SYBASE®
An SAP® Company

Administrators Guide

SAP Sybase Event Stream Processor 5.1 SP01

DOCUMENT ID: DC01611-01-0511-02
LAST REVISED: March 2013

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

Contents

CHAPTER 1: Clusters 1

Clustering Architecture 1
File and Directory Infrastructure 3

CHAPTER 2: Get Started with a Cluster 9

Planning a Cluster 9
High Availability 10
Cluster Persistence, Caching, and Multicast 11
Centralized Security 12
Adding a Node to a Cluster 12
Configuring a Cluster 13
Configure Security 18
Authentication 19
Configuring Kerberos Authentication 20
Configuring RSA Authentication 22
Username-Password Authentication 25
Configuring Cascading Authentication Methods 29
Access Control 30
Roles, Resources, and Actions 31
Configuring Access Control 34
Sample Policies for Authorization Roles 37
Enabling or Disabling Access Control 38
Recovering Administrative Access 39
Secure Sockets Layer (SSL) Connections 40
Generating the Java Keystore 40
Generating Pem Format Private Keys 42
Password Encryption on Configuration Files 43
Calling esp_cluster_admin 43

Encrypting Passwords for Configuration Files 44
Adapter Encryption and Decryption Scripts 45
Encrypting Passwords for Java External Adapters 45
Encrypting the RTView Adapter Password 47
Deploying a Project to a Cluster 48
Project Deployment Options 48
Active-Active Deployments 51
Project Instances 52
Affinities 53
Failover 54
Sample Project Configuration File 54
External Database Access 57
Configure Connections to External Databases 58
Configuring a JDBC Connection to an External Database 59
Configuring an ODBC Connection to an External Database 61
Configuring an Open Client Connection to a Database 62
Service Configuration File 63
Sample Service Configuration File 63
Linking to Your ODBC Driver Manager Library 65

CHAPTER 3: Administer a Cluster 67
Starting a Cluster 67
Stopping a Cluster 68
Connecting to a Server from ESP Studio 68
Logging 69
Cluster Node Log Configuration File 69
Project Logging 71
Logging Level 71
Cluster Administrative Tool 72
Safeguarding Your Data 76

Sharing Projects in a Git or CVS Repository 77
Adding a Project to a Git Repository 77
Adding a Project to a CVS Repository 78
Data Backup and Restoration 78
Data Backup 79
Data Restoration 81
Creating a Log Store 82
Log Stores 83
Sizing a Log Store 86
Memory Usage 88

CHAPTER 4: Monitor a Cluster 89
Monitoring a Project 89
Monitoring with Sybase Control Center 89
Monitoring with Metadata Streams 90
_ESP_Adapter_Statistics 91
_ESP_Clients 91
_ESP_Clients_Monitor 91
_ESP_Clockupdates 94
_ESP_Columns 95
_ESP_Config 95
_ESP_Connectors 95
_ESP_Keycolumns 97
_ESP_Project_Monitor 97
_ESP_RunUpdates 98
_ESP_Streams 100
_ESP_Streams_Monitor 100
_ESP_Streams_Topology 101
_ESP_Subscriptions 102
_ESP_Subscriptions_Ext 102
_ESP_Tables 102

Index 105

CHAPTER 1 Clusters

A Sybase Event Stream Processor cluster consists of one or more nodes—typically there is one node per host. Nodes run the applications, or projects, you create in ESP Studio. Clustering lets you run multiple projects simultaneously. ESP clusters promote failure recovery and data redundancy.

You can run multiple projects on a single cluster node or across multiple nodes on separate host machines. Running projects in clusters has several benefits:

- Eliminates the single point of failure risk that results from running your projects on one host.
- Provides for recovery if a project or node does fail.
- Allows the processing load to be shared across multiple hosts.

A local cluster (a cluster for development purposes in ESP Studio) is created automatically by Studio when you start a project. A local cluster resides on a single user's machine and is inaccessible to others. (All clusters, including the local cluster, require licenses.) Local clusters help you develop and test your 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 clusters, 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.

Every node is part of a cluster.

Clustering Architecture

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

A cluster consists of one or more nodes. Single-node clusters provide a convenient starting point from which to build and refine multinode clusters.

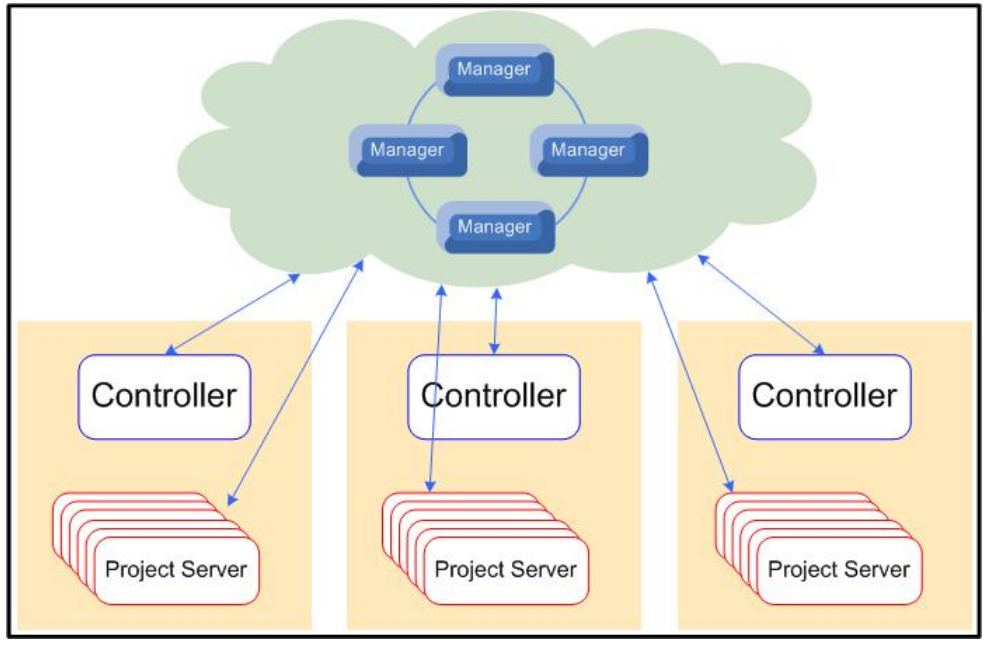
Nodes are processes that run on hosts. There are two functional types of nodes: managers and controllers. A manager is an external access point to a cluster. Managers maintain and monitor cluster and application state. A controller is responsible for starting and monitoring the processes that run projects (project servers).

Clusters can include manager-only nodes, controller-only nodes, or manager and controller nodes. The smallest clusters consist of a single node that serves as both manager and controller.

Note: In a multinode cluster—where there is more than one node with a manager role—any nodes residing on Windows machines must be either managers or controllers, but not both. (Windows also supports single-node clusters.)

A cluster launches project servers on demand and manages the project life cycle.This diagramshows projects running in a cluster.

Figure 1: Cluster Architecture



A single-node cluster refers to a cluster with a single manager node (which 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 cluster. If the manager node detects missed heartbeats from a project for too long, it assumes project failure and issues a **STOP** command. If the project had failover configured, the

manager restarts the project. For example, if your CPU utilization is operating at 100 percent, the project server may not be able to send heartbeats to the cluster manager, which stops the project. In multinode clusters, the manager responsible for monitoring a project might not be the manager through whichthe project is deployed.

All the manager nodes in a cluster store project information in a shared cache. If a manager node starts a project and subsequently fails, the shared cache enables any other manager in the cluster to take over management of the failed manager's projects.

File and Directory Infrastructure

Manage cluster configuration, project deployment, security and options like failover, affinities, and high availability using configuration files and directories.

These are the files and directories you use most often in managing Sybase Event Stream Processor.

Note: In a multinode cluster whose nodes are installed on different hosts, some files and directories must be stored on a shared drive accessible to all nodes. Shared drive requirements are listed in the table under **Needs shared drive?**

Table 1. Event Stream Processor Files and Directories

Name	Description
`<base-directory>`	Base directory to which managers in this cluster deploy projects. Holds workspaces, project instances, project working directories, and log stores. By default, the base directory in a local (Studio) cluster is the same as the workspace. See also *workspace* on page 8 and *project working directory* on page 7. • **Needs shared drive? –** Yes • **Default location –** Remote cluster: `ESP_HOME/cluster/projects/<cluster-name>` Local (Studio) cluster: `<user's-home-dir>/SybaseESP/5.1/workspace` You set the location of the `workspace` directory during installation. In Windows, `<user's-home-dir>` defaults to the `My Documents` folder. If no home directory is configured in UNIX, the base directory location defaults to `/SybaseESP/5.1/workspace`.

Name	Description
`cluster.log`	Cluster node log. Captures node-level and cluster-level errors and event messages; each node's log is unique to that node. Resides in the same directory as `<node-name>.xml` and `cluster.log.properties`. • **Needs shared drive? –** Must not be shared • **Default location –** `ESP_HOME/cluster/nodes/<node-name>/cluster.log`
`cluster.log.properties`	Cluster node log configuration file. `cluster.log.properties` is a log4j properties file; it resides in the directory where the the node is started and where `<node-name>.xml` is located. • **Needs shared drive? –** No • **Default location –** `ESP_HOME/cluster/nodes/<node-name>/cluster.log.properties`

`csi_kerberos.xml`
`csi_ldap.xml`
`csi_native_nt.xml`
`csi_native_unix.xml`
`csi_role_mapping.xml`
`csi_rsa.xml`
	Security configuration files.
	• **Needs shared drive? –** Optional; typically shared in a multinode cluster where the security directory is shared
	• **Default location –** `ESP_HOME/security`

`ESP_HOME`	Represents the Event Stream Processor installation directory. **ESP_HOME** is also an environment variable and a configuration file macro. • **Needs shared drive? –** Must not be shared • **Default location –** Windows: `C:\Sybase\ESP-5_1` UNIX: `/opt/sybase/ESP-5_1`

Name	Description
`esp_server.log`	Project log. Captures errors and events in a running project. Resides in the working directory for the project. • **Needs shared drive? –** Optional; typically shared in a multinode cluster where the base directory is shared • **Location –** `<base-directory>/<workspace-name>.<project-name>.<instance-number>/esp_server.log`
`examples`	Examples directory. Stores sample configuration files including `<node-name>.xml`, `policy.xml`, `persistence.xml`, log properties files, and authentication files. • **Needs shared drive? –** No • **Location –** `ESP_HOME/cluster/examples`
Input files for projects	Anything a project needs at runtime—for example, input CSV files, ODBC drivers, or queueing clients. • **Needs shared drive? –** Optional. If you set controller affinities to limit which nodes the project can run on, you can store input files on the specified nodes only. But if the project needs to be able to fail over to a controller on another machine, put the input files in a shared location. • **Recommended location –** Working directory for the project: `<base-directory>/<workspace-name>.<project-name>.<instance-number>`
Log store	Log store. Saves project input and output data in case of failover. • **Needs shared drive? –** Yes • **Default location –** `<base-directory>/<workspace-name>.<project-name>.<instance-number>/<logstore-name>`
`<node-name>.xml`	Cluster node configuration file. Created automatically for clusters configured at installation, manually for clusters created after installation. Defines a node as a manager, a controller, or both, and configures security, port communication, and caching. • **Needs shared drive? –** No • **Default location –** `ESP_HOME/cluster/nodes/<node-name>/<node-name>.xml`

Name	Description
`<persistence-directory>`	Persistence directory. Saves project and workspace information when a node shuts down. The cluster configuration file, `<node-name>.xml`, sets the persistence directory's path. All managers in a cluster must use the same persistence directory. If you do not enable persistence, you lose all your projects when the last node in the cluster shuts down; therefore, in a production system, enabling persistence is recommended. • **Needs shared drive? –** Yes • **Default location –** `ESP_HOME/storage`
`policy.xml`	Policy file. Defines access control policies that specify user roles, the actions available to each role, and the resources on which the actions can be performed. • **Needs shared drive? –** Yes (Sybase recommends a shared drive for the security directory) • **Default location –** `ESP_HOME/security/policy.xml`
`<project-name>.ccl`	Definition of a project in Continuous Computation Language. A project has one `<project-name>.ccl` file. When you edit the project in Studio, you modify the CCL or an ancillary file. (The ancillary files store information about the visual presentation of the project in Studio.) • **Needs shared drive? –** N/A • **Default location –** Local (Studio) cluster: `<user's-home-dir>/SybaseESP/5.1/workspace/<project-name>/<project-name>.ccl`
`<project-name>.ccr`	Project configuration or CCR file. An XML file that contains runtime and deployment configuration for the project, including adapter, parameter, and binding definitions, A project can have multiple CCR files, allowing you, for example, to specify different adapters in different deployment environments, or to change run-time parameter values. • **Needs shared drive? –** N/A • **Default location –** Local (Studio) cluster: `<user's-home-dir>/SybaseESP/5.1/workspace/<project-name>/<project-name>.ccr`

Name	Description
`<project-name>.ccx`	The compiled CCL file. You must supply the path to the CCX file when you deploy a project to a remote (non-Studio) cluster. You do not need to view or edit the CCX file. • **Needs shared drive? –** N/A • **Default location –** Local (Studio) cluster: `<user's-home-dir>/SybaseESP/5.1/workspace/<project-name>/bin/<project-name>.ccx`
Project working directory	Contains files generated by the project. May also contain input files the project needs. See also *workspace* on page 8. • **Needs shared drive? –** Optional; typically shared in a multinode cluster where the base directory is shared • **Default location –** `<base-directory>/<workspace-name>.<project-name>.<instance-number>`
`stdstreams.log`	Standard streams log. Captures output written to stdout and stderr, including SySAM licensing information and messages from third party applications. Resides in the working directory for the project. • **Needs shared drive? –** Optional; typically shared in a multinode cluster where the base directory is shared • **Location –** `<base-directory>/<workspace-name>.<project-name>.<instance-number>/stdstreams.log`
`security`	Security directory. Stores Java keystores for RSA, LDAP policy files for access control, and the `csi_cluster.xml` file for specifying security authentication on the local cluster server. All components in a cluster are subject to the security rules defined in the cluster configuration file. To easily maintain consistent security configurations among all nodes, SAP recommends that multiple cluster managers running in the same cluster share a security folder. • **Needs shared drive? –** Yes • **Default location –** `ESP_HOME/security`

Name	Description
`service.xml`	Service configuration file. Contains database service definitions, including all the properties and parameters required for the database connections needed by the projects that run on this node or in this cluster. Every node needs access to a service configuration file. • **Needs shared drive? –** No • **Default location –** `ESP_HOME/bin/service.xml`
`workspace`	Directory where ESP Studio stores project files. You set the location during installation. By default, the workspace in a local (Studio) cluster is the same as the base directory. See also *<base-directory>* on page 3 and *project working directory* on page 7. • **Needs shared drive? –** No • **Default location –** Local (Studio) cluster: `<user's-home-dir>/SybaseESP/5.1/workspace` In Windows, `<user's-home-dir>` defaults to the `My Documents` folder. If no home directory is configured in UNIX, the base directory location defaults to `/SybaseESP/5.1/workspace`.

