances, failover intervals, and project deployment type options, are set in the CCR file manually or within Studio.

Project Options

Project options are available on the **Advanced** tab of the Project Configuration editor. Project options are used as runtime parameters for the project, and include a predefined list of available option names that reflect most command line entries. Each project option also has a value field that can be filled by the user. You can implement an option only once, after which it is no longer available in the drop-down list.

This table outlines all available project options that can be set using the Project Configuration view in ESP Studio:

Project Option	Description
on-error-discard-record	If set to true, the record being computed is discarded when a computation failure occurs. If set to false, any uncomputed columns are null-padded and record processing continues. The default value is true. **Note:** If the computation of a key column fails, the record will be discarded regardless of this option.
on-error-log	If set to true, any computation errors that occur will be logged in the error message. The default value is true.
java-classpath	Set the java classpath. Value is a filepath to the classpath file.
java-max-heap	Set the max java heap for the project. Default value is 256 megabytes.
utf8	Enable UTF-8 functionality on the server (by default, this is feature is off). Default value is false, set to true to enable.

Project Option	Description
precision	Set decimal display characteristics for number characters in the project. Default value is 6.
command-port	Set the command port number. This advanced option should not generally be set. If the port is 0, or out of range 1-65535, the program selects an arbitrary port. To define a specific port, set a value between 1 and 65535.
sql-port	Set the SQL port number. This advanced option should not generally be set. If the port is 0, or out of range 1-65535, the program selects an arbitrary port. To define a specific port, set a value between 1 and 65535.
gateway-port	Set the gateway port number. This advanced option should not generally be set. If the port is 0, or out of range 1-65535, the program selects an arbitrary port. To define a specific port, set a value between 1 and 65535.
time-granularity	Define time granularity within the project. This option specifies, in seconds, how often the set of performance records—one per stream and one per gateway connection—is obtained from the running Event Stream Processor. By default, time granularity is set to 5. Set this option to 0 to disable monitoring; this also optimizes performance.
debug-level	Set a logging level for debugging the project, ranging from 0-7. Where each number is represents the following: • 0: LOG_EMERG - system is unusable • 1: LOG_ALERT - action must be taken immediately • 2: LOG_CRIT - critical conditions • 3: LOG_ERR - error conditions • 4: LOG_WARNING - warning conditions • 5: LOG_NORMAL - normal but significant conditions • 6: LOG_INFO - informational • 7: LOG_DEBUG - debug level messages
memory	Set memory usage limits for the project. Default is 0, meaning unlimited.
optimize	Suppresses redundant store updates. Default value is false, set to true to enable.
ignore-config-topology	Enable this to ignore topology between projects. Default is false, set to true to enable.
time-interval	Set the constant interval expression that specifies the maximum age of rows in a window. By default, in seconds, set to 0, meaning no timer.

Active-Active Deployments

Active-active deployments are available only when you define the project as an `ha-project` in the CCR file. An active-active deployment means that two instances of a project run simultaneously in a cluster. The two instances of the project are started by the cluster manager on two different hosts.

One instance of the project is elected as primary instance. If one of the instances is already active, it is the primary instance. If the failed instance restarts, it assumes the secondary position and maintains this position unless the current instance fails or is stopped.

When the secondary project server starts and does not find the primary project server, it reattempts a connection to the primary server for 30 seconds. If it fails to successfully connect to the existing primary server, it takes the responsibility of primary server.

Users add active-active configurations by defining a high-availability type and adding an additional affinity parameter. Users can define an active-active configuration through the CCR Project Deployment editor.

This is an example of a CCR file configured for active-active deployment.

```
<Deployment>
  <Project ha="true">
   <Options>
    <Option name="debug-level">1</Option>
   </Options>
   <Instances>
    <Instance name="primary">
     <Affinities>
      <!-- By default no need to put affinity.  -->
     <Affinity type="controller" charge="positive" strength="strong"
value="node1"/>
      <Affinity type="instance" charge="negative" strength="strong"
value="secondary"/>
     </Affinities>
    </Instance>
    <Instance name="secondary">
     <Affinities>
      <!-- By default no need to put affinity.  -->
      <Affinity type="controller" charge="positive" strength="weak"
value="node2"/>
      <Affinity type="instance" charge="negative" strength="strong"
value="primary"/>
     </Affinities>
     <Failover enable="true">
      <FailureInterval>120<FailureInterval> // numbers in sec
      <FailuresPerInterval>4<FailuresPerInterval> // counter
     </Failover>
    </Instance>
   </Instances>
  </Project>
 </Deployment>
```

Instances

The number of instances available depends on the deployment type chosen by the user, either high availability (HA) or Non-HA. When a project is configured in HA (active-active) mode, two instances are created: primary and secondary. You can set affinity and cold failover options for each instance, including failover intervals and failure per interval options.

Affinities

Affinities limit where a project runs or does not run in a cluster. There are two types of affinities:

- Controller – Used for Active-Active and non Active-Active configurations. You can have more than one affinity for each controller, but there can only be one strong positive controller affinity.
- Instance – Used only for Active-Active configuration, an instance creates two affinities that can apply to each separate project server.

These parameters must be defined for each affinity:

Field	Description
Name	Enter the name of the object of the affinity, that is, the controller name or instance name that the affinity is set for. For instance affinities, the affinity for one instance should refer to the second instance.
Strength	**Strong** or **weak**. Strong requires the project to run on a specific controller, and no others. If weak, the project preferentially starts on the defined controller, but if that controller is unavailable, it may start on another available controller.
Charge	**Positive** or **negative**. If positive, the project runs on the controller. If negative, the project does not run on the controller.

Failover

A project fails when it does not run properly or stops running properly. A failover occurs when a failed project or server switches to another server to continue processing. Failovers may result in a project restart, if defined. Restarts can be limited based on definition of failure intervals and restarts per interval. Failover options, accessed using an instance configuration, include:

Field	Description
Failover	Either **enabled** or **disabled**. When disabled, project failover restarts are not permitted. When enabled, **failure interval** and **failures per interval** fields can be accessed and restarts are permitted.

Field	Description
Failures per interval	Specifies the number of restarts the project can attempt within a given interval. This count can be reset to zero by a manual start of the project or if failures are dropped from the list because they are older than the size of the interval.
Failure interval	(Optional) This specifies the time, in seconds, that make up an interval. If left blank, the interval time is infinite.

See also

- *High Availability* on page 18

Cluster Persistence and Caching

Cluster persistence and caching are configured in the cluster configuration file.

Enable persistence for clusters to maintain application and workspace settings across starts and stops of the nodes in a cluster. When you enable persistence, you can modify the directory in which the cluster configuration settings are persisted. All nodes in a cluster must point to the same persistence directory.

The cluster cache is an in-memory distributed cache used for internal sharing of cluster state and configuration. Manager nodes are members of the cache, while controller nodes are clients of the cache. Cache properties must be configured for all nodes, but you do not need to provide port information for controller-only nodes.

Centralized Security

Security options for Event Stream Processor are configured locally through the cluster.

Authentication modes and Secure Sockets Layer (SSL) connections are configured in the cluster configuration file. Event Stream Processor supports Kerberos, RSA, and LDAP security providers. All projects running in a cluster are subject to the security rules defined for that cluster.

See also

- *Chapter 3, Security in Event Stream Processor* on page 37

Starting a Cluster

Start clusters from the command line or with Studio.

Prerequisites

Set the ESP_HOME environment variable.

Task

When starting a cluster, start manager nodes first, then controller nodes. When the cluster is running, you can deploy projects to them.

1. From the command line for Windows systems, execute:

   ```
   cd %ESP_HOME%\cluster\nodes\node1
   %ESP_HOME%\bin\esp_server.exe --cluster-node node1.xml
   ```

   And for Linux and Solaris systems, execute:

   ```
   cd $ESP_HOME/cluster/nodes/node1
   $ESP_HOME/bin/esp_server --cluster-node node1.xml
   ```

   Note: The directory from which a node is started becomes the working directory for the node. The node looks for the `cluster.log.properties` file in the working directory.

2. Retrieve and start workspaces and projects:

   ```
   esp_cluster_admin --uri=esps://<host>:<port> --
   username=<user> --password=<pass>
   ```

   Provide the cluster URI and your credentials to complete the command and begin working with cluster administration commands.

   Note: The URI protocol `esps` indicates that the cluster is SSL-enabled. URIs for clusters that are not SSL-enabled use the protocol `esp`.

See also

- *Configuring Clusters* on page 25

Connecting to a Server from ESP Studio

Studio provides a graphic interface for connecting to and starting a server, also known as a cluster manager.

By default after installation, a local cluster manager connection is already defined in the Run-Test perspective. You can use this local cluster for testing, or you can define new server connections to remote cluster managers. Both default and user-defined server connections appear in the Server View window in the Run-Test perspective.

If you do not have a server connection already defined, create a new connection by selecting **New Server URL** in the Server View in the Run-Test perspective.

View cluster setup details for ESP Studio in: `$ESP_HOME/studio/esp-studio/clustercfg/localnode.xml`. Do not modify this file, as it impacts your ability to connect to remote clusters from ESP Studio.

1. If you are not already in the Run-Test perspective, click the **Run-Test** tab at the top of the window, or select **Window** > **Open Perspective** > **Run-Test**.
2. If you do not see the Server View window, select **Window** > **Show View** > **Server View** while in the Run-Test perspective.
3. Either

   - Create a new remote cluster connection by selecting **New Server URL** and providing the host name and port for the cluster to which you want to connect.
   - Right-click on the server you want to connect to and select **Connect Server**.

   If the connection is successful, you can see the server streams below the server folder.
4. To connect all of the listed servers, select the **Reconnect All** icon from the top-right corner of the Server View window.

   Unselecting **Filter Metadata Streams** will cause all metadata streams to appear in the Server View.

Configuring Clusters

Configure clusters to enhance performance by dividing processing work among a number of servers.

If you did not configure your cluster during installation, or to create and configure a new cluster, follow these steps.

