Configuration and Administration Guide

SAP Sybase Event Stream Processor 5.1 SP04

Contents

CHAPTER 1 Clusters

An SAP® 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 SAP Sybase Event Stream Processor 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 SAP Sybase Event Stream Processor 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.

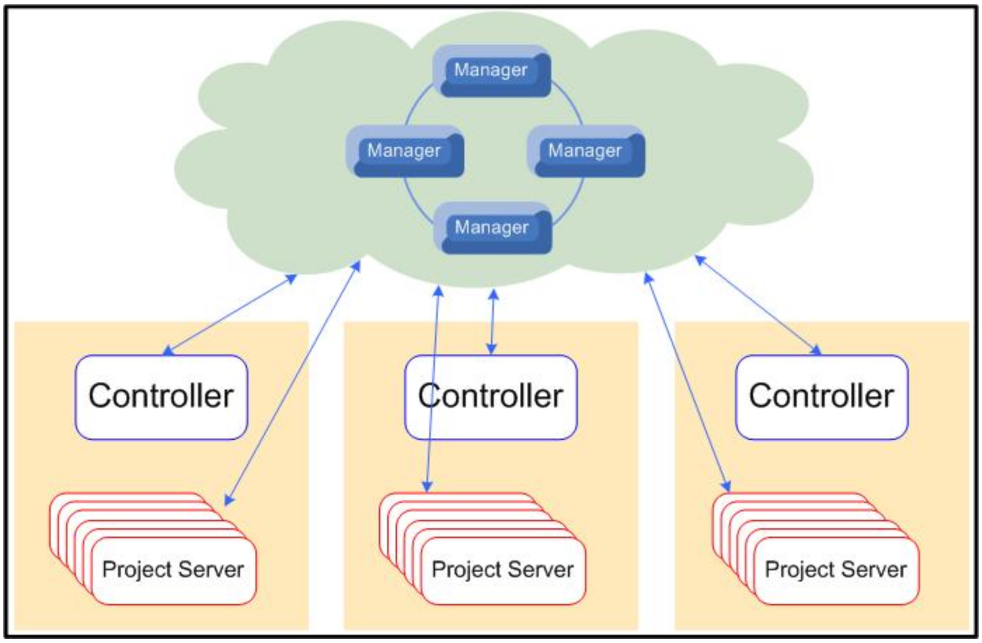
Nodes are processes that run on hosts. There are two functional types of nodes: managers and controllers. A manager is an external access point to a cluster. Managers maintain and monitor cluster and application state. A controller is responsible for starting and monitoring the processes that run projects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

Clusters can include manager-only nodes, controller-only nodes, and 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, where the one node must serve both roles.)

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

Figure 1: Cluster Architecture



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 has 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, expand your environment to include additional clusters.

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.

Every deployed project 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 has 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 which the 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.

See also

- *A Project Fails Repeatedly* on page 127

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 SAP 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 10 and *project working directory* on page 9. • **Needs shared drive? –** Yes • **Default location –** Remote cluster: `ESP_SHARED/cluster/projects/<cluster-name>` The `<base-directory>` is defined in the controller section of the cluster node configuration file, `ESP_HOME/cluster/nodes/<node-name>/<node-name>.xml`. 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`. `The <base-directory>` for Studio is defined in Studio's cluster configuration file, `<ESP_HOME>/studio/clustercfg/localnode.xml`.
`cluster.key`	The cluster key file, which is used to encrypt SSL files and passwords in `<node-name>.xml`. • **Needs shared drive? –** For security purposes, SAP does not recommend storing `cluster.key` on a shared drive. SAP recommends permissions settings that allow only the user who launches the node to access this file. • **Default location –** `ESP_HOME/cluster/keys/<cluster-name>`

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 *node working directory* on page 7, 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 (the *node working directory* on page 7). • **Needs shared drive? –** No • **Default location –** `ESP_HOME/cluster/nodes/<node-name>/cluster.log.properties`

`csi_boe.xml`
`csi_kerberos.xml`
`csi_ldap.xml`
`csi_native_nt.xml`
`csi_native_unix.xml`
`csi_role_mapping.xml`
`csi_rsa.xml`
	Security configuration files. See also *security directory* on page 9 and *examples directory* on page 6.
	• **Needs shared drive? –** Optional; typically shared in a multinode cluster where the security directory is shared
	• **Default location –** `ESP_SHARED/security`

`ESP_HOME`	Represents the Event Stream Processor installation directory. **ESP_HOME** is also an environment variable and a configuration file macro. See the *Configuring a Cluster* on page 17 topic for information on defining macros. • **Needs shared drive? –** Must not be shared • **Default location –** Windows: `C:\Sybase\ESP-5_1` UNIX: `/opt/sybase/ESP-5_1`

Name	Description
`esp_metadata`	Metadata directory that holds the metadata log store for the project if you are using the guaranteed delivery feature. • **Needs shared drive? –** Yes • **Default location –** `<base-directory>/<workspace-name>.<project-name>.<instance-number>/esp_metadata`
`ESP_SHARED`	A configuration file macro for resources that need to be on a shared drive. After moving these resources, set the macro path to the shared location. See the *Configuring a Cluster* on page 17 topic for information on defining macros. • **Needs shared drive? –** Yes • **Default location –** `ESP_HOME`
`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>`