CHAPTER 2 Get Started with a Cluster

Plan and configure an SAP® Sybase® Event Stream Processor cluster.

1. *Planning a Cluster*

   Best practices for multiple node configuration.
2. *Adding a Node to a Cluster*

   Expand your cluster by creating and adding nodes.
3. *Configuring a Cluster*

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

   Set up authentication, access control, and SSL connections, and encrypt passwords in configuration files.
5. *Deploying a Project to a Cluster*

   To run the projects you create in ESP Studio, add them to a cluster.
6. *External Database Access*

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

Planning a Cluster

Best practices for multiple node configuration.

How Many Manager and Controller Nodes?

- If you are not concerned about failure recovery or load sharing, a single-node cluster may be enough.
- When you add nodes to the cluster, you can add them on the same host machine as the first node or on different hosts. In a production environment, Sybase recommends that you install additional nodes on different hosts, with no more than one manager and one controller per host. This allows you to take advantage of the load sharing and failure recovery features offered by the clustering architecture.
- When you add the first few nodes to a cluster, Sybase recommends that you maintain a one-to-one ratio of managers to controllers. Once you have four manager nodes, the benefit of adding more diminishes. In a medium-sized or large cluster, there are typically

more controller nodes than manager nodes—add more controllers as your portfolio of projects grows.
- If you plan to use the failover feature for failure recovery, Sybase recommends that you configure at least three managers and three controllers in your cluster.

Configuring Multiple Nodes in a Cluster

- Set the same cache name and password for all manager nodes in a cluster to access the cache.
- Specify unique names for all nodes in their cluster configuration files.
- 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. All manager nodes in a cluster share common configuration files, including keystore files, LDAP, Kerberos, and RSA files. These common files, which are located by default in `ESP_HOME/security`, must reside in a shared location that all nodes in the cluster can access.
- Put input files and output file destinations in a shared location if the project needs to be able to fail over to a controller on another machine. If the project does not need to fail over, set controller affinities to limit which nodes the project can run on and store input files and output file destinations on the specified nodes only.

See also

- *Adding a Node to a Cluster* on page 12

High Availability

In Event Stream Processor, server clusters promote failure recovery and data redundancy. Event Stream Processor supports an added level of high availability at the project level called active-active mode. Active-Active mode is also known as HA (high availability) mode; the terms are interchangeable.

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 multinode cluster can protect against server failure. When a server in such a cluster fails, the projects running on the failed server are restarted on other servers if their affinities allow it. (Affinities control which server or servers a project can run on.)

When you deploy projects in active-active mode, 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

- *Cluster Persistence, Caching, and Multicast* on page 11
- *Centralized Security* on page 12
- *Active-Active Deployments* on page 51

Cluster Persistence, Caching, and Multicast

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

When you configure a cluster, enable persistence to save projects and workspaces when the cluster shuts down. All nodes in a cluster must point to the same persistence directory, which defaults to `ESP_HOME/storage`.

Note: If the nodes in a cluster are on different machines, the persistence folder must be on a shared disk.

When persistence is disabled, you lose all your projects when the last manager node in the cluster shuts down; therefore, in a production system, Sybase recommends that you enable persistence.

Caching and Multicast

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. For all nodes added to this cluster, configure the same name and password in the Cache section of <node-name>.xml.

Managers discover and join an existing cluster cache in one of two ways:

- **Multicast –** each manager node broadcasts its connection details so other managers can join if they have the correct credentials. Recommended for testing environments where all nodes in the cluster are on the same subnet. Multicast does not work well when cluster nodes are on different subnets or when multiple clusters use the same names and ports (usually the defaults).
- **Direct connect –** each manager node uses the host name and port information in the Managers section of its `<node-name>.xml` file to discover and join the cache. Recommended for production environments.

If you enable multicast, enable it on all nodes. A cluster cannot function properly unless all its nodes use the same form of communication for caching.

See also

- *High Availability* on page 10
- *Centralized Security* on page 12
- *Configuring a Cluster* on page 13

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, native OS, and LDAP security providers. All projects running in a cluster are subject to the security rules defined for that cluster. To easily maintain consistent security configurations among all nodes in a cluster, ensure that all managers running in the same cluster share a security folder.

See also

- *High Availability* on page 10
- *Cluster Persistence, Caching, and Multicast* on page 11

Adding a Node to a Cluster

Expand your cluster by creating and adding nodes.

Install ESP on a clean host—that is, a machine with no exising ESP installations.

1. Use the Sybase Event Stream Processor installer to install the software on a clean host. See the *Installation Guide* for your platform for helpful information.
2. In a text editor, open `<node-name>.xml`, the new node's cluster configuration file. Its default location is `ESP_HOME/cluster/nodes/<node-name>/<node-name>.xml`.
3. In the Cache | Managers section of the file, look for a line similar to this:
   ```
   <Manager>localhost:19001</Manager>
   ```
4. Change `localhost` to the name of the machine on which you installed the new node. Save the file but do not close it.
5. Edit the `<node-name>.xml` files for the existing nodes in the cluster:
   a) Copy the Manager line from the new node into the Cache | Managers sections of each of the files.
   b) Copy the Manager lines from the other nodes into the new node's `<node-name>.xml`.

The Managers section of every copy of `<node-name>.xml` in the cluster contains an identical set of Manager elements. (This enables the nodes to discover and join the cluster cache.) For example, in a cluster with manager nodes named dino, astro, and scooby, the Manager elements might look like this:

```
<Managers enabled="true">
  <Manager>dino:19001</Manager>
  <Manager>astro:19002</Manager>
  <Manager>scooby:19003</Manager>
</Managers>
```

6. Save and close all the `<node-name>.xml` files.
7. Start the new node.

   If the other nodes are running, there is no need to shut them down; they will discover the new node.

See also

- *Planning a Cluster* on page 9
- *Starting a Cluster* on page 67

Configuring a Cluster

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 for every cluster node.

For each node in a cluster, you configure four basic sections of the configuration file—Controller, Manager, RPC, and Cache:

```
[...]
  <Controller enabled="true">
  </Controller>
  <Manager enabled="true" />
  <Rpc>
    <Host>dino</Host>
    <Port>19011</Port>
  </Rpc>
  <Cache>
    <Host>dino</Host>
    <Port>19001</Port>
    <Name>test-name-1</Name>
    <Password>test-password-1</Password>
    <Managers enabled="true">
      <Manager>dino:19001</Manager>
      <Manager>astro:19002</Manager>
      <Manager>scooby:19003</Manager>
    </Managers>
    <Persistence enabled="true"
```

```
      <Directory>${ESP_STORAGE}</Directory>
    </Persistence>
  </Cache>
[...]
```

Configuration varies based on whether you enable the node as a controller, a manager, or both. The node defined in the example above is enabled as both a manager and a controller. (This example and the others shown in this task come from the cluster configuration file for a UNIX-based installation of Event Stream Processor. In Windows, a node cannot be both manager and controller unless it is the only node in the cluster.)

In configuration files for manager nodes, the Cache section defines the cluster by identifying the managers that belong to the cluster's shared cache.

1. Open 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 Windows 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 – the value is derived from the environment variable defined by the macro value.
   - sysproperty – the 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. (Optional) Enable the controller:

```
<Controller enabled="true">
```

6. (Optional) In a UNIX installation, if you plan to use a Java Runtime Environment (JRE) other than the one provided with Event Stream Processor, perform this step for controller-enabled nodes.

In the Controller section of `<node-name>.xml`, in the ApplicationTypes elements for both project and ha_project, locate the line that sets the **ld-preload** property. Set **ld-preload** to point to the jsig library file provided with your runtime environment. For example:

```
<Property name="ld-preload">${ESP_HOME}/lib/libjsig.so</Property>
```

The following example shows the project application type, with definitions for the base directory, host name, service configuration file, and security directory that the application uses.

Note: In this example, `StandardStreamLog` enables `stdstream.log`, which logs all output written to stdout and stderr. This includes SySAM licensing information for Event Stream Processor, as well as messages from third-party applications that write to stdout and stderr. See *Project Logging* on page 71 for more information about `stdstream.log` files.

```
<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">${ESP_HOME}/lib/libjsig.so</Property>
      <Property name="services-file">${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>
</ApplicationTypes>
```

7. (Optional) Enable the manager:

```
<Manager enabled="true" />
```

8. (Optional) Define the host node for the RPC port, which is used for all external communication with the node. Clients such as controllers, projects, SDKs, and the cluster admin tool all connect to the manager through the RPC port.

   If your machine has multiple NICs and you do not want to use the machine's default interface (localhost), create a Host element in the Rpc section of `<node-name>.xml` and enter the name or IP address of an alternate interface. For example:

```
<Host>126.55.44.33</Host>
```

Note: If the machine is set to use a proxy server or is behind a firewall, you may be unable to start a project when ESP is configured to use a network card other than the default. In such cases, set the *no_proxy* environment variable to the names and IP addresses of every system that will communicate with ESP, plus `localhost` and `127.0.0.1`. Use the fully qualified domain names. For example, on a Linux machine named archer that does not communicate with any other system:

```
no_proxy='localhost, 127.0.0.1, archer.meadow.com, 123.45.67.89'
```

On a Windows machine, do not put quotes around the value you specify for *no_proxy*. So, on a Windows machine named fletcher that communicates with a system named archer:

```
no_proxy=localhost, 127.0.0.1, archer.meadow.com, 123.45.67.89,
fletcher.meadow.com, 123.45.67.88
```

Also verify that there are no inconsistent entries in the `/etc/hosts` file. This configuration applies to any ESP client system as well as any ESP-to-ESP communication.

If using the *no_proxy* environment variable causes communication issues such as HTTP Error 503 Service unavailable, you can disable the proxy by unsetting the *http_proxy* environment variable .

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.

    The port specified in the Port element of the Cache section is used by other manager nodes for communication related to the cluster's shared cache.

```
<Port>19001</Port>
```

11. (Optional) To define the host node for the cache, modify the Host element in the Cache section of the file.

    The Host element uses the default name localhost. To allow cluster clients from other machines to connect, change the value of Host to the name of the machine on which the cluster node is running. For example:

```
<Host>dino</Host>
```

    If your machine has multiple NICs and you do not want to use the machine's default interface (localhost), enter a name or IP address in the Host element in the Cache section to specify the network interface you want cluster clients to use. For example:

```
<Host>125.66.44.33</Host>
```

    If you specify a Host value in the Cache section of the file, it must be the same host (that is, the same name or IP address) that you give for this manager node in the Managers element.

Note: If the machine is set to use a proxy server or is behind a firewall, you may be unable to start a project when ESP is configured to use a network card other than the default. In such cases, set the *no_proxy* environment variable to the names and IP addresses of every system that will communicate with ESP, plus `localhost` and `127.0.0.1`. Use the fully qualified domain names. For example, on a Linux machine named archer that does not communicate with any other system:

```
no_proxy='localhost, 127.0.0.1, archer.meadow.com, 123.45.67.89'
```

On a Windows machine, do not put quotes around the value you specify for *no_proxy*. So, on a Windows machine named fletcher that communicates with a system named archer:

```
no_proxy=localhost, 127.0.0.1, archer.meadow.com, 123.45.67.89,
fletcher.meadow.com, 123.45.67.88
```

Also verify that there are no inconsistent entries in the `/etc/hosts` file. This configuration applies to any ESP client system as well as any ESP-to-ESP communication.

If using the *no_proxy* environment variable causes communication issues such as HTTP Error 503 Service unavailable, you can disable the proxy by unsetting the *http_proxy* environment variable .

---

12. In the Cache section, define a unique name and password for the cluster. To join the cache —and thus to join the cluster—all nodes must use the same name and password when they start.

```
  <Name>test-name-1</Name>
  <Password>test-password-1</Password>
```

When the Password element has no attributes, as shown above, ESP uses the password contained in the element to start the node. You do not supply a password when you execute the start command. To prompt for a password when the node starts, use these attributes with the Password element:

Password Attribute	Behavior
prompt	• **true –** ESP prompts for the password when you try to start the node. • **false –** ESP does not prompt for a password; the **hide**, **verify**, and **query** attributes are ignored.
hide	• **true –** ESP does not display the password as you type it. • **false –** ESP displays the password.
verify	• **true –** ESP prompts for the password twice. • **false –** ESP prompts for the password once.
query	• If a query value is present, ESP uses it to prompt for the cluster password when you attempt to start the node. • If no query value is present, ESP uses default wording for the password prompt.

The Password element with prompting enabled looks like this:

```
  <Password prompt="true" hide="true" verify="false" query="Cluster
password:">test-password-1</Password>
```

13. (Optional; not recommended for production environments) To enable multicast delivery on manager-enabled nodes, set the Multicast enabled value to `true` and enter `Group` and `Port` values.

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

All nodes in the cluster must have the same multicast status.

14. (Optional; recommended for production environments) If multicast is not enabled, enable the manager node and enter its host name and port.

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

15. For manager nodes, enable or disable persistence.

    By default, this value is set to `false`. Cluster persistence enables the node to save projects and workspaces when it shuts down. When persistence is disabled, you lose all your projects when the last manager node in the cluster shuts down; therefore, in a production system, Sybase recommends that you enable persistence.

    Note: All nodes within a cluster must point to the same persistence directory.

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

16. Put the security directory for the cluster on a shared drive; all manager nodes within the cluster must have access to it. The default location is `ESP_HOME/security`.
17. Repeat these steps for each node in the cluster.

Next

Configure security for each node, including authentication, access control, and SSL connections.

See also

- *Cluster Administrative Tool* on page 72
- *Cluster Persistence, Caching, and Multicast* on page 11

Configure Security

Set up authentication, access control, and SSL connections, and encrypt passwords in configuration files.

Security in Event Stream Processor is managed centrally, by the cluster manager. All projects running in a remote cluster are subject to the security rules defined for that cluster. For information on security for projects running in the local cluster, see the *Studio Users Guide*.

See also

- *Deploying a Project to a Cluster* on page 48

Authentication

Sybase Event Stream Processor is designed to integrate with your existing authentication framework whether you are using Kerberos, RSA, LDAP, or your operating system's native credential management system.

The type of server authentication you use is determined at install time, but you can configure the server to use a different authentication type if necessary.

When a user connects to a cluster on the ESP server, his or her credentials are verified with the active security provider. If authentication succeeds, the server considers the user a valid client, and login is completed. The user receives a session ID and, in subsequent communication, the client uses the session ID to verify itself.