In a clustering configuration, you must configure all components of a node. Each node in a cluster has four basic sections of configuration, the controller, manager, RPC, and cache:

```
[...]
- <Controller enabled="true">
  </Controller>
  <Manager enabled="true" />
- <Rpc>
  <Port>19011</Port>
  </Rpc>
- <Cache>
        <Host>thehostname</Host>
  <Port>19001</Port>
  <Name>test-name-1</Name>
  <Password>test-password-1</Password>
        <Managers enabled="true">
            <Manager>localhost:19001</Manager>
            <Manager>localhost:19002</Manager>
```

```
         <Manager>localhost:19003</Manager>
        </Managers>
        <Persistence enabled="true"
            <Directory>${ESP_STORAGE}</Directory>
        </Persistence>
  </Cache>
[...]
  </Node>
```

Configuration varies based on whether you enable the node as either a controller or manager, or both.

Note: The sample file used for this task is on a UNIX-based installation of Event Stream Processor.

When configuring a cluster, the value for Host Name uses the default "localhost". To allow cluster clients from other machines to connect, the host name must be changed to that of the machine on which the cluster node is running.

1. Open the the configuration file from `${ESP_HOME}/cluster/nodes/<node-name>/<node-name>.xml` on UNIX installations, or from `%{ESP_HOME}%\cluster\nodes\<node-name>\<node-name>.xml` on Window installations.
2. Provide a unique name for the node within the cluster.

   ```
   <Name>node1</Name>
   ```

   Note: Node names are case-insensitive.
3. (Optional) Configure macro name and type elements.

   A macro is a configuration file shortcut for centralizing a repeated configuration or for acquiring properties from the environment.

   Permitted macro type entries are:
   - envar – value is derived from environment variable defined by macro value.
   - sysproperty – value is derived from the Java system property defined by the macro value.
   - value – the value specified is used.
   - prompt – when the cluster starts, the user is prompted for the value.

   ```
   <Macros>
   <Macro name="ESP_HOME" type="envar">ESP_HOME</Macro>
   </Macros>
   ```
4. (Optional) Configure system properties.

   This step can include the replacement of macro values with literal values through macro expansion.
5. Enable the controller:

   ```
   <Controller enabled="true">
   ```
6. For controller-enabled nodes, configure the pre-existing applications within the controller section. For each application, provide a name, enable the application by setting it to true, and define the application class.

The following example shows a project application, with definitions for the base directory, host name, service configuration file, and security directory that the application uses. These directories may be modified, but must be consistent across all projects within a cluster.

```
- <ApplicationTypes>
- <ApplicationType name="project" enabled="true">
  <Class>com.sybase.esp.cluster.plugins.apptypes.Project</Class>
  <StandardStreamLog enabled="true" />
- <Properties>
  <Property name="esp-home">${ESP_HOME}</Property>
  <Property name="hostname">${ESP_HOSTNAME}</Property>
  <Property name="ld-preload" />
  <Property name="services-file">${ESP_HOME}/bin/service.xml</
Property>
  <Property name="base-directory">${ESP_HOME}/cluster/projects/
test-name-1</Property>
  <Property name="ssl-key-file">${ESP_HOME}/security</Property>
  </Properties>
  </ApplicationType>
```

The **ld-preload** parameter must be modified if you are using an alternative Java Runtime Environment (JRE) than the one provided with Event Stream Processor. If you are using an alternative JRE, point to the jsig library file provided with your runtime environment.

7. Enable the manager:
   `<Manager enabled="true" />`
8. (Optional) Define the host node for the RPC port:

   Use this property to specify which network interface you want a cluster to use, particularly in environments that use multiple network cards.

   ```
   <Host>thehostname</Host>
   ```
9. Provide the RPC port value.

   (Optional) Set the `Port SSL` value to true to enable SSL for the cluster.

   ```
   <Port ssl="true">19011</Port>
   ```
10. For manager-enabled nodes, provide the cache port value.

    ```
    <Port>19011</Port>
    ```
11. (Optional) Define the host node for the cache.

    Use this property to specify which network interface you want a cluster to use, particularly in environments that use multiple network cards.

    ```
    <Host>thehostname</Host>
    ```
12. Define a unique name and password for the cache. All nodes added to this cluster must use the same name and password to interact with the cache.

    ```
      <Name>test-name-1</Name>
      <Password>test-password-1</Password>
    ```
13. (Optional) To enable multicast delivery on manager-enabled nodes, set the enabled value to `true` and enter `Group` and `Port` values.

```
<Multicast enabled="true">
            <Group>224.2.2.7</Group>
            <Port>54323</Port>
        </Multicast>
```

If multicast is not enabled, the manager node must be enabled and defined.

```
<Multicast enabled="false">
 [...]
</Multicast>
<Managers enabled="true">
  <Manager>localhost:19001</Manager>
</Managers>
```

All nodes must have the same multicast status.

14. Enable or disable persistence.

By default, this value is set to `false`. If you are enabling persistence, you can modify the directory where persistence information is stored. All nodes within a single cluster should point to the same persistence directory.

```
<Persistence enabled="true">
    <Directory>${ESP_STORAGE}</Directory>
</Persistence>
```

15. Define security values.

Security values, including `File`, `Policy` and `Keystore` must be consistent with existing security settings.

```
<Security>
        <Csi>
            <File>${ESP_HOME}/security/csi_ldap.xml</File>
            <!--Policy>${ESP_HOME}/security/policy.xml</Policy-->
        </Csi>
        <Keystore>
            <Type>JKS</Type>
            <File>${ESP_HOME}/security/keystore.jks</File>
            <Password>pass123</Password>
            <KeyPassword>pass123</KeyPassword>
            <Algorithm>RSA</Algorithm>
        </Keystore>
    </Security>
```

Note: The `<Policy>` parameter applies only to LDAP authentication.

See also

- *Starting a Cluster* on page 24

Cluster Administrative Tool

The cluster administrative tool is an auxiliary tool for cluster administration. Add and remove projects and workspaces, and query, start, and stop existing projects.

The cluster administrative tool operates in interactive mode or command line mode. In interactive mode, connect the cluster manager once and execute commands until you exit. In command line mode, the utility logs you out after each command; you must connect to a cluster manager every time you specify a command.

Note: The parameters, excluding supported commands, are case-insensitive.

To connect the cluster manager with no security configured:

```
esp_cluster_admin --uri=esp://<host>:<port>
```

To connect the cluster manager with RSA authentication:

```
esp_cluster_admin --uri=esp(s)://<host>:<port> --keyalias=
<keyalias> --storepass=<storepass> --keystore=<keystore>
```

To connect the cluster manager with Kerberos or LDAP authentication:

```
esp_cluster_admin --uri=esp(s)://<host>:<port> --username=<user> --
password=<password>
```

Table 1. Supported Commands

Command	Function
Interactive mode: **get managers** Command line mode: **--get_managers**	Returns the host-name:rpc-port pairs for the managers in the cluster.
Interactive mode: **get controllers** Command line mode: **--get_controllers**	Returns the list of controllers in the cluster.
Interactive mode: **get workspaces** Command line mode: **--get_workspaces**	Returns the names of the workspaces in the cluster.
Interactive mode: **get projects** Command line mode: **--get_projects**	Returns the list of projects, with their state, matching the argument list.
Interactive mode: **get project <workspace-name>/<project-name>** Command line mode: **--get_projectdetail**	Shows a project within an associated workspace.
Interactive mode: **get streams <workspace-name>/<project name>** Command line mode: **--get_streams**	Shows a stream within an associated workspace.
Interactive mode: **get schema <workspace-name>/<project-name> <stream-name>** Command line mode: **--get_schema**	Shows the schema of an associated stream.

Command	Function
Interactive mode: **add workspace <workspace-name>** Command line mode: **--add_workspace**	Adds a workspace.
Interactive mode: **add project <workspace-name>/<project-name> <ccx> [<ccr>]** Command line mode: **--add_project**	Adds a project. The controller name is optional. If not specified, a controller is randomly chosen at the start of the application. A randomly chosen controller has weak affinity once started. If a controller name is specified, the default controller affinity is weak.
Interactive mode: **remove workspace <workspace-name>** Command line mode: **--remove_workspace**	Removes a workspace.
Interactive mode: **remove project <workspace-name>/<project-name>** Command line mode: **--remove_project**	Removes the project. The user must first stop the project.
Interactive mode: **start project <workspace-name>/<project-name> [timeout (sec)] [<instance-index>]** Command line mode: **--start_project**	Starts the project. If the project is added with a strong controller affinity and the controller is not available, start-up fails.
Interactive mode: **stop project <workspace-name>/<project-name> [timeout (sec)] [<instance-index>]** Command line mode: **--stop_project**	Stops the project.
Interactive mode: **stop node <node-name>** Command line mode: **--stop_node**	Stops a node.
Interactive mode: **encrypt <clear-text>** Command line mode: **--encrypt_text**	Encrypts plain text data. Use to encrypt passwords in configuration files.

Command	Function
Interactive mode: **deploykey <new-username> <keystore> <storepass> <key-alias> [<store-type>]** Command line mode: **--deploy_key**	Adds a new user by deploying a new user key to the keystore.
Interactive mode: **reload policy** Command line mode: **--reload_policy**	Reloads the `policy.xml` file in a running cluster. If you have recently updated the existing policy file, the cluster is reverified against the new policy configuration upon reload.
Interactive mode: **connect**	Connect or reconnect a project to a cluster. This command is in interactive mode only.
Interactive mode: **quit** or **exit**	Logs you out of interactive mode. To reaccess the utility, provide your user name and password, unless you chose not to set up authentication.
Interactive mode: **> help** Command line mode: **--help**	Retrieves plain-text description of utility commands and usage information.

These interactive mode commands demonstrate the use of some of the parameters and commands:

```
esp_cluster_admin --uri=esp://cluster_server:19011 --username=me --
password=sybase get managers
esp_cluster_admin --uri=esp://cluster_server:19011 --username=me --
password=sybase get workspaces
esp_cluster_admin --uri=esp://cluster_server:19011 --username=me --
password=sybase get projects
```

This command line demonstrates the use of some of the parameters and commands:

```
esp_cluster_admin --uri=esp://cluster_server:19011 --username=me --
password=sybase --get_managers
esp_cluster_admin --uri=esp://cluster_server:19011 --username=me --
password=sybase --get_workspaces
esp_cluster_admin --uri=esp://cluster_server:19011 --username=me --
password=sybase --get_projects
```

Using Log Stores in a Cluster Environment

Define a relative or absolute path for the log store by specifying a path for the `filename` property within the **CREATE LOG STORE** statement.

Within a clustered environment, a filepath must be specified for the `filename` property in a **CREATE LOG STORE** statement. Create one log store per project. The filename property in the log store definition should use a relative path; the preferred destination for log files is the base directory where project files are stored.

While it is not recommended, you can define an absolute filepath for the log store instead of the recommended relative filepath. However, if a project is restarted on a different machine, the absolute path must also be valid there. Configure projects to always start on the same machine by setting strong positive controller affinity.

1. In the CCL editor, create a log store using the **CREATE LOG STORE** statement:

   ```
   CREATE [DEFAULT] LOG STORE storename
   PROPERTIES
   filename='filepath'
       [sync={ true | false},]
       [sweepamount=size,]
       [reservepct=size,]
       [ckcount=size,]
       [maxfilesize=filesize];
   ```

2. For the `filename` property enter either a relative (preferred) or absolute file path for the location of the log store:

(Preferred) Relative path	A relative path is relative to the base directory. Using a relative path means that your log store automatically points to the base directory. Relative paths do not point to the directory stack; this means that the path does not start with a drive letter or slash (/).
(Not recommended) Absolute path	An absolute path points to any location on your machine, regardless of the current working directory (base directory). For Windows systems, an absolute path begins with the drive letter; on UNIX and Solaris systems, the absolute path begins with a slash (/).

   The relative path location should be a shared disk accessible by all cluster nodes. The log store path is the path relative to that which is specified in the `filename` property within the log store definition. Using a relative path automatically places the log store under: `<base-directory>/<workspace-name>.<project-name>.<instance-number>.<logstore-path>`. You can view base directory definitions in the cluster configuration file (`<node-name>.xml`), under the controller section.

   While you can use an absolute path, a relative path is recommended. To use an absolute path, first ensure that all cluster nodes can read and write to the absolute path you specify. This means that the location must be the same for all cluster nodes. You must also ensure that no two projects use the same path for the log store location. If using a shared disk is not possible, strong affinity can be used to ensure the project is always started on the same cluster node.

3. Enter appropriate values for the remaining properties in the **CREATE LOG STORE** statement.

4. Click **Compile** (F7).

5. Click **Run Project**.

See also

- *Project Logging* on page 3
- *Sizing a Log Store* on page 7

Sample Project Configuration File

Use this example project configuration CCR file to build and modify your XML-based project configuration file.

In addition to using Studio Project Configuration Editor, you can build and modify the project configuration CCR file in a text editor by directly editing the XML.

Note: You can open the CCR file in a text editor by locating the file in your system and opening it in an editor of your choice, or by right-clicking the CCR file in the **File Explorer** pane and selecting **Open With** > **Text Editor**.

The following example is provided with your version of Studio. It is broken down in sections according to the preferences being set, including clusters, bindings, parameters, adapters, and project settings.

```
<Configuration>
    <Runtime>

        <Clusters>
          <!-- we need this only if we have a project/stream binding
-->
            <Cluster name="cluster1" type="local">
                <Username>atest</Username>
                <Password>secret</Password>
            </Cluster>
            <Cluster name="cluster2" type="local">
                <Username>user2</Username>
                <Password>Pass1234</Password>
                <!-- if cluster is remote, we need the Managers
section. if cluster local managers are retrieved internally from the
container -->
            </Cluster>
           <Cluster name="cluster4" type="local"> <!-- this is local
like cluster2 above, but with a different set of credentials -->
                <Username>user4</Username>
                <Password>Pass12345</Password>
                <!-- if cluster is remote, we need the Managers
section. if cluster local managers are retrieved internally from the
container -->
            </Cluster>
        </Clusters>

        <Bindings>
            <Binding name="BaseInput">
```

```
             <Cluster>cluster1</Cluster> <!-- this is always needed
-->
                <Workspace>ws1</Workspace>
                <Project>project1</Project>
              <BindingName>BaseInputBinding</BindingName> <!-- this
is for plat-in adapter name-->
                <RemoteStream>BaseInput</RemoteStream>
            </Binding>
            <Binding name="localStreamName2">
             <Cluster>cluster2</Cluster> <!-- this is always needed
-->
                <Workspace>ws2</Workspace>
                <Project>prj2</Project>
             <BindingName>Stream2Binding</BindingName> <!-- this is
for plat-in adapter name-->
                <RemoteStream>remoteStreamName2</RemoteStream>
            </Binding>
            <Binding name="localStreamName3">
             <Cluster>cluster2</Cluster> <!-- this is always needed
-->
                <Workspace>ws3</Workspace>
                <Project>prj3</Project>
             <BindingName>Stream3Binding</BindingName> <!-- this is
for plat-in adapter name-->
                <RemoteStream>remoteStreamName3</RemoteStream>
            </Binding>
            <Binding name="localStreamName4">
             <Cluster>cluster4</Cluster> <!-- this is always needed
-->
                <Workspace>ws4</Workspace>
                <Project>prj4</Project>
             <BindingName>Stream4Binding</BindingName> <!-- this is
for plat-in adapter name-->
                <RemoteStream>remoteStreamName4</RemoteStream>
            </Binding>
            <Binding name="localStreamName5">
             <Cluster>cluster1</Cluster> <!-- this is always needed
-->
                <Workspace>ws1</Workspace>
                <Project>prj2</Project>
             <BindingName>Stream5Binding</BindingName> <!-- this is
for plat-in adapter name-->
                <RemoteStream>remoteStreamName5</RemoteStream>
            </Binding>
        </Bindings>

        <Parameters>
            <Parameter name="myparam1">foo</Parameter>
            <Parameter name="myparam2">1234</Parameter>
            <Parameter name="myparam3">true</Parameter>
        </Parameters>

        <Adapters>
            <Adapter>
                <Properties name="datalocation1">
                    <Property name="Host">myhost1</Property>
                    <Property name="Port">5555</Property>
                </Properties>
            </Adapter>
            <Adapter>
                <Properties  name="datalocation2">
                    <Property name="Host">myhost2</Property>
                    <Property name="Port">6666</
Property>
                </Properties>
            </Adapter>
        </Adapters>

    </Runtime>

    <Deployment>

        <Project>
            <Options>
                <Option name="ignore-config-topology">true</Option>
                <Option name="jvm-heap">512MB</Option>
            </Options>
        </Project>

        <Cluster>
            <Failover></Failover>
            <Affinities>
                <Affinity type="controller"
charge="positive">myController</Affinity>
            </Affinities>
        </Cluster>

    </Deployment>

</Configuration>
```

See also

- *Project Deployment Options* on page 19
- *Clustering Architecture* on page 15
- *Configuring Clusters* on page 25

CHAPTER 3 Security in Event Stream Processor

Security in Event Stream Processor is managed centrally, by the cluster manager. All projects running in a cluster are subject to the security rules defined for that cluster.

Event Stream Processor integrates with existing Kerberos and LDAP security systems and provides built-in RSA certificate-based security. If you use RSA for security, users requesting server connections are required to provide a valid RSA key alias, keystore which contains a private key, and the password of the keystore. If you use LDAP or Kerberos for your security, users must have an LDAP or Kerberos user name and password.

LDAP supports role-based policy configuration. Further refine your security implementation by defining policies that provide or restrict access to a project or its resources, based on role. For example, you can define an administrator role that allows full access to all projects.

Safeguarding Your Data

Take additional measures to safeguard your data to provide redundancy and prevent unauthorized access.

Event Stream Processor keeps configuration information, server logs, and persistent logs in a file system. Sybase recommends you:

- Secure data and files using OS security. Because the cluster configuration file contains keystore password information, it is important that you secure this file by giving read and write access to trusted individuals or groups only.
- Take appropriate measures to secure these files. Sybase recommends using disk volume encryption and storing security-related configuration information on a separate disk
- For additional security and data redundancy, you can use third-party source control to manage your source files. Use the third-party tools to check your source code in and out, and use the ESP Studio to browse your source folder and make updates and changes in the source files that are checked out.

Authentication

Confirm a user's identity using a set of credentials.

When a user logs in to the server, his or her credentials are verified with security providers. LDAP and Kerberos expect a user name and password combination as credentials. RSA authentication requires a key alias, a keystore containing a private key, and the password of the keystore. Only one type of authentication can be configured in the server at a time. You can also choose to use no security.

The server authenticates the user when he or she user provides the appropriate security credentials to the server. If authentication succeeds, the server considers the user a valid client, and login is completed. The user receives a session ID. In subsequent communication, the client uses the session ID to verify itself.

Access control and permissions are supported only with LDAP.

Configuring the Server for No Security

Configure the server to accept any user name and password pair at login.

In no-security mode, a user is required to log in, but can do so using any user name and password pair.

1. Use any text editor to open the `$ESP_HOME/cluster/nodes/<node-name>/<node-name>.xml file` file.
2. Change the CSI field in the Security section to:`<File>csi_no_security.xml</File>`

RSA Authentication

RSA authentication requires a set of private and public keys with an alias that functions as a user name.

RSA authentication uses public and private keys instead of passwords to authenticate with the ESP Server to create a secure login system. Each key has an alias that functions as a user name for login. When users connect to the cluster manager, they provide a key alias, a keystore that contains a private key, and a password for the keystore. To sign a message, you must have a private key, and you must also provide the corresponding public key to the server.

The server uses the user name or certificate alias to get the public key from its keystore. The public key verifies the signed message that was sent by the client/user. Your public key must be deployed in the server.

Generating a RSA Key with the Java Keytool

Use the Java keytool to create public and private keys for RSA authentication if the client is in Java.

RSA authentication uses public and private keys instead of passwords to authenticate with the ESP Server. The `Java keytool` utility is used to generate RSA keys when the client is in Java.

1. Open a command prompt or terminal.
2. Set the ESP_JAVA_HOME to your Java installation.
3. Add `$ESP_JAVA_HOME/bin` in the path.