Options for server authentication include:

- Kerberos - ticket-based authentication
- RSA - requires a key alias, a keystore containing a private key, and the password of the keystore
- Username/password, implemented using one of the following:
  - LDAP credentials
  - Native operating system credentials (native OS)
  - Preconfigured username/password (in `csi_local.xml`)

Note: Do not confuse server authentication–enforced when users connect to remote clusters–with authentication on the local cluster–enforced when using the **Run Project** option within Studio. Server authentication is enforced across your network and is designed for use in a production environment. Local cluster authentication is enforced only on a user's local machine and, like the local cluster itself, is intended for a test environment. Authentication on the local cluster is limited to username/password authentication and is based on the fixed username `studio` . Users can enter any password for this username to maintain a secure connection with the local cluster for the duration of the Studio session. The password is maintained in memory and is not written to a disk. When the Studio session is terminated, the password is discarded from memory. When connecting to the local cluster in a subsequent Studio session, users are once again required to provide a password for the fixed username `studio`. This password does not have to be the same password set during the previous Studio session.

Authentication on the local cluster is provided automatically; there is no additional configuration required. For details on the local cluster password, see the *Studio Users Guide*.

See also

- *Access Control* on page 30
- *Secure Sockets Layer (SSL) Connections* on page 40
- *Generating the Java Keystore* on page 40

- *Generating Pem Format Private Keys* on page 42
- *Password Encryption on Configuration Files* on page 43

Configuring Kerberos Authentication

Sybase Event Stream Processor supports Kerberos ticket-based authentication. Configuring ESP for Kerberos authentication requires that you both configure the server and set values for certain environment variables.

Configuring the Server for Kerberos

Configure the server for Kerberos authentication by modifying the `node1.xml` and `csi_kerberos.xml` files.

By default, the installation process creates a cluster configuration file called `node1.xml`. This file contains security information for the cluster, including a reference to the file that determines the authentication type. If you created a different cluster name during installation, your cluster configuration file will take the format `<node-name>.xml`.

When Kerberos is the active authentication method, the `<node-name>.xml` file refers to a `csi_kerberos.xml` file, which provides configuration information for Kerberos authentication. Event Stream Processor provides a default `csi_kerberos.xml` file in the `ESP_HOME/security` directory that you can use as-is or modify based on your specific Kerberos implementation.

If you selected Kerberos at installation time, there is no need to modify the `<node-name>.xml` . If you installed with a different authentication type, perform these steps to enable and configure Kerberos:

1. Use a text editor to open the cluster configuration file, `ESP_HOME/cluster/<node-name>/<node-name>.xml` , and locate the following lines. If they do not exist in the file, add them.

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

2. Within the <Security> section of the cluster configuration file, in the <CSI> section, change the <File> value to `csi_kerberos.xml`, as follows:

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

3. Add the following to the `ESP_HOME/security/csi_kerberos.xml` file. You need to set the option for configuring the principal value to an ESP service name. The keytab option needs to be set to show the full path of a keytab file. The following is an example of a `csi_kerberos.xml` file entry with an ESP service name of "principal" and a defined keytab path:

   ```
   <config:configuration xmlns:config="http://www.sybase.com/csi/
   2.5/config">
   ```

```
    <config:authenticationProvider
name="com.sybase.esp.cluster.security.KerberosLoginModule"/>
    <config:options name="principal" value="esp/myhost"/>
    <config:options name="keyTab" value="C:/Documents and
Settings/user/krb.keytab"/>
    <config:provider
name="com.sybase.security.core.NoSecAuthorizer"
type="authorizer"/>
    <config:provider
name="com.sybase.security.core.NoSecAttributer"
type="attributer"/>
</config:configuration>
```

4. Restart the server and all of the cluster managers.

Setting Environment Variables for Kerberos

Configuring Kerberos for ticket-based authentication requires that you set specific environment variables.

Note: *KRB5CCNAME* and *KRB5_CONFIG* are required for MIT Kerberos/KFW and Heimdal. Other C/C++ GSSAPI libraries may have their own configuration requirements.

Table 2. Environment Variables for Kerberos

Variable	Value

ESP_GSSAPI_LIB	The path of the shared library file containing the GSSAPI function implementations.
	• libgssapi32.dll on Windows for MIT KFW
	• libgssapi.so for Heimdal
	• libgssapi_krb5.so for MIT Kerberos
	Example:
	`<Root>/bin/gssapi32.dll`
	`<Root>/krb/lib/libgssapi_krb5.so`
	Note: You may need to modify either the *PATH* or *LD_LIBRARY_PATH* variables by adding additional directory paths. This is done in order to satisfy any dependency that the GSSAPI library may have.
	Here is an example of a file path for a PATH variable:
	`<Root>/bin;%PATH%`
	Here is an example of a file path for a LD_LIBRARY_PATH variable:
	`<Root>/lib:$LD_LIBRARY_PATH`

Variable	Value
ESP_SERV-ICE_NAME	The ESP cluster/service principal name. Example: `esp/myhost`

KRB5CCNAME	The ticket cache.
	Example:
	`<Root>/Documents and Settings/user/krb5cc_user`
	`<Root>/tmp/krb5cc_1000`

KRB5_CONFIG	The Kerberos configuration file used by the Kerberos Library.
	Example:
	`<Root>/kfw/krb5.ini`
	`<Root>/krb/etc/krb5.conf`

Configuring 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 – either through Studio prompts or a command argument – 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.

Configuring ESP for RSA authentication requires that you configure both the server to add an RSA authentication provider, and the client to create a keystore and generate keys.

Configuring the Server for RSA

To configure the server for RSA authentication, modify the `node1.xml` and `csi_rsa.xml` files.

By default, the installation process creates a cluster configuration file called `node1.xml`. This file contains security information for the cluster, including a reference to the file that determines the authentication type. If you created a different cluster name during installation, your cluster configuration file will take the format `<node-name>.xml`.

When RSA is the active authentication method, the `<node-name>.xml` file refers to a `csi_rsa.xml` file, which provides configuration information for RSA authentication. Event Stream Processor provides a default `csi_rsa.xml` file in the `ESP_HOME/security` directory that you can use as-is or modify based on your specific RSA implementation.

If you selected RSA at installation time, there is no need to modify the `<node-name>.xml`. If you installed with a different authentication type, perform these steps to enable and configure RSA authentication:

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

1. Use a text editor to open the cluster configuration file, `ESP_HOME/cluster/<node-name>/<node-name>.xml`.
2. Within the <Security> section of the cluster configuration file, in the <CSI> section, change the <File> value to `csi_rsa.xml`, as follows:

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

3. Restart the server and all of the cluster managers.

Configuring a New Alias for Client RSA Authentication

Use the Java **keytool** utility to create public and private keys for RSA authentication, then add the public key and RSA alias to the cluster's keystore.

RSA authentication uses public and private keys instead of passwords to authenticate with the ESP Server. Use the Java **keytool** to generate RSA keys for ESP clients.

When this task is complete, you can authenticate ESP clients and other ESP components in the cluster using the new alias plus either the new local keystore or a private key. See the *Utilities Guide* for information on RSA authentication of ESP utilities like **esp_cluster_admin**. You can also use RSA authentication for:

- ESP C and .NET SDK APIs - use the new alias with a private key file
- ESP Java SDK APIs - use the new alias with the new local keystore
- ESP bindings - use the new alias with a private key file

This procedure involves up to three keystores:

- Cluster keystore – stored on an ESP node and serves the entire cluster.
- Client keystore – a local keystore used by a Java client for RSA authentication (for example, `esp_cluster_admin --keystore`).
- Work keystore – a local keystore used to generate a new alias, public key, and private key. The work keystore can be the same as the client keystore.

1. Open a command prompt or terminal.
2. Set the ESP_JAVA_HOME environment variable to point to your Java installation.
3. Add `$ESP_JAVA_HOME/bin` to the path.
4. To create a private/public key, enter a command of this form, specifying a local work or client keystore:

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

   <alias/username> is thealias chosen by you for the private and public keys; it will function as a user name for logging in using RSA. `keystore.jks` is the file where keys are added. You can specify an absolute path to create the file in a specific directory. If no path is specified, the command assumes `keystore.jks` resides in the current directory and creates it if it is not present. The default keystore in the security directory is `ESP_HOME/security/keystore_rsa.jks`. *<password>* sets the password you use to access the private key associated with the alias.
5. Use the cluster admin tool to deploy the new public key and alias from your local keystore to the cluster keystore on the server:

   ```
   esp_cluster_admin --uri=esp[s]://host-name:port
   --auth=rsa
   --keystore=<keystore>
   --storepass=<storepass>
   --keypass=<keypass>
   --key-alias=<alias>
   > deploykey <new-username> <keystore> <storepass> <key-alias>
   [<storetype>]
   ```

   - `--auth=rsa` enables RSA authentication
   - `--keystore=<keystore>` is the location where the private/public key pairs created in step *4* on page 24 are stored (a local work keystore or client keystore)
   - `--storepass=<storepass>` is the user-defined password for the keystore
   - `--keypass=<keypass>` is the user-defined key password
   - `--key-alias=<alias>` is the user-defined alias name for the key
   - `deploykey` is the command that deploys the new public key and alias to the cluster
     - `<new-username>` is the alias you are giving to the new public key in the cluster keystore; can be the same as `<key-alias>`
     - `<keystore>` may be (but need not be) the same keystore specified in step *4* on page 24 and in `--keystore=<keystore>` in this step
     - `<storepass>` is the password for `<keystore>`
     - `<key-alias>` is the alias in the local keystore whose public key you are deploying
     - `[<storetype>]` is JKS (the default) or PKCS12

   For example:

```
$ESP_HOME/bin/esp_cluster_admin --uri=esps://bedrock:19333
--auth=rsa --key-alias=serverkey --storepass=538931 --keystore=
$ESP_HOME/security/keystore_rsa.jks
> deploykey fredf fredf.jks fredf fredf
```

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.

See also

- *Generating Pem Format Private Keys* on page 42

Username-Password Authentication

Sybase Event Stream Processor provides different options for implementing username-password authentication, namely LDAP, the operating system's native credential management system (native OS), and a preconfigured username-password option.

Configuring the Server for LDAP

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

By default, the installation process creates a cluster configuration file called `node1.xml`. This file contains security information for the cluster, including a reference to the file that determines the authentication type. If you created a different cluster name during installation, your cluster configuration file will take the format `<node-name>.xml`.

When LDAP is the active authentication method, the `<node-name>.xml` file refers to a `csi_ldap.xml` file, which provides configuration information for LDAP authentication. Event Stream Processor provides a default `csi_ldap.xml` file in the `ESP_HOME/security` directory that you can use as a template and modify based on your specific LDAP implementation. At a minimum, you must provide values for the **ServerType**, **ProviderURL**, **DefaultSearchBase**, **RoleSearchBase**, and **AuthenticationScope** parameters, as determined by your LDAP implementation.

If you selected LDAP at installation time, there is no need to modify the `<node-name>.xml` . If you installed with a different authentication type, perform these steps to enable and configure LDAP authentication:

1. Use a text editor to open the LDAP configuration file provided by Event Stream Processor, `csi_ldap.xml`, located in `ESP_HOME/security`.
2. Add implementation-specific values for the **ServerType**, **ProviderURL**, **DefaultSearchBase**, **RoleSearchBase**, and **AuthenticationScope** parameters, as well as any other parameters you want to set.

Note: While setting values for the parameters mentioned here is sufficient in the majority of cases, you may, depending on your environment settings, have to specify additional

values. The sample `csi_ldap.xml` file located in `$ESP_HOME/cluster/examples` contains additional parameters and descriptions you can use for reference.

3. Save and close the `csi_ldap.xml` file.
4. Use a text editor to open the cluster configuration file, `ESP_HOME/cluster/<node-name>/<node-name>.xml`.
5. Within the <Security> section of the cluster configuration file, in the <CSI> section, change the <File> value to `csi_ldap.xml`, as follows:

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

6. Restart the server, including all cluster managers.

Configuring Native Operating System Authentication

Configure the server for native OS authentication by modifying the `<node-name>.xml` and `csi_native_nt.xml` or `csi_native_unix.xml` files.

Native OS authentication requires the same username-password credentials that users enter to log in to their machines. Native OS authentication relies on the underlying operating system's built-in authentication framework.

By default, the installation process creates a cluster configuration file called `node1.xml`. This file contains security information for the cluster, including a reference to the file that determines the authentication type. If you created a different cluster name during installation, your cluster configuration file's name will take the form `<node-name>.xml`.

When native OS is the active authentication method, the `<node-name>.xml` file refers to either the `csi_native_nt.xml` or the `csi_native_unix.xml` file, which provide configuration information for native OS authentication for Windows and Linux/Solaris respectively. Event Stream Processor provides default `csi_native_nt.xml` and `csi_native_nt.xml` files in the `ESP_HOME/security` directory that you can use as-is, or modify based on your specific implementation.

If you performed a typical installation, or selected Native OS at installation time, there is no need to modify the `<node-name>.xml` file. If you installed with a different authentication type, perform these steps to enable and configure native OS authentication:

1. Use a text editor to open the cluster configuration file, `ESP_HOME/cluster/<node-name>/<node-name>.xml`.
2. Within the <Security> section of the cluster configuration file, in the <CSI> section, change the <File> value to `csi_native_nt.xml` or `csi_native_unix.xml`, as follows:

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

3. Restart the server and all of the cluster managers.

Next

If you are configuring native OS authentication for Linux or Solaris, configure a pluggable authentication module (PAM). This step is not necessary for Windows.

Configuring a Pluggable Authentication Module (PAM) for UNIX

If you selected the Native OS authentication option during installation, perform additional configuration to allow login using accounts on either Linux or Solaris.

1. Using a login account with root privileges, configure the pluggable authentication module for your platform:

Platform	Action
Solaris	Append the contents of the `<Install-dir>/ESP-5_1/security/pam/pam.conf` file (provided with Event Stream Processor) to the `/etc/pam.conf` file on your Solaris platform.
Linux	If you are installing on RHEL 6, copy the `<Install-dir>/ESP-5_1/security/pam/rhel6/sybase-csi` file (provided with Event Stream Processor) to the `/etc/pam.d` directory on your Linux platform. For previous versions of RHEL, copy the `<Install-dir>/ESP-5_1/security/pam/sybase-csi` file (provided with Event Stream Processor) to the `/etc/pam.d` directory on your Linux platform. **Note:** The `sybase-csi` file provided with Event Stream Processor is not compatible with the most recent SUSE Linux versions. For SUSE 11 and later, see the example at the end of this topic.

   Note: In the table above, the portion of the path that indicates the operating system might differ slightly from what is shown.

2. If the host UNIX system is not using a directory lookup for authentication (yp or NIS, for example) and authentication is carried out against the local `/etc/passwd` file, any user account that executes Event Stream Processor requires read access to `/etc/shadow`. To provide this access, use the **usermod** command to add the applicable user accounts to the shadow group. For example, for user account User_123 use: **usermod -G shadow User_123**

Example: PAM for SUSE Linux 11 and later

For SUSE 11 and later, do not use the `sybase-csi` file provided with Event Stream Processor. Instead, in your `/etc/pam.d` directory, create a `sybase-csi` file that contains:

```
# sybase-csi PAM Configuration (SUSE style)
auth       include      common-auth
account    include      common-account
```

```
password   include      common-password
session    include      common-session
```

Enabling the Preconfigured Username-password Option

Enable the preconfigured username-password authentication option by modifying the `node1.xml` and `csi_local.xml` files.

By default, the installation process creates a cluster configuration file called `node1.xml`. This file contains security information for the cluster, including a reference to the file that determines the authentication type. If you created a different cluster name during installation, your cluster configuration file will take the format `<node-name>.xml`.

When preconfigured username-password is the active authentication method, the `<node-name>.xml` file refers to the `csi_local.xml` file, which contains the username and password you configure. Event Stream Processor provides a default `csi_local.xml` file you can use as a basis for creating your own preconfigured username-password combination. This file may initially reside in the `ESP_HOME/cluster/examples` folder. In such cases, copy the file to the `ESP_HOME/security` folder.

Enabling the preconfigured username-password option requires that you create and encrypt the password, configure `csi_local.xml` to specify a username and enter the encrypted password, then modify the `<node_name>.xml` file to specify the preconfigured username-password option as the active authentication method.

1. Create and encrypt a password:
   a) From a command line, run the **encode_text** command in the `esp_cluster_admin` tool:

      On Windows: **esp_cluster_admin.exe --encode_text**

      On Linux or Solaris: **esp_cluster_admin.bin --encode_text**
   b) At the prompts, enter and then confirm the password you want to encrypt.
   c) The encrypted password displays on screen. Copy the password to paste it into the `csi_local.xml` file.
2. Modify the `csi_local.xml` file to specify the username and password:
   a) Copy the `csi_local.xml` file from `ESP_HOME/cluster/examples` to `ESP_HOME/security`.
   b) Use a text editor to open `ESP_HOME/security/csi_local.xml` .
   c) In the <Configuration> section, locate <options name="username" and change its value to the desired user name.
   d) In the <Configuration> section, locate <options name="password" and change the value by pasting the encrypted password you copied from the CSI utility. Enclose the password in quotation marks.
   e) Save and close the `csi_local.xml` file.

3. Set the preconfigured username-password option as the active authentication method:
   a) Use a text editor to open the cluster configuration file, `ESP_HOME/cluster/<node-name>/<node-name>.xml`.
   b) Within the <Security> section of the cluster configuration file, in the <CSI> section, change the <File> value to `csi_local.xml`, as follows:

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

   c) Save and close the file.
4. Restart the server and all of the cluster managers.

Configuring Cascading Authentication Methods

Configure Event Stream Processor to cascade through the list of authentication methods to automatically enable the first available method.

If, for example, you are using LDAP authentication and your LDAP server goes down, users will no longer be able to authenticate using LDAP. Cascading through the authentication methods allows the ESP server to continue authenticating users without manual intervention.

As with systems using singularly-enabled authentication methods, the `<node_name>.xml` file references a `csi_*.xml` file that contains the authentication parameters. The difference with a system using cascading authentication is that this `csi_*.xml` file contains the <authenticationProvider> definitions for all the authentication methods you want to make available.

To configure your system for cascading authentication, use one of the `csi_*.xml` files provided in `ESP_HOME/security`, such as `csi_kerberos.xml`, as a basis for a new CSI file, called, for example, `csi_all.xml`. Into the new `csi_all.xml` file, copy the <authenticationProvider> definitions from each of the `csi_*.xml` files corresponding to the authentication methods you want to enable. For example, if you want to cascade through the Kerberos, RSA, and LDAP authentication methods, copy the <authenticationProvider> definition from each of the `csi_kerberos.xml`, `csi_rsa_xml`, and `csi_ldap.xml` files.

The ESP server iterates through the list of providers in order, starting with the first one in the list. Under normal circumstances, this will be the authentication method for your system. If that method becomes unavailable, the server looks at the second method in the list and tries to use it. If this attempt is successful, users can continue authenticating against the server (using the new authentication method) with no manual intervention required.

With each subsequent authentication request, the server returns to the top of the list of authentication providers and tries them in order. Therefore, when the preferred authentication method becomes available again, the server reverts to that method; there is no need to restart the server or the cluster managers.

To configure Event Stream Processor to use cascading authentication:

1. Open the desired `csi_*.xml` file and save it with a different name, such as `csi_all.xml`.
2. For each authentication method you want to make available, open the corresponding `csi_*.xml` file.
3. In each file, copy the <authenticationProvider> section and paste it into `csi_all.xml`. Place the sections in the order you want them enabled. The authentication method placed earliest in the file is used first. If that method is not available, the server cascades through the list, in order, until it encounters an authentication method it can use.
4. Save and close `csi_all.xml`.
5. Use a text editor to open the cluster configuration file, `ESP_HOME/cluster/<node-name>/<node-name>.xml`.
6. Within the <Security> section of the cluster configuration file, in the <CSI> section, change the <File> value to `csi_all.xml`, as follows:

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

7. Save and close the file.
8. Restart the server, including all cluster managers.

Access Control

Access control is an additional layer of security available with LDAP, preconfigured login, and native OS authentication.

If you choose to use access control, you can limit user activities in a more granular fashion than simple login authentication allows. You configure access control in part by creating role-based policies in the cluster's policy file. The policies enable the cluster to restrict users' access to resources like projects, workspaces, and streams.

To use access control, you must enable it in `<node-name>.xml`. For authentication through your native OS or preconfigured logins, you must also enable access control in the CSI security files for those authentication types. The sections that follow explain how to perform all the configuration required for access control.

See also

- *Authentication* on page 19
- *Secure Sockets Layer (SSL) Connections* on page 40
- *Generating the Java Keystore* on page 40
- *Generating Pem Format Private Keys* on page 42
- *Password Encryption on Configuration Files* on page 43

Roles, Resources, and Actions

To restrict user access through the access control system, each user must have a defined role. This role must be associated with resources and authorized actions for the resources. You configure roles, resources, and actions in the `policy.xml` file.

Roles

Roles in the `policy.xml` file are equivalent to group names, which are defined in the security provider (LDAP or your operating system). In the access control process, the security provider server determines whether 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.

The special *any role includes everyone. 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 and Policy Types

A policy.xml file can include policies of three types: Cluster, project, and Node.

Cluster policies apply to these resources:

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

Cluster resources are not hierarchical and do not support inheritance of entitlement.

The Project policy type includes resources such as streams and windows (which run in the project server) Resources in the policy file are defined in a file path or tree-like hierarchy 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 the policy file like this:

- `<Resource>workspace1</Resource>`
- `<Resource>workspace1/project1</Resource>`
- `<Resource>workspace1/project1/stream1</Resource>`
- `<Resource>workspace1/project1/window1</Resource>`

For Project resources, Event Stream Processor supports inheritance of entitlement: a user who is authorized for an action for resource workspace1 is automatically authorized the same action for all resources under workspace1 in the hierarchy.

The special *any resource refers to all the resources available for a policy type. *any is especially useful for the Project policy type because there are so many possible resources. You cannot define the *any resource in a granular fashion, such as workspace1/*any.

The Node policy type applies only to the Node resource. To enable Sybase Control Center to monitor a node, you must add a Node policy to the node's `policy.xml` file. For details, see the online help for *Sybase Control Center for Event Stream Processor*.

Node resources use only the READ and STOP actions.

Actions

You can specify four actions (access methods) for resources in `policy.xml`: READ, WRITE, START, and STOP. The availability and meaning of each action depend on the policy type and resource.

Policy Type	Resource	Action	Description
Cluster	Application (project)	READ	Get the list of projects and information about the projects. Get streams, windows, and schemas. Monitor projects and streams. Monitor connections to projects, streams, and windows.
Cluster	Application (project)	WRITE	Add projects to the cluster or remove them from the cluster.
Cluster	Application (project)	START	Start projects in the cluster.
Cluster	Application (project)	STOP	Stop projects in the cluster.
Cluster	Node	READ	Get the list of managers and controllers and information about those nodes.
Cluster	Node	STOP	Stop nodes.
Cluster	Security	WRITE	Upload the policy file. Add a user by deploying a public key to the cluster's keystore.
Cluster	Workspace	READ	Get the list of workspaces in the cluster and information about the workspaces.
Cluster	Workspace	WRITE	Add workspaces to the cluster or remove them from the cluster.
Cluster	*any	–	Encompasses all Cluster resources. Set READ, WRITE, START, and STOP actions as for the other Cluster resources. Actions are ignored for resources that do not support them.

Policy Type	Resource	Action	Description
Node	Node	READ	Get the list of nodes and information about those nodes. Use to enable monitoring by Sybase Control Center.
Node	Node	STOP	Stop nodes. Use to enable management by Sybase Control Center.
Project	Project path	READ	Subscribe to streams and windows in the project.
Project	Project path	WRITE	Publish to all streams and windows in the project. Play back to all streams and windows in the project. Upload to all streams and windows in the project.
Project	Project path	START	Start all adapters in the project.
Project	Project path	STOP	Stop all adapters in the project.
Project	Stream or window path	READ	Subscribe to a stream or window.
Project	Stream or window path	WRITE	Publish to the stream or window. Play back to the stream or window. Upload to the stream or window.
Project	Stream or window path	START	Start adapters attached to the stream or window.
Project	Stream or window path	STOP	Stop adapters attached to the stream or window.
Project	Workspace path	READ	Subscribe to all streams and windows in the workspace.
Project	Workspace path	WRITE	Publish to all streams and windows in the workspace. Play back to all streams and windows in the workspace. Upload to all streams and windows in the workspace.
Project	Workspace path	START	Start all adapters in the projects in the workspace.
Project	Workspace path	STOP	Stop all adapters in the projects in the workspace.
Project	*any	–	Encompasses all Project resources. Set READ, WRITE, START, and STOP actions as for the other Project resources.

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

Define access control policies that specify user roles, the actions available to each role, and the resources on which the actions can be performed. If Event Stream Processor is configured to authenticate through the native OS or preconfigured logins, enable access control. Set up role mapping.

Access control policies are maintained in a single XML policy file used by all manager nodes in a cluster. If no access control policies are defined, authorization is not restricted based on user roles, and therefore all authenticated users will have full access.

Access control is enabled by default for LDAP authentication. To enable access control for native OS or preconfigured login authentication, edit the appropriate csi files in the `security` directory, as described in the steps below. If the roles you configure in the policy file do not have names identical to the names of groups in LDAP or your OS, you must also configure role mappings in `ESP_HOME/security/csi_role_mapping.xml`. Role mappings link roles in the policy file to OS or LDAP groups.

The policy file, `policy.xml`, is loaded automatically when you start the cluster manager. If you modify the policy file, use the cluster admin tool to reload it at runtime.

1. Use any text editor to open the policy file, `ESP_HOME/security/policy.xml`.
2. To start a new policy, add <Policy> tags to the <Policies> element.

   You can include more than one <Policy> within the <Policies> tags.
3. Specify the policy type as Project, Node, or Cluster. For example:

```
<Policy type="Project">
```

4. 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 roles defined in one <Policy> element are associated with the same set of resources and actions. For a role with different resources and actions, create a separate policy using the <Policy> tag.
5. Add a group or role to the new role being created within the <Role> tags.
6. To associate resources with the role, specify each resource with <Resource> tags, and enclose these in the <Resources> element.
7. To associate actions with the resources, specify each action (read, write, start, or stop) with <Action> tags and enclose these in the <Actions> telement.

8. Save and exit the file.
9. (Optional) If you are configuring access control for use with native OS authentication, edit `ESP_HOME/security/csi_native_nt.xml` or `ESP_HOME/security/csi_native_unix.xml` to enable access control.
   a) Put comment tags (`<!--` and `-->`) around the line that configures the NoSecAuthorizer provider.
   b) Uncomment the line that configures the RoleCheckAuthorizer provider.
   c) If the roles in your policy file do not correspond to existing groups in your OS, also uncomment the lines that configure the XMLFileRoleMapper provider and specify the role map file, `csi_role_mapping.xml`.

   This sample `csi_native_unix.xml` file enables access control and role mapping (RoleCheckAuthorizer and XMLFileRoleMapper, which points to `csi_role_mapping.xml`, are outside the comment tags, while NoSecAuthorizer is inside).

```
<?xml version="1.0" encoding="UTF-8"?>
<config:configuration xmlns:config=http://www.sybase.com/csi/2.5/config
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <config:authenticationProvider controlFlag="sufficient"
name="com.sybase.security.os.UnixProxyLoginModule"/>
  <config:provider name="com.sybase.security.core.NoSecAttributer"
type="attributer"/>
  <config:provider name="com.sybase.security.core.RoleCheckAuthorizer"
type="authorizer"/>
  <config:provider name="com.sybase.security.core.XMLFileRoleMapper"
type="roleMapper">
    <config:options name="RoleMapFile" value="${esp.home}/security/
csi_role_mapping.xml"/>
  </config:provider>
  <!--
  <config:provider name="com.sybase.security.core.NoSecAuthorizer"
type="authorizer"/>
  -->
</config:configuration>
```

10. (Optional) If you are configuring access control for use with preconfigured logins, edit `ESP_HOME/security/csi_local.xml` to enable access control.
    a) Put comment tags (`<!--` and `-->`) around the line that configures the NoSecAuthorizer provider.
    b) Add this line inside the <configuration> element

```
<provider name="com.sybase.security.core.RoleCheckAuthorizer"
type="authorizer"/>
```

11. (Optional) For preconfigured logins, configure roles in `csi_local.xml` that match the ones in `policy.xml`.

    Add roles in the <options> element, and put the <options> element in the same <authenticationProvider> element as the login's user name and password. You can include multiple role names in the value field—separate them with commas. This sample

<authenticationProvider> defines a login called sybase whose roles are espAdmin, espUser, and investment:

```
  <authenticationProvider controlFlag="sufficient"
name="com.sybase.security.core.PreConfiguredUserLoginModule">
    <options name="username" value="sybase"/>
    <options name="password"
value="{SHA-256:gIQWZYOPQVM=}jqHtsTPcw8kGkZt1PQeveUAhQncAQhHXJBrjZAqTfk4
="/>
    <options name="roles" value="espAdmin,espUser,investment"/>
  </authenticationProvider>
```

If the file contains more than one login, configure roles for each one.

12. (Optional) If you are using role mapping with LDAP or native OS authentication, modify `ESP_HOME/security/csi_role_mapping.xml` to map roles specified in your policy file to groups in LDAP, Windows, or UNIX.