4. To create a private/public key with the alias specified by the user, enter:

   `keytool -genkey -keyalg RSA -alias <`*`alias/username`*`> -keystore keystore.jks -storepass <`*`password`*`> -keypass <password>`

   <alias/username> is the user-chosen alias for the private and public keys that will function as a user name for logging in using RSA. *<password>* is the user-chosen password required to access the private key associated with the alias.

5. Use the cluster admin tool to deploy the public key to the server:

   ```
   esp_cluster_admin --uri=esp[s]://host-name:port
   --keystore=<keystore>
   --storepass=<storepass>
   --keypass=<keypass>
   --key-alias=<alias>
   ```

   This makes the public key available to the cluster manager. This key becomes the public key that the cluster manager uses to verify the signature messages sent by the client's private key during the authentication process.

Configuring the Server for RSA

To configure the server for RSA authentication, modify the `server.xml`, `csi.xml`, and `csi_rsa.xml` files.

The security configuration information is maintained in an XML-based configuration file that is accessible by all cluster managers.

Note: To achieve proper security with Sybase Event Stream Processor, use both RSA authentication and SSL. Do not use RSA alone.

1. Open the file:

   `$ESP_HOME/cluster/<node-name>/<node-name>.xml`

   You can edit this using any text editor.

2. Within the Security section of the cluster configuration file, in the CSI field, enter:

   ```
   <File>csi_rsa.xml</File>
   ```

Configuring the Server for Kerberos

To configure the server for Kerberos authentication, modify the `csi.xml` and `csi_kerberos.xml` files.

Kerberos support in Event Stream Processor is limited as it does not support Kerberos ticket-based authentication. To authenticate Kerberos, provide your user name and password. The security configuration information is maintained in an XML-based configuration file that is accessible by all cluster managers.

1. Use a text editor to open the `$ESP_HOME/cluster/<node-name>/<node-name>.xml` file. The file should contain the following lines:

```
<Property name="java.security.krb5.realm">REALM_PLACEHOLDER</
Property>
        <Property name="java.security.krb5.kdc">KDC_PLACEHOLDER</
Property>
```

2. Add the following to the`$ESP_HOME/cluster/<node-name>/<node-name>.xml`file:

   ```
   <Csi>
           <File> csi_kerberos.xml</File>
   </Csi>
   ```

3. Restart the server. All of the cluster managers must be restarted.

LDAP Authentication

LDAP is a user name and password authentication option that also allows access control.

Sign into LDAP using UNIX login credentials.

LDAP authentication is the only form of authentication in the Event Stream Processor that lets you restrict user access to specific functions based on user roles or levels of authorization.

Configuring the Server for LDAP

To configure the server for LDAP authentication, modify the `csi.xml` or `csi_ldap.xml` files.

During installation, you can configure LDAP for both LDAP authentication and access control simultaneously.

The security configuration information is maintained in an XML based configuration file which is accessible by all the cluster managers.

1. Use any text editor to open the `$ESP_HOME/cluster/<node-name>/<node-name>.xml` file.
2. Add the following to the file:

   ```
   <Csi>
           <File> csi_ldap.xml</File>
   </Csi>
   ```

3. If not already configured, edit the `csi_ldap.xml` (or `csi_openldap.xml`) file.
4. Restart the server, including all cluster managers.

Access Control

LDAP supports an added level of security in the form of access control, which allows the server to restrict user access based on user roles.

Access control is implemented in the cluster manager and configured in the `policy.xml` file. An administrator can provide relationships among the resources he or she wants to access, the roles he or she has, and which actions are available. One configuration works for all cluster managers.

Roles, Resources, and Actions

To restrict user access through the access control system, a user must have a defined role. This role must be associated with resources and authorized actions for the resource.

Roles

Roles are equivalent to group names, which are defined in the security provider server. In the access control process, the security provider server determines if the user belongs to a particular group. If so, the group is considered to be his or her role, and limits the available resources and actions the user can access.

There is also the option of the *any role, which implies that everyone is part of the role. If the *any role is used, no call is made to the security provider server to check whether the user is part of the role.

Resources

There are two types of resources: cluster and project server. Cluster resources are divided into the following categories:

- Application – add, remove, start, stop, and get projects
- Workspace – add, remove, and get workspaces
- Security – for adding RSA users and reloading the policy file
- Node – get controller and manager nodes, and stop nodes

Resources such as streams and windows are considered project (which runs in project server) resources. Resources in the policy file are defined in a tree like format using "/" to indicate children. For example, if you have a project called workspace1/project1 which has stream1 and window1 elements, you can define these resources in various ways in the policy file:

- workspace1
- workspace1/project1
- workspace1/project1/stream1
- workspace1/project1/window1

Event Stream Processor supports hierarchical resource entitlement, which means if a user is authorized for an action for resource workspace1, then the user is automatically authorized the same action for all resources under the workspace1.

The *any option can also be used as part of the resources. It refers to all the resources in the cluster. You cannot define the *any resource option in a granular fashion, such as workspace1/ *any.

Actions

There are four action types (access methods) available for all resources:

Action Type	Description
READ	Ability to open, get, and subscribe to a specific resource, but not make any changes.
WRITE	Ability to write, add, remove, and update a specific resource.
START	Ability to start a project.
STOP	Ability to stop a project or node.

Access Control Scenario

When the client makes a login call, the security services authenticate the user. When a user of Role A tries to access Resource B, verification ensures the user is authorized to access the resource and perform the desired action on the resource.

A policy file is configured where Resource B can be accessed by users of Role A with Action READ. If a user with Role A tries to perform a WRITE action in Resource B, the user is not authorized. However, if the user is trying to READ Resource B, this action is authorized.

Configuring Access Control

Create and develop relationships among roles, resources, and actions by editing the XML policy file.

Use the cluster manager to manage access control. The relationships between the roles are resources maintained in a single XML policy file used by all cluster managers in a project. This file is loaded automatically when you start the cluster manager. Use the cluster admin tool to reload the policy file at runtime.

1. Use any text editor to open the XML policy file.
2. Add <Policies/> tags to hold all of the policies you create.

   You can include more than one policy within the <Policies/> tags.
3. To start a new policy, add <Policy/> tags.
4. Specify the Policy type as Project or Cluster.

   ```
   <Policy type="Project">
   ```
5. To create a new role for the policy, add <Role/> tags within <Subjects/> tags.

   You can include more than one role in the <Subjects/> tags, however, all the resources and actions will be associated are roles contained in the <Subjects/> tags. For a role with different resources and actions, create a separate policy using the <Policy> tags.
6. Add a group or role to the new role being created within the <Role/> tags.
7. To associate resources with the role, specify each resource with <Resource/> tags, and enclose these in the <Resources/> tag.
8. To associate actions with the resources, specify each action with <Action/> tags and enclose these in the <Actions/> tags.

This is a sample policy file. The investment role can read, write, start, and stop the two resources.

```
 <Policies>
  <Policy type= "Project">
        <Subjects>
           <Role>investment</Role>
        </Subjects>
        <Resources>
          <Resource>Default/PassThrough/vwapTrades</Resource>
          <Resource>Default/Pass1</Resource>
        </Resources>
        <Actions>
          <Action>read</Action>
          <Action>write</Action>
          <Action>stop</Action>
          <Action>start</Action>
        </Actions>
  </Policy> </Policies>
```

See also

- *Configuring the Server for Access Control* on page 45

Sample Policies for Authorization Roles

Use the `policy.xml` file to control access to cricial functions based on authorization role.

Common authorization roles include adminstration, development, business user, support user, auditor, or customization. The following samples illustrate how you can create policies for each of these roles.

Administration

```
<Policy type="Cluster">
    <Subjects>
        <Role>admin</Role>
    </Subjects>
    <Resources>
        <Resource>*any</Resource>
    </Resources>
    <Actions>
        <Action>stop</Action>
        <Action>start</Action>
    </Actions>
</Policy>
```

Development

```
<Policy type="Project">
    <Subjects>
        <Role>developer</Role>
    </Subjects>
    <Resources>
        <Resource>DevWorkspace</Resource>
    </Resources>
    <Actions>
        <Action>read</Action>
        <Action>write</Action>
        <Action>start</Action>
        <Action>stop</Action>
    </Actions>
</Policy>
```

Business User

```
<Policy type="Project">
    <Subjects>
        <Role>businessuser</Role>
    </Subjects>
    <Resources>
        <Resource>Workspace1/Project1/vwapTrades</Resource>
    </Resources>
    <Actions>
        <Action>read</Action>
    </Actions>
</Policy>
```

Support User

```
<Policy type="Project">
    <Subjects>
        <Role>support</Role>
    </Subjects>
    <Resources>
        <Resource>*any*</Resource>
    </Resources>
    <Actions>
        <Action>read</Action>
    </Actions>
</Policy>
```

Auditor

```
<Policy type="Project">
    <Subjects>
        <Role>audit</Role>
    </Subjects>
    <Resources>
        <Resource>*any*</Resource>
    </Resources>
    <Actions>
        <Action>read</Action>
    </Actions>
</Policy>
```

Customization

```
<Policy type="Project">
    <Subjects>
        <Role>customization</Role>
    </Subjects>
    <Resources>
        <Resource>*any*</Resource>
    </Resources>
    <Actions>
        <Action>start</Action>
        <Action>stop</Action>
    </Actions>
</Policy>
```

Configuring the Server for Access Control

Only LDAP can be used as a security provider for access control.

The relationships among the roles, resources, and actions are managed through the XML policy file. Use the CSI file to configure the server to use access control.

1. Configure the security settings to use LDAP authentication.
2. When editing the CSI file, specify the policy file as follows:

   ```
   <Csi>
           <Policy>policy.xml</Policy>
           <File>csi_openldap.xml</File>
   </Csi>
   ```
3. Edit the `csi_ldap.xml` file as appropriate.
4. Configure the roles, resources, and actions in the policy file.

When the client makes a login call, the security provider authenticates the user. When a user tries to perform an action on a resource, the server determines if the user's role grants access to the action and resource. If so, the user is authorized for the action for the resource. Otherwise, action is denied.