Use a Mapping element for each mapping. LogicalName is the role in your policy file; MappedName is the group whose members need that role. This is a sample mapping:

```
<Mapping>
  <LogicalName>investment</LogicalName>
  <MappedName>espInvestmentRole</MappedName>
</Mapping>
```

This is a sample policy file. The investment role enables users to 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>
```

Note: You assign users to groups through the security provider (LDAP, the operating system, or for preconfigured logins, the `csi_local.xml` file).

Next

Enable access control in `node-name.xml`.

See also

- *Enabling or Disabling Access Control* on page 38

Sample Policies for Authorization Roles

Use the `policy.xml` file to control access 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>read</Action>
        <Action>write</Action>
        <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>
```

Enabling or Disabling Access Control

To enable access control, set the location of the policy file in `<node-name>.xml`. To disable it, comment the policy line out.

Prerequisites

- Create role-based access control policies in the `policy.xml` file.
- Enable access control for native OS or preconfigured logins in CSI files. (Access control is enabled by default for LDAP.)

- (Optional) Configure role mappings.

Task

By default, the location of the policy file is commented out of the cluster node configuration file.

1. Edit the node's configuration file, `ESP_HOME/cluster/nodes/<nodename>/<node-name>.xml`, to uncomment the line that points to the policy file. In the Csi element in the Security section, change this:

   ```
   <!--Policy>${ESP_HOME}/security/policy.xml</Policy-->
   ```

   To this:

   ```
   <Policy>${ESP_HOME}/security/policy.xml</Policy>
   ```

   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.
2. To disable access control, open `ESP_HOME/cluster/nodes/<nodename>/<node-name>.xml` and comment out the Policy element (in Csi in the Security section):

   ```
   <!--Policy>${ESP_HOME}/security/policy.xml</Policy-->
   ```

   The server performs no access control checking; any authenticated user can perform any action on any resource.

See also

- *Configuring Access Control* on page 34

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 `ESP_HOME/security/csi_ldap.xml` file to point to a replica LDAP server.
- Modify your Kerberos configuration.
- Modify the `ESP_HOME/security/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

- *Authentication* on page 19
- *Access Control* on page 30
- *Generating the Java Keystore* on page 40
- *Generating Pem Format Private Keys* on page 42
- *Password Encryption on Configuration Files* on page 43

Generating the Java Keystore

Java keystores provide a convenient mechanism for storing and deploying X.509 certificates and private keys. Use keystores with Secure Sockets Layer (SSL) to have the ESP Server store and read the key. Use keystores to encrypt passwords for external servers and applications (like databases) to avoid storing passwords as clear text in configuration file. Also use keystores for RSA authentication because it stores user certificates.

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.

See also

- *Authentication* on page 19
- *Access Control* on page 30
- *Secure Sockets Layer (SSL) Connections* on page 40
- *Generating Pem Format Private Keys* on page 42
- *Password Encryption on Configuration Files* on page 43

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**.

See also

- *Authentication* on page 19
- *Access Control* on page 30
- *Secure Sockets Layer (SSL) Connections* on page 40
- *Generating the Java Keystore* on page 40
- *Password Encryption on Configuration Files* on page 43

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.

These configuration files may include encrypted or encryptable passwords:

- Configuration files (for example `adapter.xml`, `adapter_config.xml`) for adapters under `ESP_HOME/adapters`
- The project configuration (CCR) file, `<project-name>.ccr`
- The service configuration file for each project: default location `ESP_HOME/bin/service.xml`

See also

- *Authentication* on page 19
- *Access Control* on page 30
- *Secure Sockets Layer (SSL) Connections* on page 40
- *Generating the Java Keystore* on page 40
- *Generating Pem Format Private Keys* on page 42

Calling esp_cluster_admin

Log in to 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.

Note: If you omit the password parameter when you call the esp_cluster_admin tool, Event Stream Processor prompts you for the password and hides it as you type, which improves security.

2. Run commands as desired. Enter **help** to view a list of the utility's commands.
3. To exit the utility, enter **exit** or **quit**.

Encrypting Passwords for Configuration Files

The **esp_cluster_admin** tool includes commands that let you encrypt passwords to avoid having sensitive data in plain text within configuration files.

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. Within the configuration file, copy the password text you want to encrypt.

   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>
```

3. From a command line, navigate to `ESP_HOME/bin` and launch the `esp_cluster_admin` tool using the **encrypt_text** argument and the password you want to encrypt. This command also requires host and port information as well as credentials for the ESP server. For example, where `<text>` is the password you want to encrypt, the syntax is:

```
esp_cluster_admin --uri=esp://<host>:<port> --username=<username>
--password=<passwords> --encrypt_text --text=<text>
```

   The `esp_cluster_admin` tool writes the encrypted password to the display.

Note: If you omit the password parameter when you call the esp_cluster_admin tool, Event Stream Processor prompts you for the password and hides it as you type, which improves security.

Note: To encode passwords in a CSI configuration file (`csi_*.xml`, stored by default in `ESP_HOME/security`), use the **encode_text** command. See the *Utilities Guide* for details.

4. Copy and paste the encrypted text from the utility into the configuration file you opened in Step 1. Replace the original password in the **Password** parameter with the encrypted text, then change the `encrypted="false"` attribute for the parameter to `encrypted="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.
5. Save and close 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
Native OS (user name/password)	user_password

   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.

Deploying a Project to a Cluster

To run the projects you create in ESP Studio, add them to a cluster.

Prerequisites

Create a project.

Task

1. Set project options using the Project Configuration view in ESP Studio or by editing the project's CCR file.

   Read about project options in the sections that follow.
2. Add the project to the node:

```
esp_cluster_admin --uri=esp://cluster_server:19011 --username=me
--password=sybase
> add project <workspace-name>/<project-name> <project-name.ccx>
>
```

   Note: If you omit the password parameter when you call the esp_cluster_admin tool, Event Stream Processor prompts you for the password and hides it as you type, which improves security.

See also

- *Configure Security* on page 18
- *External Database Access* on page 57

Project Deployment Options

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

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.

This table outlines the project options you can set using the Project Configuration view in ESP Studio or by editing the CCR file.

Note: When you change options in a deployed project, use Studio or **esp_cluster_admin** to stop and remove the project from the node, then redeploy (add) the project.

Project Option	Description
Debug Level	Set a logging level for debugging the project, ranging from 0 to 7. The default level is 3. Each number 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
Performance Monitor Refresh Interval	Define performance monitor refresh interval 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, performance monitor refresh interval is set to 5. Set this option to 0 to disable monitoring; this also optimizes performance.
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.
Bad Record File	To save bad records to a file, select the bad-record-file option for **Project Type**, and indicate the file name of an ESP project for **Value Field**. If a file name is not specified, bad records are discarded. Default file name is esp_bad_record_file.
Utf8	Enable Utf8 functionality on the server. Default value is true, set to false to disable.
Web Service Enabled	When this value is set to true, it enables project access to Web services so that Web services clients can connect to the ESP Web Services Provider. This connection allows access to project data and can be used to publish data to project streams and windows. Default value is false.
Optimize	Suppresses redundant store updates. Default value is true, set to false to disable.

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.
Time Interval	Set the constant interval expression that specifies the maximum age of rows in a window, in seconds. Default value is set to 1.
Precision	Set decimal display characteristics for number characters in the project. Default value is 6.
Memory	Set memory usage limits for the project. Default is 0, meaning unlimited.
Command Port	Set an explicit command port number if you need to expose the port outside the firewall. Otherwise, do not modify this value. If you set an explicit command port, ensure that port is available on all machines that can run the project. If the port is 0, the program selects an arbitrary port. To define a specific port, set a value between 1 and 65535. Default value is 0.
SQL Port	Set an explicit SQL port number if you need to expose the port outside the firewall. Otherwise, do not modify this value. If you set an explicit SQL port, ensure that port is available on all machines that can run the project. If the port is 0, the program selects an arbitrary port. To define a specific port, set a value between 1 and 65535. Default value is 0.

Project Option	Description
Gateway Port	Set an explicit gateway port number if you need to expose the port outside the firewall. Otherwise, do not modify this value. If you set an explicit gateway port, ensure that port is available on all machines that can run the project. If the port is 0, the program selects an arbitrary port. To define a specific port, set a value between 1 and 65535. Default value is 0.

See also

- *Active-Active Deployments* on page 51
- *Project Instances* on page 52
- *Affinities* on page 53
- *Failover* on page 54
- *Sample Project Configuration File* on page 54

Active-Active Deployments

For high availability at the project level, configure active-active mode to create two instances of a project that run simultaneously.

To deploy a project in active-active or HA (high availability) mode, set `<Project ha="true">` in the CCR file. In an active-active deployment, two instances of a project run simultaneously in a cluster. Active-active projects are typically configured so that the cluster starts the two instances of the project on different nodes (hosts). This feature avoids the risk of a single point of failure at the project level.

One instance of the project is elected as the 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.

When you configure an active-active project, you can set instance affinities to control whether the instances can run on the same node.

This example shows the Deployment section of the CCR file for an HA deployment with failover enabled and affinities configured on both instances of an active-active project.

```
<Deployment>
  <Project ha="true">
    <Options>
      <Option name="debug-level">1</Option>
    </Options>
```

```
    <Instances>
      <Instance name="primary">
        <Affinities>
        <!-- Affinities are optional.  -->          <Affinity
type="controller" charge="positive" strength="weak" value="node1"/>
          <Affinity type="instance" charge="negative" strength="strong"
value="secondary"/>
        </Affinities>
      </Instance>
      <Instance name="secondary">
        <Affinities>
        <!-- Affinities are optional.  -->
          <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> <!-- in seconds -->
          <FailuresPerInterval>4<FailuresPerInterval> <!-- counter -->
        </Failover>
      </Instance>
    </Instances>
  </Project>
</Deployment>
```

See also

- *Project Deployment Options* on page 48
- *Project Instances* on page 52
- *Affinities* on page 53
- *Failover* on page 54
- *Sample Project Configuration File* on page 54
- *High Availability* on page 10

Project Instances

High availability increases resiliency to failure by creating two project instances that run simultaneously.

When a project is deployed in HA (active-active) mode, two instances are created: primary and secondary. Whether the project is in HA mode or not, you can set affinity and cold failover options for each instance, including failover intervals and failure per interval options. Non-HA projects have one instance, numbered 0 (zero). HA project instances are numbered 0 and 1. Some commands require instance numbers to identify instances of a project.

See also

- *Project Deployment Options* on page 48
- *Active-Active Deployments* on page 51
- *Affinities* on page 53

- *Failover* on page 54
- *Sample Project Configuration File* on page 54

Affinities

Set controller and instance affinities to determine which nodes a project can run on.

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

- Controller – for active-active and non-active-active configurations. Controller affinities let you establish rules and preferences as to which controller nodes your project can run on. A project can have affinities for more than one controller, but it can have a strong positive affinity for only one controller.
- Instance – only for active-active configurations. The two instances of an active-active project can have affinities for each other. For example, if you want such instances never to run on the same node, set strong negative instance affinities. If you want them to avoid running on the same node if possible, set weak negative instance affinities.

Define these parameters 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 must refer to the second instance.
Strength	Specify **Strong** or **weak**. Strong requires the project to run on a specific controller, and no others. If you have strong positive affinity set for a controller node, and that node fails, the failover process tries to restart the project on that node. If the node has not recoverd, the project restart fails and you must restart manually. If the affinity is weak, the project preferentially starts on the defined controller, but if that controller is unavailable, it may start on another available controller.
Charge	Specify **Positive** or **negative**. If positive, the project runs on the controller. If negative, the project does not run on the controller.

When failover is enabled for a project, affinities can affect restarts. Suppose your project has a strong positive affinity for controller node A—that is, the project can run only on controller A. Suppose further that controller A crashes while your project is running. When your project tries to restart, controller A is still down. A project cannot attempt more than one restart if no appropriate controller is available for the project to run on. Because of its strong positive affinity for controller A, there is at most one appropriate controller to try, so your project can try to restart only once. You must restart the project manually when controller A returns to service or reconfigure the affinities.

See also

- *Project Deployment Options* on page 48
- *Active-Active Deployments* on page 51
- *Project Instances* on page 52
- *Failover* on page 54
- *Sample Project Configuration File* on page 54

Failover

Enable or disable failover and set failover options.

A project fails when it does not run properly or stops running properly. If failover is enabled, a failover occurs when a failed project switches to another server to continue processing. Failover typically results in a project restart, though a strong positive affinity to a node that is not available can prevent a project from restarting. Restarts can be limited based on 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.
Failures per interval	Specifies the number of restarts the project can attempt within a given interval. This count resets to zero if you restart the project manually 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 makes up an interval. If left blank, the interval time is infinite.

See also

- *Project Deployment Options* on page 48
- *Active-Active Deployments* on page 51
- *Project Instances* on page 52
- *Affinities* on page 53
- *Sample Project Configuration File* on page 54

Sample Project Configuration File

Use these project configuration CCR file examples to build and modify your XML-based project configuration file.

You can build and modify the project configuration CCR file using the Studio Project Configuration Editor or a text editor.

In this example, notice that the CCR file is organized in sections according to the preferences being set, including clusters, managers, bindings, parameters, adapters, and project settings. (For information on bindings and parameters, see the *Studio Users Guide*. For information on adapters and adapter property sets, see the *Studio Users Guide* and the *Adapters Guide*.)

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration xmlns="http://www.sybase.com/esp/project_config/
2010/08/">
  <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="remote">
        <Username>user2</Username>
        <Password>Pass1234</Password>
        <!-- Managers section is required for a remote cluster. Managers
for a local cluster are retrieved internally from the node. -->
        <Managers>
          <Manager>host:19011</Manager>
        </Managers>
      </Cluster>
    </Clusters>
    <Bindings>
      <Binding name="stream1">
        <Cluster>cluster1</Cluster>  <!-- this is always needed. -->
        <Workspace>w1</Workspace>
        <Project>p1</Project>
        <BindingName>b1</BindingName>
        <RemoteStream>remote1</RemoteStream>
      </Binding>
      <Binding name="stream2">
        <Cluster>cluster2</Cluster>  <!-- this is always needed -->
        <Workspace>w2</Workspace>
        <Project>p2</Project>
        <BindingName>b2</BindingName>
        <RemoteStream>remote2</RemoteStream>
      </Binding>
      <Binding name="stream3">
        <Cluster>cluster3</Cluster>  <!-- this is always needed -->
        <Workspace>w3</Workspace>
        <Project>p3</Project>
        <BindingName>b3</BindingName>
        <RemoteStream>remote3</RemoteStream>
      </Binding>
    </Bindings>
    <Parameters>
      <Parameter name="myparam1">foo</Parameter>
      <Parameter name="myparam2">1234</Parameter>
      <Parameter name="myparam3">true</Parameter>
    </Parameters>
    <AdaptersPropertySet>
      <PropertySet name="datalocation1">
```

```
        <Property name="myhost1">5555</Property>
      </PropertySet>
      <PropertySet name="datalocation2">
        <Property name="myhost2">6666</Property>
      </PropertySet>
    </AdaptersPropertySet>
  </Runtime>
  <Deployment>
    <Project ha="false">
      <Options>
        <Option name="time-granularity" value="5"/>
        <Option name="debug-level" value="3"/>
        <Option name="java-max-heap" value="512"/>
      </Options>
      <Instances>
        <Instance>
          <Failover enable="false"/>
          <Affinities>
            <Affinity charge="positive" strength="strong"
type="controller" value="myController"/>
          </Affinities>
        </Instance>
      </Instances>
    </Project>
  </Deployment>
</Configuration>
```

This example shows the Deployment section of the CCR file for an HA deployment with failover enabled and affinities configured on both instances of an active-active project.

```
<Deployment>
  <Project ha="true">
    <Options>
      <Option name="debug-level">1</Option>
    </Options>
    <Instances>
      <Instance name="primary">
        <Affinities>
        <!-- Affinities are optional.  -->          <Affinity
type="controller" charge="positive" strength="weak" value="node1"/>
          <Affinity type="instance" charge="negative" strength="strong"
value="secondary"/>
        </Affinities>
      </Instance>
      <Instance name="secondary">
        <Affinities>
        <!-- Affinities are optional.  -->
          <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> <!-- in seconds -->
          <FailuresPerInterval>4<FailuresPerInterval> <!-- counter -->
        </Failover>
```

```
      </Instance>
    </Instances>
  </Project>
</Deployment>
```

See also

- *Project Deployment Options* on page 48
- *Active-Active Deployments* on page 51
- *Project Instances* on page 52
- *Affinities* on page 53
- *Failover* on page 54

External Database Access

The ESP 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, ODBC, or Open Client™ connection with the appropriate JDBC, ODBC, or OCS driver for the desired external database installed.
- Specify service definitions in the `service.xml` file for your desired database connections.

Adapter	Supported Drivers	Supported Databases
Adaptive Server® Enterprise Output Adapter	Open Client™	Adaptive Server Enterprise
Database Input and Output Adapters	JDBC	• Adaptive Server Enterprise • IBM DB2 • Oracle • Kx Systems KDB+ • Microsoft SQL Server • SAP HANA®

Adapter	Supported Drivers	Supported Databases
Database Input and Output Adapters	ODBC	• Adaptive Server Enterprise • Sybase IQ • SQL Anywhere® • IBM DB2 • Oracle • Microsoft SQL Server • TimesTen • MySQL 5.x • PostgreSQL • SAP HANA
SAP HANA Adapter	ODBC	SAP HANA
Sybase IQ	ODBC	Sybase IQ

On UNIX systems, SAP recommends upgrading to version 2.3.0 or later of unixODBC. If you are using a version lower than 2.3.0, set a parameter for the driver that instructs the database manager not to synchronize database access. To do this, add a line that says "Threading = 0" for your driver in the `odbcinst.ini` file.

If you are running the SAP HANA Output adapter on a UNIX system, use only unixODBC 2.3.0 or higher.

- If you are using version 2.3.0, add "Threading=0" in the `odbcinst.ini` file to ensure optimal adapter performance.
- If you are using version 2.3.1, create a symbolic link under `<2.3.1 installation folder>/lib` as follows:

  ```
  ln -s libodbc.so.2.0.0 libodbc.so.1
  ```

  This link is required because ESP links to `libodbc.so.1`, which unixODBC 2.3.1 has renamed `libodbc.so.2`. With the link, ESP will now use `libodbc.so.2`.

See also

- *Deploying a Project to a Cluster* on page 48

Configure Connections to External Databases

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

Prerequisites

- On UNIX systems, SAP recommends upgrading to version 2.3.0 or later of unixODBC. If you are using a version lower than 2.3.0, set a parameter for the driver that instructs the

database manager not to synchronize database access. To do this, add a line that says "Threading = 0" for your driver in the `odbcinst.ini` file.

- If you are running the SAP HANA Output adapter on a UNIX system, only use unixODBC 2.3.O or higher.
  - If you are using version 2.3.0, add "Threading=0" in the `odbcinst.ini` file to ensure optimal adapter performance.
  - If you are using version 2.3.1, create a symbolic link under `<2.3.1 installation folder>/lib` as follows:

    ```
    ln -s libodbc.so.2.0.0 libodbc.so.1
    ```

    This link is required because ESP links to `libodbc.so.1`, which unixODBC 2.3.1 has renamed `libodbc.so.2`. With the link, ESP will now use `libodbc.so.2`.

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>
```

See also

- *Linking to Your ODBC Driver Manager Library* on page 65

Configuring a JDBC Connection to an External Database

Create a service definition for a JDBC connection to the database of your choice.

To set up a JDBC connection from within Event Stream Processor, edit `ESP_HOME/bin/service.xml`.

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:

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
SAP HANA	esp_db_jdbc_lib

5. If you are connecting to SAP HANA, set the **DataSource** parameter as shown:

```
<Parameter  Name="DataSource">com/sap/db/jdbcext/DataSourceSAP</
Parameter>
```

6. 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.

7. 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.

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

   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. For guidelines on the format of this parameter, see the documentation provided with the database to which you wish to connect. 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>`

Configuring an ODBC Connection to an External Database

Create a service definition for an ODBC connection to the database of your choice.

To set up an ODBC connection from within 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 to which you want to connect:

Database	DriverLibrary Value
All supported databases	esp_db_odbc_lib or esp_db_odbc64_lib

Note: ForWindows, the **DriverLibrary** parameter can only be set to esp_db_odbc_lib. For Linux/Unix, there exist two types of driver managers. One is built with the SQLLEN size set to 32-bits and the other with SQLLEN set to 64-bits. Drivers are only compatible with one or the other of the driver managers. For Linux/Unix, set the **DriverLibrary** parameter to

esp_db_odbc_lib if your driver requires 32-bit SQLLEN, or esp_db_odbc64_lib if your driver requires 64-bit SQLLEN.

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. 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.

Configuring an Open Client Connection to a Database

Create a service definition for an Open Client (OCS) connection to the Adaptive Server Enterprise database. OCS connections are supported only through the ASE Output adapter.

To set up an OCS connection from within 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 `esp_db_ocs_lib` to connect to the Adaptive Server Enterprise database.
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. (Optional) Set the **TDSPacketSize** parameter for optimal performance. If not set, the default value for Open Client is used.
   See the **CS_PACKETSIZE** connection property in the Open Client documentation for more information.
8. (Optional) Set the **AppName** parameter to help identify Open Client database connections used by the ASE Output adapter. If not set, the default value of "ASEOutputAdapter" is used.
   See the **CS_APPNAME** connection property in the Open Client documentation for more information.

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>

 <!--  When defining a service with ODBC connectivity for Linux and
-->
 <!--  Solaris, the size of SQLLEN and SQLULEN type in the driver
manager -->
 <!--  determines the value to be    used for the DriverLibrary
parameter. -->
 <!--  For managers built with the above types being 8 bytes, -->
 <!--  esp_db_odbc64_lib should be used. -->
 <!--  If the type sizes are 4 bytes, use esp_db_odbc_lib. -->
 <!--  For example if unixODBC is the driver manager, the odbcinst --
>
 <!--  command can be used to query this information: -->
 <!--  odbcinst -j -->
 <!--  unixODBC 2.3.1 -->
 <!--  DRIVERS............: /usr/local/etc/odbcinst.ini -->
 <!--  SYSTEM DATA SOURCES: /usr/local/etc/odbc.ini -->
 <!--  FILE DATA SOURCES..: /usr/local/etc/ODBCDataSources -->
 <!--  USER DATA SOURCES..: /usr/u/user/.odbc.ini -->
 <!--  SQLULEN Size.......: 8 -->
 <!--  SQLLEN Size........: 8 -->
 <!--  SQLSETPOSIROW Size.: 8 -->

 <!--  For Windows platform, use esp_db_odbc_lib. -->
 <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>

 <!--  ONLY MEANT TO BE USED WITH ASE OUTPUT ADAPTER   -->
 <Service Name="SampleOCSService" Type="DB">
  <Parameter Name="DriverLibrary">esp_db_ocs_lib</Parameter>
  <Parameter Name="Host">localhost</Parameter>
  <Parameter Name="Port">5000</Parameter>
  <Parameter Name="User">sa</Parameter>
  <Parameter Name="Password"  />
  <Parameter Name="AppName">ASEOutputAdapter</Parameter>
 </Service>
</Services>
```

Here is an example of connecting to the SAP HANA database using a generic JDBC connection through the Database Input or Output adapter:

First, copy the SAP HANA `ngdbc.jar` file to the `$ESP_HOME/libj` directory. Then, create a service definition for the connection in the `service.xml` file.

```
<Service Name="HANAarcherJDBC" Type="DB">
  <Parameter Name="DriverLibrary">esp_db_jdbc_lib</Parameter>
  <Parameter Name="DataSource">com/sap/db/jdbcext/DataSourceSAP</
```

```
Parameter>
  <Parameter Name="Host">archer</Parameter>
  <Parameter Name="Port">30015</Parameter>
  <Parameter Name="User">SYSTEM</Parameter>
  <Parameter Name="Password">Password1</Parameter>
</Service>
```

Linking to Your ODBC Driver Manager Library

To successfully use an ODBC driver manager library on UNIX systems, link to your ODBC driver manager library and include this link in *LD_LIBRARY_PATH* for the ESP Server.

On UNIX systems, Event Stream Processor expects your ODBC driver manager library to be called `libodbc.so.1`.

1. Ensure that your driver manager library has this name or create a symbolic link from `libodbc.so.1` to your ODBC driver manager library.
2. Include the symbolic link in your *LD_LIBRARY_PATH* for the ESP Server.

See also

- *Configure Connections to External Databases* on page 58

CHAPTER 3 Administer a Cluster

Starting a Cluster

Start a cluster by starting its nodes.

Prerequisites

Set the ESP_HOME environment variable.

Task

A cluster must be running before you can deploy projects to it. To start a cluster, start manager nodes first, then controller-only nodes. Follow these steps for each node in the cluster, where <node_name> represents the name of a node in the cluster.

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

   ```
   cd %ESP_HOME%\cluster\nodes\<node_name>
   %ESP_HOME%\bin\esp_server.exe --cluster-node <node_name>.xml
   ```

   And for Linux and Solaris systems, execute:

   ```
   cd $ESP_HOME/cluster/nodes/<node_name>
   $ESP_HOME/bin/esp_server --cluster-node <node_name>.xml
   ```

   On Linux and Solaris systems, you can start a cluster in the background by executing:

   ```
   cd $ESP_HOME/cluster/nodes/<node_name>
   nohup $ESP_HOME/bin/esp_server --cluster-node <node config> 2>&1 >
   esp-node-console.out &
   ```

   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=esp[s]://<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`.

Note: If you omit the password parameter when you call the esp_cluster_admin tool, Event Stream Processor prompts you for the password and hides it as you type, which improves security.

Stopping a Cluster

Shut down the nodes in a cluster.

Prerequisites

Sybase recommends that you stop all projects running in the cluster. Stopping a node this way does not stop any projects running on the node unless the node is the only manager node.

Task

To stop a cluster, shut down the controller-only nodes first, then shut down the manager nodes. Follow these steps for each node in the cluster; <node_name> represents the name of a node.

From the Windows command line, execute:

```
cd %ESP_HOME%\cluster\nodes\<node_name> esp_cluster_admin
--uri=esp[s]://cluster_server:19011 --username=<name> --
password=<pass> --stop_node <node_name>
```

On a Linux or Solaris system, execute:

```
cd $ESP_HOME/cluster/nodes/<node_name> esp_cluster_admin
--uri=esp[s]://cluster_server:19011 --username=<name> --
password=<pass> --stop_node <node_name>
```

Note: These examples use the authentication syntax for LDAP or native OS. For RSA authentication, use this command:

```
esp_cluster_admin --uri=esp[s]://<host>:<port> --keyalias=<keyalias>
--storepass=<storepass> --keystore=<keystore> --stop_node
<node_name>
```

Note: If you omit the password parameter when you call the esp_cluster_admin tool, Event Stream Processor prompts you for the password and hides it as you type, which improves security.

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 affects your ability to connect to the local cluster 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.

Logging

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

In Event Stream Processor, projects run on local and remote clusters. Cluster-level logging is available for remote clusters only. The local cluster does not generate a cluster log file. Projects that run under the local node do generate project-level logs.

Note: Projects started in Studio run on the local cluster unless the project has an explicit connection to a remote cluster defined. For information on running projects on the local cluster, see the *Studio Users Guide*.

Event Stream Processor stores logs in flat files. You can use third-party log file analyzer tools to perform analysis on the logs.

Cluster Node Log Configuration File

The configuration file for node logging is `cluster.log.properties`, which is 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, on Windows, the node base directory is `%ESP_HOME%\cluster\projects\<cluster-name>` and on Linux/

Solaris, the node base directory is `$ESP_HOME/cluster/projects/<cluster-name>`.

If you configure a cluster during installation, by default, the cluster node log configuration file is placed in `ESP_HOME/cluster/nodes/<node-name>`. Ensure that the cluster node log configuration file is located in the same directory 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
```

In this sample configuration, the cluster node log is written to the log file `cluster.log`, which 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.

See also

- *File and Directory Infrastructure* on page 3

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 whether the project is deployed on a local or remote cluster.

The files generated by a project in the local cluster, including the project log file, `esp_server.log`, are placed in the project working directory, which defaults to `<user's-home-dir>\SybaseESP\5.1\workspace\<workspace-name>.<project-name>.<instance-number>`.

Remote cluster nodes have their own node-specific base directories. The default base directory is `<ESP_HOME>/cluster/projects/<cluster-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*.

To modify logging levels for a project at runtime, use **esp_client**:

```
esp_client -p [<host>:]<port></workspace-name/project-name> -c
<username>:<password> "loglevel <level>"
```

Log level changes made with **esp_client** do not persist—you lose your changes to the logging level if you restart the project without also changing the logging level in the `<project-name>.ccr` file. After you change the logging level in `<project-name>.ccr`, stop and remove the project from the node, then redeploy the project to activate the new logging level.

The project working directory also contains the `stdstreams.log` file. This file receives all output written to stdout and stderr. This includes SySAM licensing information for Event Stream Processor, as well as messages from third party applications that write to stdout and stderr.

See also

- *File and Directory Infrastructure* on page 3

Logging Level

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

You can set logging levels:

- In the cluster node configuration file, `<node-name>.xml`. Logging levels in `<node-name>.xml`apply to all projects that run on the node unless you set a different logging level in a project's CCR file.
- In the project configuration file, `<project-name>.ccr`.
- Using **esp_client** at runtime.

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

Cluster Administrative Tool

The cluster administrative tool is one of several options you can use for cluster administration. Use it to add and remove projects and workspaces, and to query, start, and stop existing projects.

You can perform the same tasks in Event Stream Processor Studio and in Sybase Control Center.

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

Interactive mode requires less typing. Command line mode is intended for scripting.

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

To connect to the cluster manager with RSA authentication:

```
esp_cluster_admin --uri=esp[s]://<host>:<port> --auth=rsa --
keyalias=
<keyalias> --storepass=<storepass> --keystore=<keystore>
```

To connect to the cluster manager with ticket Kerberos authentication:

```
esp_cluster_admin --uri=esp[s]://<host>:<port> --auth=krb --krb-
cache=
<krb-cache> --krb-kdc=<kdc-host> --krb-realm=<realm> --krb-
service=<service>
```

To connect to the cluster manager with LDAP or native OS (user name/password) authentication:

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

Note: If you omit the password parameter when you call the esp_cluster_admin tool, Event Stream Processor prompts you for the password and hides it as you type, which improves security.

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

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

These command line mode examples 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
```

Table 3. esp_cluster_admin 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.