See also

- *Configuring Access Control* on page 42

Recovering Administrative Access

If you lose administrative access rights to Sybase Event Stream Processor, there are manual steps you can take to recover them.

Most cases of lost administrative access are a result of configuration changes to your LDAP server or your `policy.xml` file.

To restore your administrative access rights: In the unlikely event that you need to recover administrative access, manually shut down the Sybase Event Stream Processor server through the host machine's operating system

1. With machine-level administrator rights, log in to the machine running the Event Stream Processor server.
2. Using operating system controls, force the Event Stream Processor server to shut down.
3. Manually change your configuration settings as necessary:

  - Modify your LDAP configuration. For example, if the current LDAP server is unresponsive, modify the URL in the `csi_ldap.xml` file to point to a replica LDAP server.
  - Modify your Kerberos configuration.
  - Modify the `policy.xml` file.

4. Restart the server.
   Your administrative rights to Sybase Event Stream Processor are restored.

Secure Sockets Layer (SSL) Connections

Event Stream Processor supports SSL connections over the network to ensure the privacy of communication between client applications and ESP Server.

SSL is supported for remote procedure calls (XMLRPC) in the cluster. A node in the cluster supports either HTTP or HTTPS, but not both simultaneously. When SSL is enabled for a cluster, all components in the cluster are also enabled for SSL.

By default, SSL is enabled for Event Stream Processor. Users can install Event Stream Processor without SSL. If you have not configured a cluster during installation or want to create a new cluster, you can enable SSL in the cluster configuration file.

See also

- *Configuring Clusters* on page 25

Generating the Java Keystore

Java keystores provide a convenient mechanism for storing and deploying X.509 certificates and private keys.

To create a private key, use the keystore tool located in the `$JAVA_HOME/bin` directory. You are required to use these private keys when calling Event Stream Processor utilities. For instance, **esp_cluster_admin** requires a self-signed private key.

Note: Steps 2 to 9 use sample values. The values you enter may vary.

1. From the command line, run the following script to generate a self-signed key:

   ```
   keytool -genkey -alias username -keyalg RSA -keysize 1024 -
   keystore filename.jks
   ```

   Note: The user name and keystore filename required in the command are variable.

   Press **Return**.
2. Enter a new keystore password.

   ```
   Enter keystore password: testpass
   ```

   Note: The password does not appear as you type for security reasons.

Press **Return**.

3. Re-enter the new keystore password.

```
Re-enter new password: testpass
```

Note: The password does not appear as you type for security reasons.

Press **Return**.

4. Enter your first and last name.

```
What is your first and last name?
  [Unknown]:  john smith
```

Press **Return**.

5. Enter the name of your organizational unit.

```
What is the name of your organizational unit?
  [Unknown]:  business
```

Press **Return**.

6. Enter the name of your organization.

```
What is the name of your organization?
  [Unknown]:  company name
```

Press **Return**.

7. Enter the name of your city or locality.

```
What is the name of your City or Locality?
  [Unknown]:  new york
```

Press **Return**.

8. Enter the name of your state or province.

```
What is the name of your State or Province?
  [Unknown]:  new york
```

Press **Return**.

9. Enter your two-letter country code.

```
What is the two-letter country code for this unit?
  [Unknown]:  us
```

Press **Return**.

10. Enter `yes` or `y` to verify that your information is correct.

```
Is CN=john doe, OU=business, O=company name, L=new york, ST=new
york, C=us correct?
  [no]:  y
```

Press **Return**.

11. Enter your key password for `<ceptest>` and press **Return**. If the key password and keystore password are the same, simply hit **Return** to provide the necessary value.

```
Enter key password for <ceptest>
        <RETURN if same as keystore password>:
```

Note: The password does not appear as you type for security reasons.

Your new keystore file is created.

Generating Pem Format Private Keys

Convert the Java keytool to generate pem format private keys.

Prerequisites

Ensure that you have JDK 1.6 installed on your machine.

Task

Several Event Stream Processor utilities, including **esp_client**, **esp_convert**, **esp_upload**, **esp_subscribe**, **esp_cnc**, and **esp_query**, require the pem format private key.

The keystore tool is located in the `$JAVA_HOME/bin` directory.

1. From the command line, run the following script to export the Java keystore to a PKCS12 format keystore, which is used by OpenSSL:

   ```
   keytool -importkeystore -srckeystore filename.jks  -
   destkeystore exportfilename.p12 -deststoretype PKCS12
   ```

   Note: The filename and exportfilename required in this command are variable.

   Press **Return**.
2. Run the following command to convert the PKCS12 format keystore to a pem format private key:

   ```
   openssl pkcs12 -in filename.p12 -out username.private.pem -
   nodes
   ```

   Note: The user name required in this command is variable.

   Press **Return**.

Viewing the Security Log

The security log file describes all the security changes, such as password changes, that have been made.

View the server log in the cluster log file.

CHAPTER 4 Password Encryption on Configuration Files

The Event Stream Processor **esp_cluster_admin** utility supports password encryption for internal adapter, service, and project configuration files. Event Stream Processor also provides the `encrypt.sh` script for encrypting passwords in external adapter configuration files.

Calling esp_cluster_admin

Log and exit the **esp_cluster_admin** utility.

Prerequisites

Configure your environment variables.

Task

For more information on the utility and its supported commands, see the *Utilities Guide*.

1. Call **esp_cluster_admin**:

   ```
   $ESP_HOME/bin/esp_cluster_admin --uri=esp://<host>:<port> --
   username=<user-name> --password=<password>
   ```

   You must provide a user name and password to log in to the utility. Login credentials vary, depending on your chosen authentication method.
2. Run commands continually as desired.
3. Exit the utility by using **exit** or **quit**.

Encrypting Passwords for Configuration Files

esp_cluster_admin includes an **encrypt** command for applying encryption to configuration file passwords.

Prerequisites

Set the ESP_HOME environment variable.

Task

Modify the adapter `.cnxml` and the database service configuration file only during project environment setup; however, since access to project configuration files is required beyond setup, Studio provides an environment in which to modify project file properties. For more information on configuring project files in Studio, see the *Studio Users Guide*.

1. Use a text editor to open the desired configuration file.
2. Call **esp_cluster_admin** and provide your credentials.
3. To run the **encrypt** command against the password in your configuration file, select and copy the password text from the text editor.

   In the following sample configuration file, the password is "Pass1234".

```
<?xml version="1.0" ?>
- <Services>
- <Service Name="MyDBService" Type="DB">
  <Parameter Name="DriverType">JDBCASE</Parameter>
  <Parameter Name="Host">localhost</Parameter>
  <Parameter Name="Port">5000</Parameter>
  <Parameter Name="User">testID</Parameter>
  <Parameter Name="Password" encrypted="false">Pass1234</
Parameter>
  </Service>
  </Services>
```

4. In the utility, enter the **encrypt** command and paste the configuration file password text beside it.

```
--encrypt Pass1234
```

5. Run the command.

   The action produces a string of encrypted text that contains your hidden password:

```
OJ5f+g5FmzcEdcbonmSREyIHPoAf3O3o5LAK9drQp7J5a5snY4luj/
kdnc61LHNARLA7fOQbp2x20PFMRyti2RTl5qgoUxMjIptDXBm3GIOvXso6AoPBG/
RUaA1dV8giMySEK/GJfnxSSsURfAJm5OHSK8pdt7OBmil0CaSUZdc=
```

6. Copy and paste the encrypted text from the utility to the text editor containing the configuration file. Replace the original **password** under the Password parameter with the **encrypted** text, then create and set an encrypted attribute for the parameter to true.

   This attribute ensures that the server recognizes the password as encrypted text and decrypts it at runtime. If the attribute is set to false, the server does not recognize the password as encrypted text and, therefore, tries to process the password without decrypting it, resulting in errors.

   Note: The following example uses the database service configuration file.

```
<?xml version="1.0" ?>
- <Services>
- <Service Name="MyDBService" Type="DB">
  <Parameter Name="DriverType">JDBCASE</Parameter>
  <Parameter Name="Host">localhost</Parameter>
  <Parameter Name="Port">5000</Parameter>
  <Parameter Name="User">testID</Parameter>
  <Parameter Name="Password" encrypted="true">OJ5f
+g5FmzcEdcbonmSREyIHPoAf3O3o5LAK9drQp7J5a5snY4luj/
kdnc61LHNARLA7fOQbp2x20PFMRyti2RTl5qgoUxMjIptDXBm3GIOvXso6AoPBG/
RUaA1dV8giMySEK/GJfnxSSsURfAJm5OHSK8pdt7OBmil0CaSUZdc=</
Parameter>
```

```
  </Service>
  </Services>
```

7. Save the configuration file.

Adapter Encryption and Decryption Scripts

Event Stream Processor provides a pair of scripts useful for encrypting external adapter configuration values and testing decryption of the encrypted values.

The `encrypt.sh/bat` and `decrypt.sh/bat` are available at `$ESP_HOME/adapters`. These are independent utilities that can encrypt or decrypt using any independent keystore.

Values you need to supply include keystore, alias, and the keystore password.

Encrypting Passwords for Java External Adapters

Use an independent keystore to encrypt passwords in external adapter configuration files, and to tell the Server to decrypt the encrypted value at runtime.

Prerequisites

Set the ESP_HOME environment variable. Event Stream Processor supports Java Runtime Environment 1.6.0.22 or later.

Task

Java external adapter configuration files contain an encryption algorithm that the Server uses to authorize decryption.

1. Use any text editor to open the desired external adapter configuration file.
2. Call the `encrypt.sh` script:

```
$JAVA_HOME/bin/java -cp jar/adapterapi.jar:jar/commons-
codec-1.3.jar com.sybase.esp.adapter.api.CryptUtils encrypt
<password> <alias/user> RSA <keystorepath>/keystore.jks
<keystorepassword>
```

   a) Copy the password string from the external adapter configuration file and paste it in the position of the `<password>` variable in the `encrypt.sh` script.
   b) Replace the `<alias/user>` variable with the store key-alias (user name).
   c) Provide the name of the authentication method the external adapter is using. The default is RSA.
   d) Replace the `<keystorepath>` variable with the filepath to the `keystore.jks` file.

e) Replace the `<keystorepassword>` variable with the keystore password.

f) Run the script.

The action produces a string of encrypted text that contains your hidden password:

```
ilNkDIv7MK99CvRHkVmDunuAvErHEyNdGZ
+VTe63PBMEbyZ2CfZf6iHhCtDXD6fR9jPYIT/
3FcyHmX2VL5xEeDL29KJP4xPS6d9/
TUIozJvJb9YhA8yyHUGv9iGUmtJdcN4vvQ1XJPSGHD84vIKSHQOfz8UlZKl07u
Jl54b47JXi+hIt1X3hZtGAaKuNt9BDo3KIgD4McehJFH2eT0vYmLHjWAL
+JoO4V0/+e9ZlgF4hzjpVkYaO5zik7WyWbvVzLcv4sT4A77CGq4/uo
+ZsJlGdBQ/qlSXDBUKBacHhmYBV1j5xZgxLPu2feEl1OGP/+27126/Lz0M/
JVeShDOw==
```

Note: Use the `decrypt.sh` script to validate encrypted text. To run the decrypt command against the encrypted text, call the `decrypt.sh` script and provide the same credentials you provided for the `encrypt.sh` script.

g) Copy and paste the encrypted text from the script to the text editor containing the configuration file. Replace the original password under the **espPassword** parameter with the encrypted text, then create and set the **encrypted** attribute for the parameter to true.

If set to true, this attribute ensures that the Server recognizes the password as encrypted text and is able to decrypt the password at runtime. If the attribute is set to false, the Server does not recognize the password as encrypted text and, therefore, tries to process the password without decrypting it, resulting in errors.

```
<espPassword
encrypted="true">ilNkDIv7MK99CvRHkVmDunuAvErHEyNdGZ
+VTe63PBMEbyZ2CfZf6iHhCtDXD6fR9jPYIT/
3FcyHmX2VL5xEeDL29KJP4xPS6d9/
TUIozJvJb9YhA8yyHUGv9iGUmtJdcN4vvQ1XJPSGHD84vIKSHQOfz8UlZKl07u
Jl54b47JXi+hIt1X3hZtGAaKuNt9BDo3KIgD4McehJFH2eT0vYmLHjWAL
+JoO4V0/+e9ZlgF4hzjpVkYaO5zik7WyWbvVzLcv4sT4A77CGq4/uo
+ZsJlGdBQ/qlSXDBUKBacHhmYBV1j5xZgxLPu2feEl1OGP/+27126/Lz0M/
JVeShDOw==</espPassword>
```

3. The external adapter configuration file contains a `<espConnection>` section that includes the parameters needed to connect to **esp_server**. Provide values for **espHost** and **espPort**, and in the case of a cluster, supply the cluster URI under **espConnection**.

```
<!-- Event Stream Processor settings -->
 <esp>
   <espConnection>
    <espHost>localhost</espHost>
    <espPort>22000</espPort>
<!--    <espProjectUri>esp://localhost:19011/ws1/p1</
espProjectUri> -->
   </espConnection>
```

4. The `<espSecurity>` section contains parameters required to enable authentication for the external adapter, such as user name and password. Specify an authentication type for **espAuthType**.

Authentication Type	Required Value
Kerberos	user_password
LDAP	user_password
keystore, keystore password	server_rsa
none	Connect without authentication

Example using the Kerberos authentication value:

```
<espAuthType>user_password</espAuthType>
```

5. Provide values for other required fields, based on the chosen authentication type.

   Regardless of authentication type, if the password is encrypted, you must define values for **espRSAKeyStore** and **espRSAKeyStorePassword**.

```

<!--    <espRSAKeyFile>/keyfilepath/espuser.private.der</
espRSAKeyFile>   -->
    <espRSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
    <espRSAKeyStorePassword>Sybase123</espRSAKeyStorePassword>
    <espEncryptionAlgorithm>RSA</espEncryptionAlgorithm>
```

6. Modify the authentication type specified for **espEncryptionAlgorithm** as needed. The default value is `RSA`. Your other option is `DSA`.
7. Save the configuration file.

Encrypting the RTView Adapter Password

If the **isEncrypted** property is set to true, encrypt the RTView adapter password.

1. Create a keystore. See *Generating the Java Keystore* for more info on how to do this.
2. Place the `keystore.jks` file in the RTV project folder.

   For the example project, place the keystore file in the example folder.
3. Modify the `encrypt.bat` script under `%ESP_HOME%\adapters` to generate an encrypted password for a given plain text password.
4. Use the RTView Builder to delete the ESP connection.
5. Create a new ESP connection.
6. Provide the encrypted password in the password field.
7. Save the connection.
8. Select **No** to replace the existing `.ini` file.

   If you choose **Yes**, the file under `%RTV_HOME%\lib` is updated.

Note: Updating the `.ini` file directly may not work as the encrypted password usually contains "=" and "\" characters that have to be properly escaped and encoded. Sybase recommends doing this using the graphic user interface.

CHAPTER 5 External Database Access

The Server accesses external databases by using database services defined in the `service.xml` configuration file.

For the Server to communicate with external databases you must:

- Have a working JDBC or ODBC connection with the appropriate JDBC or ODBC driver for the desired external database installed.
- Modify the `service.xml` file with the appropriate configuration information.

The Server supports using JDBC drivers for connecting to the following external databases:

- Adaptive Server® Enterprise
- IBM DB2
- Oracle
- Kx Systems KDB+
- Microsoft SQL Server

The Event Stream Processor Server supports using ODBC drivers for connecting to the following external databases:

- Adaptive Server Enterprise
- Sybase IQ
- SQL Anywhere®
- IBM DB2
- Oracle
- Microsoft SQL Server
- TimesTen
- MySQL 5.x
- PostgreSQL

Service Configuration File

The service configuration file (`service.xml`) contains database service definitions, which include all the properties and parameters required for a database connection.

Use these properties to create the database connection for a service. Adapters that require database access obtain connections from the database manager by specifying the service that the connection is created for. For example, you can define services for connecting to an Adaptive Server Enterprise database through JDBC and to SQL Server through ODBC. So if you create a project with a database (DB) adapter attached to a source stream that retrieve input data from an Adaptive Service Enterprise database over JDBC; specify the Adaptive Server Enterprise service name as part of the adapter configuration (**service** property). At runtime, the adapter obtains a connection from the database manager based on the properties in the service definition, and executes queries over it.

Each <Service> block represents one service definition entry with two attributes on the <Service> node, Name and Type. The Type attribute must be "DB" to indicate the service is a DB service type. The <Service> node has multiple <Parameter> tags, each representing one property or setting. A <Parameter> tag consists of a Name attribute and the actual parameter value. Event Stream Processor DB drivers parse this information and set up connections accordingly.

Sample Service Configuration File

A sample of the `service.xml` configuration file in `ESP_HOME/bin`. You can use this as a reference for creating your custom service configuration file.

```
<?xml version="1.0"?>
<Services>
        <Service Name="SampleJDBCService" Type="DB">
                <Parameter
Name="DriverLibrary">esp_db_jdbc_sybase_lib</Parameter>
                <Parameter Name="Host">localhost</Parameter>
                <Parameter Name="Port">12345</Parameter>
                <Parameter Name="User">user</Parameter>
                <Parameter Name="Password">password</Parameter>
                <Parameter Name="Database">db</Parameter>
        </Service>
        <Service Name="SampleODBCService" Type="DB">
                <Parameter Name="DriverLibrary">esp_db_odbc_lib</
Parameter>
                <Parameter Name="DSN">dsn</Parameter>
                <Parameter Name="User">user</Parameter>
                <Parameter Name="Password">password</Parameter>
        </Service>
</Services>
```

Configuring Connections to External Databases

Use the `service.xml` file to store service definitions that store connection properties required for a database connection.

Create a separate service definition for every external database you want the Event Stream Processor Server to connect to.

You can use the sample service configuration file in `ESP_HOME/bin` as a basis to create your custom `service.xml` file. To use the file in a running project, modify the **services-file** parameter in the cluster configuration file of the node on which you will run the project. This ensures that the project can find the `service.xml` file. For example, `<Property name="services-file">${ESP_HOME}/bin/service.xml</Property>`.

To set up this type of connection from within Sybase Event Stream Processor:

1. Set the **Service Name** parameter to a unique service name for the database service. This name is:
   - case-sensitive
   - must begin with a letter
   - may contain a character string consisting of either letters, numbers, underscores, dots, and colons.

   This service name is the value you specify to components, such as the Database adapter, that accesses external databases.
2. Set the **Type** attribute of the service parameter to `DB`.
3. (Optional) Add a description of your service entry in the **Description** parameter.
4. Set the **DriverLibrary** parameter to the Event Stream Processor library of the database you want to connect to:

   JDBC drivers:

Database	DriverLibrary Value
Adaptive Server Enterprise	esp_db_jdbc_sybase_lib
Microsoft SQL Server	esp_db_jdbc_mssql_lib
IBM DB2	esp_db_jdbc_db2_lib
Oracle	esp_db_jdbc_oracle_lib
Kx Systems KDB+	esp_db_jdbc_kdb_lib

   ODBC drivers:

Database	DriverLibrary Value
All databases	esp_db_odbc_lib OR esp_db_odbc64_lib

   Note: Set the **DriverLibrary** parameter to esp_db_odbc_lib if you are using a 32-bit ODBC driver manager, or esp_db_odbc64_lib if you are using a 64-bit manager. The libraries are built against different driver managers, with the SQLLEN size set to 4 and 8 bytes respectively, to provide compatibility against driver managers and database drivers built with the aforementioned SQLLEN sizes. They are not duplicate libraries.

5. Set the **User** parameter to the user name that you want to use when communicating with the external database.

This value is unencrypted, so anyone with access to the `services.xml` may read the user name.

6. Set the **Password** parameter to the password for your user name.

   To encrypt this password, add the `encrypted="true"` attribute after the password value and use the **esp_cluster_admin** utility to generate encrypted text.
7. (ODBC drivers only) Set the **DSN** parameter to the data source name to be used by your service.