Command	Function
Interactive mode: **get project <workspace-name>/<project-name>** Command line mode: **--get_projectdetail --workspace-name=<workspace-name> --project-name=<project-name>**	Returns information about the specified project, including whether it is running, on which node it is running, and runtime details. For an active-active project, the command returns information for each instance, and identifies the primary and secondary instance.
Interactive mode: **get streams <workspace-name>/<project-name>** Command line mode: **--get_streams --workspace-name=<workspace-name> --project-name=<project-name>**	Returns thestreams associated with a workspace.
Interactive mode: **get schema <workspace-name>/<project-name> <stream-name>** Command line mode: **--get_schema --workspace-name=<workspace-name> --project-name=<project-name> --stream-name=<stream-name>**	Returns the schema of the specified stream.
Interactive mode: **add workspace <workspace-name>** Command line mode: **--add_workspace --workspace-name=<workspace-name> [ignore-error]**	Adds a workspace. Use the optional **ignore-error** argument to add the workspace even when doing so causes a workspace error.
Interactive mode: **add project <workspace-name>/<project-name> <project-name>.ccx [<project-name>.ccr]** Command line mode: **--add_project --workspace-name=<workspace-name> --project-name=<project-name> --ccx=<project-name>.ccx [--ccr=<project-name>.ccr]**	Adds a project. `<project-name>.ccx` is the compiled project file. Specify the path to the file. `<project-name>.ccr` is the project's runtime configuration file. Include the CCR file for an HA (active-active) project or a project with affinities. Specify the path to the file `<project-name>.ccr` and `<project-name>.ccx` are always located in the same directory.
Interactive mode: **remove workspace <workspace-name>** Command line mode: **--remove_workspace --workspace-name=<workspace-name> [ignore-error]**	Removes a workspace. Use the optional **ignore-error** argument to remove the workspace even when doing so causes a workspace error.

Command	Function
Interactive mode: **remove project <workspace-name>/<project-name>** Command line mode: **--remove_project --workspace-name=<workspace-name> --project-name=<project-name>**	Removes a project. Prerequisite: Stop the project. You cannot remove a running project.
Interactive mode: **start project <workspace-name>/<project-name> [timeout (sec)] [<instance-index>]** Command line mode: **--start_project --workspace-name=<workspace-name> --project-name=<project-name> [--timeout=<timeout-in-seconds>] [--instance-index=<instance-index>]**	Starts the project. If the project is added with a strong controller affinity and that controller is not available, start-up fails. <timeout-in-seconds> specifies how long the call waits to verify that the project has started. For an HA (active-active) project, the instance index specifies which of the two instances to start. Valid values are 0 and 1. Use **get project**or **--get_project_detail** to determine whether and where the instances are running.
Interactive mode: **stop project <workspace-name>/<project-name> [timeout (sec)] [<instance-index>]** Command line mode: **--stop_project --workspace-name=<workspace-name> --project-name=<project-name> [--timeout=<timeout-in-seconds>] [--instance-index=<instance-index>]**	Stops the project. <timeout-in-seconds> specifies how long the call waits to verify that the project has stopped. For an HA (active-active) project, the instance index specifies which of the two instances to stop. Valid values are 0 and 1. Use **get project**or **--get_project_detail** to determine whether and where the instances are running
Interactive mode: **stop node <node-name>** Command line mode: **--stop_node --node-name=<node-name>**	Stops the node but does not stop any projects running on the node unless this node is the only manager node. A warning appears if there are active projects on the node.
Interactive mode: **encrypt <clear-text>** Command line mode: **--encrypt_text --text=<clear-text>**	Encrypts plain text data. Use this command to encrypt passwords in configuration files.

Command	Function
Interactive mode: **deploykey <new-username> <keystore> <storepass> <key-alias> [<storetype>]** Command line mode: **--deploy_key --new-user=<new-username> --keystore=<keystore> --storepass=<storepass> --key-alias=<key-alias> [--storetype=<storetype>]**	Adds a new user by deploying a new user key to the keystore. When you deploy a new user key, the node to which you send the deploy command 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.
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.
Interactive mode: **help** Command line mode: **--help**	Displays a plain-text description of the **esp_cluster_admin** utility's commands and usage information.

See also

- *Configuring a Cluster* on page 13

Safeguarding Your Data

Protect your data to improve system redundancy and prevent unauthorized access.

Sybase recommends that you:

- Secure data using OS security. Because the cluster configuration file contains keystore password information from RSA authentication, it is important that you secure this file by giving read and write access to trusted individuals or groups only.
- Take steps to secure files. Sybase recommends using disk volume encryption and storing security-related configuration on a separate disk.

- Use third-party source control to manage your project source files and provide redundancy. When source files are checked out of the source control system, use ESP Studio to browse your source folder and make changes in the source files.
- Perform regular backups of project data, including log stores.

Sharing Projects in a Git or CVS Repository

Use thirdparty source control to manage your source files.

You can manage, protect, and share your projects by adding them to a Git or CVS repository. Git and CVS are third-party source control plug-ins.

Adding a Project to a Git Repository

Add a project to a Git repository to share the project.

Prerequisites

The system administrator must provide the necessary Git elements for adding a project to the repository: Repository, Working directory, and Path within repository.

Task

1. In the **Authoring** perspective, right-click a project from the File Explorer view.
   Studio displays a pop-up menu.
2. Select **Team** > **Share Project**.
   Studio displays the **Share Project** screen.
3. Select **Git** as the repository type.
4. Click **Next**.
   Studio displays the **Enter Repository Location Information** screen.
5. Enter the following information:
   - **Repository –** Provide the name of the repository.
   - **Working directory –** Provide the directory where the project files are located.
   - **Path within repository –** Provide the path within the repository where the project files will be found.
6. Click **Finish**.
   The project is added to the Git repository.

Note: For more information on the Git source code management (SCM) system, please visit *http://www.eclipse.org/egit/documentation/*.

Adding a Project to a CVS Repository

Add a project to a CVS repository to share the project.

Prerequisites

The system administrator must have provided the necessary CVS elements for adding a project to the repository: Host, Repository path, User, and Password.

Task

1. In the **Authoring** perspective, right-click a project from the File Explorer view.
   Studio displays a pop-up menu.
2. Select **Team** > **Share Project**.
   Studio displays the **Share Project** screen.
3. Select **CVS** as the repository type.
4. Click **Next**.
   Studio displays the **Enter Repository Location Information** screen.
5. Enter the following information:
   - **Host –** Provide the name of the server on which the CVS repository resides.
   - **Repository path –** Provide the full path to the repository.
   - **User –** Provide the ID of a user with permission to enter the repository.
   - **Password –** Provide that user's password.
6. For the connection type, select **pserver** from the drop down menu.
7. Select **Use default port**.
8. Click **Finish**.
   The project is added to the CVS repository.

Note: For more information on the CVS source code management (SCM) system, please visit *http://help.eclipse.org/indigo/index.jsp?topic=%2Forg.eclipse.platform.doc.user%2FgettingStarted%2Fqs-60_team.htm*.

Data Backup and Restoration

Back up project data with the project running or stopped. View backup files and restore the data to your system.

Performing backups while the project is offline (stopped) is preferred. You can, however, perform a backup while the project is running.

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

You can restore data from backup files for Linux, Solaris, and Windows systems.

Data Backup

Manage and protect data by performing a backup.

On Linux and Solaris systems, back up:

- project files ending in `.ccl`, `.ccr`, and `.ccx` in the workspace folder, `/SybaseESP/5.1/workspace` unless you overrode the default when installing ESP
- `service.xml` file in *`$ESP_HOME`*`/bin`
- node configuration files in *`$ESP_HOME`*`/cluster/nodes/node1`, *`$ESP_HOME`*`/cluster/nodes/node2`, ...
- `cluster.log.properties` files in *`$ESP_HOME`*`/cluster/nodes/node1`, *`$ESP_HOME`*`/cluster/nodes/node2`, ...
- security configuration files in *`$ESP_HOME`*`/security`
- log store files in the folder you specified when you created the log stores
- any external files used by your projects

On Windows systems, back up:

- project files ending in `.ccl`, `.ccr`, and `.ccx` in the workspace folder, `C:\Users\`*`userid`*`\My Documents\SybaseESP\5.1\workspace` unless you overrode the default when installing ESP
- `service.xml` file in *`%ESP_HOME%`*`\bin`
- node configuration files in *`%ESP_HOME%`*`\cluster\nodes\node1`, *`%ESP_HOME%`*`\cluster\nodes\node2`, ...
- `cluster.log.properties` files in *`%ESP_HOME%`*`\cluster\nodes\node1`, *`%ESP_HOME%`*`\cluster\nodes\node2`, ... if you have modified them
- security configuration files in *`%ESP_HOME%`*`\security` if you have modified them
- log store files in the folder you specified when you created the log stores
- `ODBC.INI` file in `C:\Windows` if you are using the ODBC driver for ESP
- any external files used by your projects

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.

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-number>]
   ```

   For *port*, enter the port number used by your Event Stream Processor installation. For *espuser* and *password*, enter your user credentials.

   Note: If you omit the password parameter when you call the esp_cluster_admin tool, Event Stream Processor prompts you for the password and hides it as you type, which improves security.

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.

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.

Viewing Backup Files

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

To view your backup files, use the **tar -tvf backup.tar** command for Linux and Solaris, and use the **pkunzip -v backup.zip** command for Windows.

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.

On Linux and Solaris systems, the **tar -xvf** command puts the backup files in their original directories:

- project files ending in `.ccl`, `.ccr`, and `.ccx` in the workspace folder, `/SybaseESP/5.1/workspace` unless you overrode the default when installing ESP
- `service.xml` file in *`$ESP_HOME`*`/bin`
- node configuration files in *`$ESP_HOME`*`/cluster/nodes/node1`, *`$ESP_HOME`*`/cluster/nodes/node2`, ...
- `cluster.log.properties` files in *`$ESP_HOME`*`/cluster/nodes/node1`, *`$ESP_HOME`*`/cluster/nodes/node2`, ...
- security configuration files in *`$ESP_HOME`*`/security`
- log store files in the folder you specified when you created the log stores
- any external files used by your projects

On Windows systems, the **WinZip** or **pkunzip** command puts the backup files in their original folders:

- project files ending in `.ccl`, `.ccr`, and `.ccx` in the workspace folder, `C:\Users\`*`userid`*`\My Documents\SybaseESP\5.1\workspace` unless you overrode the default when installing ESP
- `service.xml` file in *`%ESP_HOME%`*`\bin`
- node configuration files in *`%ESP_HOME%`*`\cluster\nodes\node1`, *`%ESP_HOME%`*`\cluster\nodes\node2`, ...
- `cluster.log.properties` files in *`%ESP_HOME%`*`\cluster\nodes\node1`, *`%ESP_HOME%`*`\cluster\nodes\node2`, ... if you have modified them
- security configuration files in *`%ESP_HOME%`*`\security` if you have modified them
- log store files in the folder you specified when you created the log stores
- `ODBC.INI` file in `C:\Windows` if you are using the ODBC driver for ESP
- any external files used by your projects

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.

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.

Creating a Log Store

If failover is enabled, configure a log store to capture the data that flows through a project.

Note: Log stores do not store Sybase Event Stream Processor event logs (cluster logs, server logs, or project logs).

Create one log store per project. The preferred destination for log store files is the base directory where project files are stored.

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:

Relative path (preferred)	A relative path is relative to the ESP 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 (/).
Absolute path (not recommended)	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 must be a shared disk accessible by all cluster nodes. The log store path 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>`. You can view base directory definitions in the cluster configuration file (`<node-name>.xml`), under the controller section.

   Sybase recommends that you use a relative path. 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, configure a strong affinityto ensure the project always runs 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**.

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 in which all the streams are directly or indirectly derived from it, 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:

$96 + 32 * \text{ceiling}(\log_2(\text{number\_of\_records\_in\_the\_stream}))$

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

$96 + 32 * \text{ceiling}(\log_2(1000)) = 96 + 32 * 10 = 416$

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 fills 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.

Sizing a Log Store

Calculate the size of the log store your project requires. 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 in Studio or the **esp_playback** utility to record and play back real data to get a better idea of the amount of data you need to store. (See the *Utilities Guide* for details on **esp_playback**.)

   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 value 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 it consumes 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. You must add up the data from all the input paths. 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 down to all the streams in the log store. Add 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.

Memory Usage

To maximize performance, the project server ensures that only one copy of a record can exist. If necessary, you can adjust the memory available to a project's Java virtual machine.

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.

In addition, each ESP project launches a Java virtual machine (JVM), which runs any Java UDFs or Java internal adapters associated with the project. Memory available to the JVM is controlled by the **java-max-heap** option in the Deployment section of the project configuration (CCR) file; the default value is 256 MB.

If your project triggers Java out of memory errors, increase the heap size for the project's JVM. For example:

```
<Option name="java-max-heap" value="512"/>
```

CHAPTER 4 Monitor a Cluster

You can monitor clusters using command line tools, metadata streams, or Sybase Control Center, a management application with a graphical user interface.

Monitoring a Project

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 RPC port 31415, use the following command:

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

For more information on the **esp_monitor** tool, see the *Utilities Guide*.

Monitoring with Sybase Control Center

Sybase Control Center for Sybase Event Stream Processor is a Web-based tool for managing and monitoring ESP Server nodes, clusters, projects, and other components of the Event Stream Processor environment.

The SCC architecture allows multiple administrators using Web clients to monitor and control all the Event Stream Processor components in an enterprise through one or more SCC servers. SCC for Event Stream Processor provides availability monitoring, historical performance

monitoring, and administration capabilities in a scalable Web application that is integrated with management modules for other Sybase products. It offers shared, consolidated management of heterogeneous resources from any location, alerts that provide state- and threshold-based notifications about availability and performance in real time, and intelligent tools for spotting performance and usage trends, all via a thin-client, rich Internet application (RIA) delivered through your Web browser.

Use SCC for Event Stream Processor to track a variety of performance metrics, gathering statistics that over time will give you powerful insight into patterns of use. You can display collected data as tables or graphs. By plotting results over any period of time you choose, from a minute to a year, you can both see the big picture and focus on the particulars. Detailed knowledge of how your Event Stream Processor environment has performed in the past helps you ensure that Event Stream Processor meets your needs in the future.

You can install Sybase Control Center using the Event Stream Processor installer. To read about SCC, go to *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.

Monitoring with 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 changes 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. For example, you can use the **esp_subscribe** utility. This command subscribes to the streams _ESP_Connectors and _ESP_Streams from the project default/prj1, which is running on a cluster manager on localhost:11180, and prints all stream data in XML format on standard output.