   You should already have this data source set up with the ODBC driver manager.
8. (Oracle and TimesTen ODBC drivers only) Set the **WriteBigIntAsChar** parameter to true to force the ODBC driver plugin to insert bigint type data to a database as chars. Valid values are true or false.

   For example, the Oracle ODBC driver does not support SQL_C_SBIGINT/ SQL_C_UBIGINT parameters, causing errors when the Database Output adapter tries to write `long` and `interval` Event Stream Processor types to `bigint` type columns. To avoid this issue, set this parameter to `true` (`<Parameter Name="WriteBigIntAsChar">true</Parameter>`) when using Oracle and TimesTen ODBC drivers or tables with bigint type columns.
9. (JDBC drivers only) Set:

Parameter	Description
Host	Database server host name.
Port	Database server port number.
Database	Database name. This parameter is not necessary for Oracle and KDB.

or

Parameter	Description
ConnectString	(Optional) This is a JDBC style connect string that contains the host, port and database information. This overrides the three separate **Host**, **Port**, and **Database** parameters. If you want to enable password encryption between your driver and an external server, add the **EncryptPassword** property to the connect string and set it to true. For example: `<Parameter Name="ConnectString>jdbc:sybase:Tds:localhost:5000/cep?ENCRYPT_PASSWORD=true</Parameter>`

Important: For the Oracle JDBC driver, replace the **Database** parameter with the **Instance** parameter. For example, `<Parameter Name="Instance">orcl</Parameter>`.

See also

- *Chapter 4, Password Encryption on Configuration Files* on page 49

APPENDIX A Metadata Streams

Metadata streams are automatically created by Event Stream Processor. Query and subscribe to these streams to obtain important health and performance information about the currently running project.

Some metadata streams contain static information that never change while the project is running, for example, _esp_streams. Other streams continuously update at various periods or on various events. You can subscribe, query, and view metadata streams in the same way as regular streams.

Note: The schema for metadata streams can change between releases as the set of statistics the streams report expands. New columns are added to the end of the schema for existing metadata streams. Keep this in mind when coding.

Cases where metadata streams differ from general streams:

- Metadata streams have reserved names. No other objects can use these names.
- Metadata streams store their records in a special store called ESPMetadataStore. No other streams can use this store.
- Metadata streams cannot be used in CCL or serve as an input for a stream in a project. For example, the following usage is not possible:

  ```
  INSERT INTO myStream SELECT * FROM _ESP_Connectors WHERE latency >
  1
  ```

_ESP_Clients

Contains information about all the currently active gateway client connections.

Column	Type	Description
Handle	`long`	A unique integer ID of the connection.
user_name	`string`	The user name to log in to the connection, shown once the user is authenticated.
IP	`string`	The address of the client machine.
host	`string`	The symbolic host name of the client machine, if available. If not available, host is the IP address of the client machine.
port	`integer`	The TCP port number from which the connection originates.

Column	Type	Description
login_time	`timestamp`	The time the server accepts (but does not authenticate) the connection, in GMT.
conn_tag	`string`	The user-set symbolic connection tag name. If not set by the user, conn_tag is NULL.

_ESP_Clients_Monitor

Contains information about the performance of all currently active gateway client connections. It contains a copy of data from the _esp_clients stream. Monitoring data is available only if the **time-granularity** option is set in the project configuration (CCR) file.

Column	Type	Description
Handle	`long`	A unique integer ID of the connection.
user_name	`string`	The user name to log in to the connection, shown once the user is authenticated.
IP	`string`	The address of the client machine, as a string.
host	`string`	The symbolic host name of the client machine, if available. If not available, host is the IP address of the client machine.
port	`integer`	The TCP port number from which the connection originates.
login_time	`time-stamp`	The time the server accepts (but does not authenticate) the con-nection, in GMT.
conn_tag	`string`	The user-set symbolic connection tag name. If not set by the user, conn_tag is NULL.
cpu_pct	`float`	The CPU usage, as a percentage, by the client's gateway thread since the last update.
last_update	`date`	The time of the current update.
subscribed	`integer`	The status of a subscription to a stream, 1 if subscribed, other-wise 0.
sub_trans_per_se c	`float`	The client's performance, in transactions per second, received by the client since the last update.
sub_rows_per_se c	`float`	The client's performance, in data rows per second, received by the client since the last update.

Column	Type	Description
sub_inc_trans	`long`	The number of transactions, envelopes, or messages received by the client since the last update.
sub_inc_rows	`long`	The number of data rows received by the client since the last update.
sub_total_trans	`long`	The total number of transactions, envelopes, or messages received by the client.
sub_total_rows	`long`	The total number of data rows received by the client.
sub_dropped_rows	`long`	The total number of data rows dropped in the gateway because they were not read quickly enough by the client. For lossy subscriptions.
sub_accum_size	`integer`	The current number of rows collected in the accumulator to be sent in the next pulse. For pulsed subscriptions.
sub_queue	`integer`	The number of rows queued for transmission to the client.
sub_queue_fill_pct	`float`	The current sub_queue, as a percentage, relative to the queue size limit. If sub_queue_fill_pct reaches 100 percent, any future attempts to post data to this client are blocked, propagating the flow control back to the source of the post.
sub_work_queue	`integer`	The number of rows for transmission to the client that are being transferred from the proper queue to the socket buffer. The rows can be regrouped by envelopes.
pub_trans_per_sec	`float`	The client's performance, in transactions per second, sent by the client since the last update. Envelopes and any service messages count as transactions.
pub_rows_per_sec	`float`	The client's performance, in data rows per second, sent by the client since the last update.
pub_inc_trans	`long`	The number of transactions, envelopes, or messages sent by the client since the last update.
pub_inc_rows	`long`	The number of data rows sent by the client since the last update.
pub_total_trans	`long`	The total number of transactions, envelopes, or messages sent by the client.
pub_total_rows	`long`	The total number of data rows sent by the client.

Column	Type	Description
pub_stream_id	`long`	The numeric ID of the stream to which the client is trying to currently publish data. Typically, pub_stream_id is -1, meaning the client is not trying to currently publish data.

_ESP_Clockupdates

Delivers notifications of changes in the logical clock of the project.

Column	Type	Description
Key	`string`	The type of the update, currently "CLOCK".
Rate	`float`	The rate of the logical clock relative to the real time.
Time	`float`	The current time in seconds since the UNIX epoch.
Real	`integer`	The real time flag, 1 if the logical clock matches the system time and 0 if the times do not match.
stop_depth	`integer`	The number of times the clock resume command must be called to resume the flow of time. When the clock is running, stop_depth is 0.
max_sleep	`integer`	The time, in real milliseconds, that guarantees all sleepers discover changes in the physical clock rate or time.

_ESP_Columns

Contains information about all columns of all streams.

Column	Type	Description
usename	`string`	Hard-coded as "user".
relname	`string`	The name of the stream that contains columns described by this row.
attname	`string`	The name of the column described by this row.

Column	Type	Description
attypid	`integer`	The internal PostgreSQL value representing the type of this column. Valid values: • For integer – 23 • For long – 20 • For money – 701 • For float – 701 • For date – 1114 • For timestamp – 1114 • For string – 1043
attnum	`integer`	The position of this column in the schema, starting from 0.

_ESP_Config

Contains the current CCX of the project.

Column	Type	Description
key	`string`	Hard-coded as "XML".
value	`string`	The text of the current CCX.

_ESP_Connectors

Contains information about all in-process adapters defined in the project.

Column	Type	Description
name	`string`	The name of the adapter, as defined in the project.
stream	`string`	The name of the stream on which the adapter is defined.
type	`string`	The adapter type defined in the **ATTACH ADAPTER** statement.
input	`integer`	Values are 1 for InConnection or 0 for OutConnection.
ingroup	`string`	The StartUp group where this connector belongs.

Column	Type	Description
state	`string`	The state of the adapter, one of: • READY – ready to be started. • INITIAL – performing start-up and initialization. • CONTINUOUS – continuously receiving real-time data. • IDLE – currently not receiving data but attempting to re-connect the to the data source or link. • DONE – no remaining input or output data; the adapter is about to exit. • DEAD – the adapter thread exited. The adapter remains in this state until explicitly requested to restart.
total_rows	`long`	The total number of data records recognized in the input data.
good_rows	`long`	The number of data records successfully processed.
bad_rows	`long`	The number of data records that experienced errors. The fields total_rows, good_rows, and bad_rows are updated once in a few seconds to reduce the overhead.
last_error_time	`date`	The time that the error occurred in YYYY-MM-DD hh:mm:ss format.
last_error_msg	`string`	The complete text of the error message as written to the log.
latency	`interval`	The latency introduced by the adapter. For an input adapter, this is the amount of time it takes the adapter to receive data from its source and publish the data to the stream. For an output adapter, this is the amount of time it takes for the adapter to receive a message from the stream and publish the data to its destination. The update period for latency information is adapter-dependent, and is typically specified in seconds. For adapters that do not report latency information, the column value is NULL.

_ESP_Keycolumns

Contains information about the primary key columns of all the streams. If a stream has a primary key, the columns that make up the key are listed in this stream.

Column	Type	Description
table	`string`	The name of the stream owning the column described by this row.
field	`string`	The name of the column described by this row.
type	`integer`	The internal PostgreSQL value representing the type of this column. The possible values are: • For integer – 23 • For long – 20 • For money – 701 • For float – 701 • For date – 1114 • For timestamp – 1114 • For string – 1043

_ESP_RunUpdates

Delivers notifications of changes during debugging. The Server sends notifications only when the project is in trace mode.

Column	Type	Description
key	`strin g`	The type of the update. See table below.
value	`inte- ger`	A number associated with the update, determined by the Key column.
stream	`strin g`	The name of the stream if the update notifies an event related to an individual stream; otherwise, stream NULL.
info	`strin g`	Additional information associated with the update. Its format depends on the type of the update.

Below is the table of the types of updates that the Server sends as debugging commands. The Value and Stream columns in this table correspond to the Value and Stream rows under Column in the _ESP_RunUpdates stream.