```
esp_subscribe -c user-id:password -s _ESP_Connectors,_ESP_Streams -p
localhost:11180/default/prj1
```

For details on **esp_subscribe**, see the *Utilities Guide*.

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_Adapter_Statistics

Reports statistics unique to each adapter. Both internal and external adapters can publish statistics to this stream.

For information on the statistics that each adapter publishes, see the *Adapters Guide*.

Column	Type	Description
adapter_name	`string`	A unique name of the adapter instance.
stat_name	`string`	The name of an adapter statistic, as defined by the adapter.
last_update	`bigdate-time`	The time that the statistic was last updated.
value	`string`	The value of the statistic (converted to a string).

_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.
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 and a copy of data from the _ESP_Clients stream. Monitoring data is available only if the **time-granularity** option in the project configuration (CCR) file is set to greater than 0. The

frequency of updates corresponds to the value of the **time-granularity** option. For example, if set to 1, an update is published every second, if set to 30, an update is published every 30 seconds, and if set to 0, reporting is disabled.

Column	Type	Description
Handle	`long`	A unique integer ID of the connection.
user_name	`string`	The user name provided by the client during connection establishment. Shown once the user is authenticated.
IP	`string`	The IP 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	`timestamp`	The time the server accepts (but does not authenticate) the connection.
conn_tag	`string`	The user-set symbolic connection tag name. If not set by the user, conn_tag is NULL.
cpu_pct	`float`	Total CPU usage for the client thread, as a percentage of a single CPU core.
last_update	`date`	The time of the current update.
subscribed	`integer`	The status of a subscription to a stream. 1 if subscribed, 0 if not subscribed.
sub_trans_per_sec	`float`	The client's performance, in transactions per second, received by the client since the last update.
sub_rows_per_sec	`float`	The client's performance, in data rows per second, received by the client since the last update.
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.

Column	Type	Description
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.
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.

Column	Type	Description
node_cpu_pct	`float`	Total CPU usage for the client, as a percentage of all CPU cores on the machine. Total CPU usage equals system CPU usage plus user CPU usage.
node_cpu_pct_system	`float`	System CPU usage for the client, as a percentage of all CPU cores on the machine.
node_cpu_pct_user	`float`	User CPU usage for the client, as a percentage of all CPU cores on the machine.
cpu_time	`interval`	Total CPU time since the creation of the client, in microseconds. Total CPU time equals system CPU time plus user CPU time.
cpu_time_system	`interval`	Total system CPU time, in microseconds, since the creation of the client thread.
cpu_time_user	`interval`	Total user CPU time, in microseconds, since the creation of the client thread.
time_since_start	`interval`	Duration of lapsed real time since the creation of the client thread.

_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.
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 internal 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.

Column	Type	Description
input	`integer`	Values are 1 for InConnection or 0 for OutConnection.
ingroup	`string`	The StartUp group where this connector belongs.
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_Project_Monitor

Contains information on project CPU usage, memory consumption, and number of threads. Monitoring data is available only if the **time-granularity** option in the project configuration (CCR) file is set to greater than 0. The frequency of updates corresponds to the value of the **time-granularity** option. For example, if set to 1, an update is published every second, if set to 30, an update is published every 30 seconds, and if set to 0, reporting is disabled.

Column	Type	Description
project_name	`string`	Currently hard-coded to the word "project".
node_cpu_pct	`float`	Total CPU usage, as a percentage, by the project since the last update. Total CPU usage equals the system CPU usage plus the user CPU usage. Valid values are in the range from 0.0 to 100.00%. On multi-core machines, the percentage is relative to the total number of available cores. A value of 100% indicates a usage of 100% of all cores on the machine.

Column	Type	Description
node_cpu_pct_system	`float`	The system (Kernel on Windows) CPU usage, as a percentage, by the project since the last update. Valid values are in the range from 0.0 to 100.00%. On multi-core machines, the percentage is relative to the total number of available cores. A value of 100% indicates a usage of 100% of all cores on the machine.
node_cpu_pct_user	`float`	The user CPU usage, as a percentage, by the project since the last update. Valid values are in the range from 0.0 to 100.00%. On multi-core machines, the percentage is relative to the total number of available cores. A value of 100% indicates a usage of 100% of all cores on the machine.
cpu_time	`interval`	Total CPU time for the project, in microseconds. Total CPU time is equal to the system CPU time plus the user CPU time.
cpu_time_system	`interval`	Total system CPU time for the project, in microseconds.
cpu_time_user	`interval`	Total user CPU time for the project, in microseconds.
time_since_start	`interval`	Duration of lapsed real time since the project was started, in microseconds.
startmem_usage_vm	`long`	Total amount of virtual memory, in bytes, used by the project at the time of the update.
mem_usage_rss	`long`	Total amount of system memory (RSS), in bytes, used by the project at the time of the update.
num_threads	`integer`	Total number of threads used by the project at the time of the update.
last_update	`bigdatetime`	Time of the current update.

_ESP_RunUpdates

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

Column	Type	Description
key	`string`	The type of the update. See table below.

Column	Type	Description
value	`integer`	A number associated with the update, determined by the Key column.
stream	`string`	The name of the stream if the update notifies an event related to an individual stream; otherwise, stream NULL.
info	`string`	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".
REQUESTEXIT	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".
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, and a copy of data from the _ESP_Streams stream. Monitoring data is available only if the **time-granularity** option in the project configuration (CCR) file is set to greater than 0. The frequency of updates corresponds to the value of the **time-granularity** option. For example, if set to 1, an update is published every second, if set to 30, an update is published every 30 seconds, and if set to 0, reporting is disabled.

Column	Type	Description
stream	`string`	The name of the stream.
target	`string`	The name of the target element.
cpu_pct	`float`	Total CPU usage for the stream thread, as a percentage of a single CPU core.
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.

Column	Type	Description
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.
node_cpu_pct	`float`	Total CPU usage for the stream, as a percentage of all CPU cores on the machine. Total CPU usage equals system CPU usage plus user CPU usage.
node_cpu_pct_system	`float`	System CPU usage for the stream, as a percentage of all CPU cores on the machine.
node_cpu_pct_user	`float`	User CPU usage for the stream, as a percentage of all CPU cores on the machine.
cpu_time	`interval`	Total CPU time since the creation of the stream, in microseconds. Total CPU time equals system CPU time plus user CPU time.
cpu_time_system	`interval`	Total system CPU time, in microseconds, since the creation of the stream.
cpu_time_user	`interval`	Total user CPU time, in microseconds, since the creation of the stream.
time_since_start	`interval`	Duration of lapsed real time since the project was started, in microseconds.

_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.
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.

Column	Type	Description
user name	`string`	Hard-coded as "user".
remarks	`string`	The stream's numeric ID, a decimal number as an ASCII string.

Index

*any resource 31
*any role 31

A

access control 30
access control management 30
    *any resource 31
    *any role 31
    access methods 30
    actions 31, 34
    available actions 30
    enabling access control 34
    enabling or disabling access control 38
    resources 31, 34
    role hierarchy 30
    roles 31, 34
    sample roles 37
actions in access control 31
active-active projects 10, 51
affinities 53
ASE
    configuring a JDBC connection 59
authentication
    configuring 29
    configuring for UNIX 27
    Kerberos authentication 19, 20
    LDAP authentication 19, 25
    multiple methods 29
    RSA authentication 19, 22
    server (remote cluster) vs. local cluster authentication 19
Authentication 25
authentication through local ESP accounts 28
authorization 30
    actions 31, 34
    configuration 38
    policy file 34
    resources 31, 34
    roles 31, 34
    sample roles 37

B

backing up data 79
    offline backup 80
    online backup 80
    viewing backup files 81
base directory
    location 3

C

caches 11
CCL file 3
CCR file 3
    example 54
CCX file 3
cluster manager
    Kerberos authentication 19
    LDAP authentication 19
    RSA authentication 19, 22
    security 19
cluster node configuration file 3
cluster node logging
    log4j configuration file 69
cluster persistence 11
    directory, setting 13
    enabling and disabling 13
cluster.log file 3
cluster.log.properties file 3
clustering
    adding nodes 12
    administrative tool 72
    affinity types and options 53
    architecture 1
    best practices 9
    cache 11
    configuration 13
    connection to a server from ESP Studio 68
    controllers 1
    deploying projects 48
    esp_cluster_admin utility 72
    file and directory locations 3
    listing controllers 72
    listing managers 72
    listing streams 72
    local and remote clusters 1
    log stores 82
    managers 1
    managing projects 72
    managing workspaces 72

nodes 1
persistence 11
planning 9
restrictions on single-node clusters in Windows 1
starting a cluster 67
stopping a cluster or a node 68
controllers 1
determining how many you need 9
csi_local.xml authentication file 28
configuring access control 34
location 3
csi_role_mapping.xml file 34
CVS 77, 78

D

data
backing up 79
offline backup 80
online backup 80
restoration 81
restoring backup files on Windows systems 82
restoring files on Linux and Solaris systems 82
viewing backup files 81
database access
configuring JDBC connections 59
DB2
configuring a JDBC connection 59
debugging
logging levels 71
decryption
external adapter decryption script 45
directory locations
base 3
ESP_HOME 3
examples 3
persistence 3
project working directory 3
security 3
workspace 3
driver manager library
ODBC 65
drivers
JDBC 57
OCS 57
ODBC 57

E

encryption
encrypting the RTView adapter password 47
esp_cluster_admin utility 44
external adapter encryption script 45
files containing encrypted passwords 43
internal adapter configuration files 44
Java external adapter configuration files 45
project configuration files 44
service configuration file 44
ESP 20, 25
esp_cluster_admin utility 72
starting the utility 43
ESP_HOME directory
location 3
esp_monitor 89
esp_server.log 3
esp_subscribe utility 90
examples directory
location 3
external database access
configure connections 58
configuring OCS connections 62
configuring ODBC connections 61
JDBC drivers 57
OCS driver 57
ODBC drivers 57

F

failover 54
file locations
CCL 3
CCR 3
CCX 3
cluster log configuration (cluster.log.properties) 3
cluster node configuration (node-name.xml) 3
csi_local.xml 3
input files for projects 3
logs 3
project configuration (project-name.ccr) 3
project log (esp_server.log) 3
service.xml 3
files
csi_local.xml 34
csi_role_mapping.xml 34
node-name.xml, configuring with 13
policy.xml 34

service configuration, sample 63
with encrypted passwords 43

G

Git 77

H

HA (high availability) projects 10, 51

I

IBM DB2
configuring a JDBC connection 59
instances 52

J

Java heap size, changing 88
Java keytool 23
generating a self-signed private key 40
pem format private key 42
JDBC drivers 57
JRE, configuring alternate to default 13

K

Kerberos authentication 19, 20
configuration 20
Kx Systems KDB+
configuring a JDBC connection 59

L

LDAP authentication 19
configuration 25
local clusters 1
lockout
recovery 39
log stores
creating 82
location 3
managing 83
sizing 83, 86
logging
cluster log configuration file 3
project log 69
server log 69

M

managers 1
determining how many you need 9
map roles for access control 34
memory
usage 88
metadata streams 90
_ESP_Adapter_Statistics 91
_ESP_Clients 91
_ESP_Clients_Monitor 91
_ESP_Clockupdates 94
_ESP_Columns 95
_ESP_Config 95
_ESP_Connectors 95
_ESP_Keycolumns 97
_ESP_Project_Monitor 97
_ESP_RunUpdates 98
_ESP_Streams 100
_ESP_Streams_Monitor 100
_ESP_Streams_Topology 101
_ESP_Subscriptions 102
_ESP_Subscriptions_Ext 102
_ESP_Tables 102
Microsoft SQL Server
configuring a JDBC connection 59
monitoring
projects 89
with esp_monitor 89
with metadata streams 90
with Sybase Control Center 89
multicast 11
enabling and disabling 13

N

native OS authentication 26
node-name.xml file 3
using to configure a cluster 13
nodes 1
adding to clusters 12
determining how many you need 9
starting 67
stopping 68

O

OCS driver 57
ODBC connections to databases 61
ODBC driver manager library
    linking 65
    symbolic link 65
    UNIX 65
ODBC drivers 57
options for project deployment 48
Oracle
    configuring a JDBC connection 59
out of memory errors, Java 88

P

password encryption
    See encryption
passwords
    prompting for at node start-up 13
persistence directory
    location 3
pluggable authentication modules for UNIX
        authentication 27
policy.xml file
    configuring access control 34
    sample access control policies 37
preconfigured username-password option 28
privileges
    admin, recovering 39
project configuration
    CCR file 3
    project deployment options 48
    sample file 54
project logging
    debug logging levels 71
    project working directory 71
project servers 1
project-name.ccl 3
project-name.ccr 3
project-name.ccx 3
projects
    deploying 48
    log file location 3
    memory usage 88
    multiple project support 1
    project working directory location 3
    workspace location 3

R

remote clusters 1
resources in access control 31
restoring data 81
    Linux and Solaris systems 82
    Windows systems 82
role mapping for access control 34
roles
    authorization 37
    in access control 31
    sample 37
RSA authentication 19, 22
    configuration 22
    key generation 23
    RSA key 23
RSA key generation
    Java keytool 23
RTView adapter
    encrypting the password 47

S

sample project configuration file 54
sample service configuration file 63
SAP HANA
    configuring a JDBC connection 59
SCC for ESP 89
security 18, 26
    access control management 30, 31, 34, 38
    authorization 30, 31, 34, 38
    data 76
    Kerberos authentication 19, 20
    LDAP authentication 19, 25
    personal data 76
    protecting your data 76
    RSA authentication 19, 22
security directory
    location 3
service configuration file 3, 63
    sample 63
service.xml file 3
shared drive requirements for ESP files and
        directories 3
sizing
    log stores 83
sizing log stores 86
source control 77
Source Control 77, 78
SQL Server
    configuring a JDBC connection 59
SSL
    configuration 40

stores
    log stores 82
Sybase Control Center for Sybase Event Stream Processor 89

T

third-party plug-ins 77

U

UNIX
    configuring authentication 26, 27
    locations of ESP files and directories 3
unlocking the server 39
User/password 25
users
    configuring authentication for 19
    creating by deploying user keys 76

V

viewing backup files 81

W

wildcard for access control resources
    See *any resource
wildcard for access control roles
    See *any role
Windows
    configuring authentication 26
    locations of ESP files and directories 3
    restrictions on single-node clusters 1
workspace
    location 3