Key	Value	Stream	Description
TRACE	0 or 1	None	Enabled (1) or disabled (0).
RUN	0 or 1	None	Event Stream Processor paused (0) or running (1).
STEP	<count>	None	The project was single-stepped, either manually or automatically. The value contains the number of the steps made. No details are provided about the streams that were stepped.
BREAK	<bp-id>	<stream-name>	The breakpoint with ID <bp-id> was triggered on stream <stream-name>. These updates may come either before or after the corresponding update "RUN 0".
NO BREAK	<bp-id>	<stream-name>	A breakpoint with ID <bp-id> on the stream <stream-name> had its leftToTrigger count decreased, but has not triggered yet.
EXCEPTION	None	<stream-name>	An exception occurred on stream <stream-name>. These updates may come either before or after the corresponding update "RUN 0".
REQUES-TEXIT	None	None	A request to shut down the project received.
EXIT	None	None	All the user streams have exited and the project is about to shut down.

_ESP_Streams

Contains information about all streams, delta streams, and windows.

Column	Type	Description
user_name	`string`	Hardcoded as "user".
stream_name	`string`	The name of the stream described by this row.
handle	`long`	The stream's numeric ID.
type	`string`	The type of the stream: "stream", "deltastream", "window", or "metadata".

Column	Type	Description
visibility	`string`	The visibility of the stream: "input", "output", "local", or "intermediate".
target	`string`	The name of the target stream for streams with "intermediate" visibility value. For streams with all other visibility values, target is the same as the stream_name column.

_ESP_Streams_Monitor

Contains information about the performance of streams. It contains a copy of data from _ESP_Streams stream. Monitoring data is available only if the **time-granularity** option is set in the project configuration (CCR) file.

Column	Type	Description
stream	`string`	The name of the stream.
target	`string`	The name of the target element.
cpu_pct	`float`	The CPU usage, as a percentage, by the stream's thread since the last update.
trans_per_sec	`float`	The stream's performance, in transactions per second, since the last update.
rows_per_sec	`float`	The stream's performance, in rows per second, since the last update.
inc_trans	`long`	The number of transactions processed by the server since the last update.
inc_rows	`long`	The number of rows processed by the server since the last update.
queue	`integer`	The current input queue size.
store_rows	`long`	The current number of records in the stream's store.
last_update	`date`	The time of the current update.
sequence	`long`	The sequence number of the current update.
posting_to_client	`long`	The numeric ID of the client connection to which the stream is trying to currently publish data. Typically, posting_to_client is -1, meaning the stream is not trying to currently publish data.

_ESP_Streams_Topology

Contains pairs of names for streams that are directly connected.

Column	Type	Description
src_stream	`string`	The name of the source stream.
dst_stream	`string`	The name of the destination stream.

_ESP_Subscriptions

Contains information about all currently active subscriptions. A dropped connection is considered unsubscribed from everything to which it was subscribed.

Column	Type	Description
stream_handle	`long`	The handle the server assigns to the subscribed stream.
conn_handle	`long`	The handle the server assigns to the subscribed connection.

_ESP_Subscriptions_Ext

Contains information about all currently active subscriptions.

In some situations, there is a delay in updating this stream as new subscriptions are added or existing streams are dropped.

Column	Type	Description
stream_handle	`long`	The handle of the stream the server subscribes to.
conn_handle	`long`	The handle of the connection the server subscribes to.
stream_name	`string`	The name of the stream.
stream_user	`string`	The user name of the owner of the stream.
subscriber_user	`string`	The login name of the user account that owns the subscription.
ip	`string`	The IP address of the client machine.
host	`string`	The symbolic host name of the client machine, if available. If not available, its value is the IP address of the client machine.

Column	Type	Description
port	`integer`	The TCP port number from which the connection owning this subscription originates.
login_time	`time-stamp`	The time the server accepts the connection owning the subscription.

_ESP_Tables

An internal stream that contains a copy of information contained in the _ESP_Streams stream.

Note: Do not use this stream; use the _ESP_Streams stream instead.

Column	Type	Description
relname	`string`	The name of the stream described by this row.
user name	`string`	Hard-coded as "user".
remarks	`string`	The stream's numeric ID, a decimal number as an ASCII string.

APPENDIX B Time Zones

A time zone is a geographic area that has adopted the same standard time, usually referred to as the local time.

Most adjacent time zones are one hour apart. By convention, all time zones compute their local time as an offset from GMT/UTC. GMT (Greenwich Mean Time) is an historical term, originally referring to mean solar time at the Royal Greenwich Observatory in Britain. GMT has been replaced by UTC (Coordinated Universal Time), which is based on atomic clocks. For all Sybase Event Stream Processor purposes, GMT and UTC are equivalent. Due to political and geographical practicalities, time zone characteristics may change over time. For example, the start date and end date of daylight saving time may change, or new time zones may be introduced in newly created countries.

Internally, Event Stream Processor always stores date and time type information as a number of seconds, milliseconds, or microseconds since midnight January 1, 1970 UTC, depending on the datatype. If a time zone designator is not used, UTC time is applied.

Daylight Saving Time

Daylight saving time is considered if the time zone uses daylight saving time and if the specified timestamp is in the time period covered by daylight savings time. The starting and ending dates for daylight saving time are stored in a C++ library.

If the user specifies a particular time zone, and if that time zone uses daylight saving time, Event Stream Processor takes these dates into account to adjust the date and time datatype. For example, since Pacific Standard Time (PST) is in daylight saving time setting, the engine adjusts the timestamp accordingly:

```
to_timestamp('2002-06-18 13:52:00.123456 PST','YYYY-MM-DD
HH24:MI:SS.ff TZD')
```

Transitioning from Standard Time to Daylight Savings Time and Vice-Versa

During the transition to and from daylight saving time, certain times do not exist. For example, in the US, during the transition from standard time to daylight savings time, the clock changes from 01:59 to 03:00; therefore 02:00 does not exist. Conversely, during the transition from daylight saving time to standard time, 01:00 to 01:59 appears twice during one night because the time changes from 2:00 to 1:00 when daylight saving time ends.

However, since there may be incoming data input during these undefined times, the engine must deal with them in some manner. During the transition to daylight savings time, Event Stream Processor interprets 02:59 PST as 01:59 PST. When transitioning back to standard time, Event Stream Processor interprets 02:00 PDT as 01:00 PST.

Index

A

access control 37, 40–42, 45
access control management 40
  access methods 40
  actions 41, 42
  available actions 40
  configuration 45
  resources 41, 42
  role hierarchy 40
  roles 41, 42
active-active 18
authentication
  Kerberos authentication 37, 39
  LDAP authentication 37, 40
  no security 37, 38
  RSA authentication 37–39
authorization 40
  actions 41, 42
  configuration 45
  policy file 42
  resources 41, 42
  roles 41, 42
  sample roles 43

B

backing up data 4
  data backup 4
  offline backup 5
  online backup 5
  viewing backup files 6

C

cluster manager
  Kerberos authentication 37
  LDAP authentication 37, 40
  no security 37
  RSA authentication 37, 38
  security 37
cluster node logging
  log4j configuration file 1
clustering
  administrative tool 28
  architecture 15
  best practices 17
  cache 23
  cluster configuration file 17
  configuration 25
  connection to a server from ESP Studio 24
  esp_cluster_admin utility 28
  file and directory infrastructure 17
  local and remote clusters 18
  log stores 31
  persistence 23
  project configuration file 17
  starting a cluster 24

D

data
  backing up 4
  offline backup 5
  online backup 5
  restoration 4, 6
  restoring backup files on Windows systems 7
  restoring files on Linux and Solaris systems 6
  viewing backup files 6
daylight saving time (DST) 73
debugging
  logging levels 3
decryption
  external adapter decryption script 51
drivers
  JDBC 55
  ODBC 55
DST 73

E

encryption
  encrypting the RTView adapter password 53
  esp_cluster_admin utility 49
  external adapter encryption script 51
  internal adapter configuration files 49
  Java external adapter configuration files 51
  project configuration files 49
  service configuration file 49

esp_cluster_admin utility 28
    starting the utility 49
esp_monitor 12
external database access
    configuring connections 56
    JDBC drivers 55
    ODBC drivers 55

H

high availability
    active-active 18
    multiple node clustering 18

J

Java keytool 38
    generating a self-signed private key 46
    pem format private key 48
JDBC drivers 55

K

Kerberos authentication 37
    configuration 39

L

LDAP authentication 37, 40
    configuration 40
lockout
    recovery 45
log stores
    cluster environment 31
    managing 9
    sizing 7, 9
logging
    project log 1
    server log 1

M

memory
    usage 12
metadata streams 61
    _ESP_Clients 61
    _esp_clients_monitor 62
    _ESP_Clockupdates 64
    _ESP_Columns 64
    _ESP_Config 65
    _ESP_Connectors 65
    _ESP_Keycolumns 67
    _ESP_RunUpdates 67
    _ESP_Streams 68
    _esp_streams_monitor 69
    _ESP_Streams_Topology 70
    _ESP_Subscriptions 70
    _ESP_Subscriptions_Ext 70
    _ESP_Tables 71
monitoring projects 12

N

no security
    configuration 38

O

ODBC drivers 55

P

policy file 42
privileges
    admin, recovering 45
project configuration
    active-active deployment 19
    affinities 19
    failover 19
    instances 19
    project deployment options 19
    sample file 33
project logging
    debug logging levels 3
    project working directory 3
projects
    multiple project support 18
provisioning users 1

R

restoring data 6
    Linux and Solaris systems 6
    restoring 4
    Windows systems 7
roles
    authorization 43

sample 43
RSA authentication 37, 38
configuration 39
key generation 38
RSA key 38
RSA key generation
Java keytool 38
RTView adapter
encrypting the password 53

S

sample project configuration file 33
sample service configuration file 56
security 37
access control management 40–42, 45
authentication 38
authorization 40–42, 45
configuration 38
data 37
Kerberos authentication 37, 39
LDAP authentication 37, 40
no security 37, 38
personal data 37
RSA authentication 37–39
safeguarding your data 37
security changes 48
security log 48
service configuration file 55
sample 56
sizing
log stores 7, 9
SSL 37
configuration 46
stores
log stores 31

T

time zones 73

U

unlocking the server 45

V

viewing backup files 6