Name	Description
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`
Node working directory	Directory from which the node is started. `cluster.log.properties` must reside in this directory; `cluster.log` and `<node-name>.xml` reside here by default. Not to be confused with *workspace* on page 10 or *project working directory* on page 9. • **Needs shared drive? –** No • **Default location –** `ESP_HOME/cluster/nodes/<node-name>`
`<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. • **Needs shared drive? –** Yes • **Default location –** `ESP_SHARED/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 (SAP recommends a shared drive for the security directory) • **Default location –** `ESP_SHARED/security/policy.xml`

Name	Description
`<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`
`<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`

Name	Description
Project working directory	Contains files generated by the project. May also contain input files the project needs. Not to be confused with *node working directory* on page 7 or *workspace* on page 10. • **Needs shared drive? –** Optional; typically shared in a multinode cluster where the base directory is shared. **Note:** Put the project working directory on a shared drive if you are using guaranteed delivery, unless you elected to locate the metadata log store elsewhere. Metadata log stores for guaranteed delivery are stored by default in `esp_metadata` in the project working directory. Metadata log stores require a shared drive. • **Default location –** `<base-directory>/<workspace-name>.<project-name>.<instance-number>`
`security`	Security directory. Stores Java keystores for RSA, 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_SHARED/security`
`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`
`server.key` `server.crt`	Unencrypted SSL files. The ESP installer lays down only the encrypted (.enc) versions of the SSL files; you need to provide unencrypted versions only if you disable SSL encryption. • **Needs shared drive? –** For security purposes, SAP does not recommend storing unencrypted SSL files on a shared drive. • **Default location –** `ESP_HOME/cluster/keys/<cluster-name>`

Name	Description
`server.key.enc` `server.crt.enc`	Encrypted SSL files. ESP requires a set of encrypted SSL files (by default) or a set of unencrypted SSL files, but not both. • **Needs shared drive? –** No • **Default location –** `ESP_HOME/cluster/keys/<cluster-name>`
`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`
`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 4, *node working directory* on page 7, and *project working directory* on page 9. • **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`.

See also

- *Configuring a Cluster* on page 17
- *High Availability* on page 12
- *Centralized Security* on page 15
- *Configuring Security* on page 22
- *Deploying a Project to a Cluster* on page 63

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. *Configuring 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. *Configuring 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, SAP 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, SAP 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, SAP 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 16

High Availability

Event Stream Processor supports a set of high availability features that promote failure recovery and data redundancy.

Table 2. High Availability Features

Feature		Where Set	Description
Clusters		Cluster configuration	Though a single-node cluster provides project-level failure recovery, it 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 restart on other servers if their affinities allow it. (Affinities control which server or servers a project can run on.)

Feature		Where Set	Description
Cold Failover		Project configuration	In cold failover, an ESP node detects when a project stops running unexpectedly and, if the configuration allows, restarts the project on the same node or a different node.
Active-Active Mode		Project configuration	When you deploy a project in active-active HA mode, two instances of the same project run in the cluster, preferably on separate machines. One version of the project is designated the primary instance, and the other is designated the secondary instance. All connections from outside the cluster (adapters, clients, Studio) are directed to the primary project server. If the primary instance fails, a hot failover occurs and 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.
Zero Data Loss			Using the three zero data loss features—guaranteed delivery, consistent recovery, and auto checkpoint—you can protect a project against data loss in the event of a server crash or loss of connection.
	Guaranteed delivery	• Window properties in Studio • Adapter CNXML files • Client applications (via SDKs) • Binding parameters in CCR files	Guaranteed delivery (GD) uses log stores to ensure that a GD subscriber registered with a GD window receives all the data processed by that window even if the client is not connected when the data is produced. GD is supported only on windows (not on streams or delta streams) and each GD window requires a log store.

Feature		Where Set	Description
	Consistent recovery	Project configuration	The consistent recovery feature can restore all the windows in a project to a consistent state after a server or connection failure. (Recovery consistency depends on following guidelines for log stores.) When consistent recovery is enabled, the server uses coordinated checkpoints to save data in log stores. When any log store fails to complete a checkpoint, all the log stores for that project roll back to their state as of the previous successful checkpoint. This rule ensures that even if a server or connection fails, all log stores in a project are consistent with one another. However, any input data that has not been checkpointed is not recovered upon restart.
	Auto checkpoint	Project configuration	Auto checkpoint lets you control how often log store checkpoints occur across all input streams and windows in the project. More frequent checkpoints mean less data is lost if the server crashes. At the maximum checkpoint frequency of every input transaction (value of 1), all input data is protected except the data from the last transaction, which might not be checkpointed before a crash. When you set checkpoint frequency, you make a trade-off: with frequent checkpoints you can reduce the amount of data at risk, but performance and latency may suffer as a result. The alternative is to increase performance but risk a larger amount of data loss by setting infrequent checkpoints.
Persistent subscribe pattern (PSP)		Shape context menu in Studio	Persistent subscribe pattern is an early HA feature similar to guaranteed delivery. SAP recommends that you use guaranteed delivery if possible. However, you might prefer PSP if: • You need to guarantee delivery of data from a stream, from a delta stream, or from a window assigned to a memory store. • You do not want the guaranteed delivery store to be a log store for performance reasons. Using a memory store allows recovery when the client restarts, but not when the project restarts.

See also

- *Cluster Persistence, Caching, and Multicast* on page 15
- *Centralized Security* on page 15
- *Active-Active Deployments* on page 68
- *File and Directory Infrastructure* on page 3
- *Deploying a Project to a Cluster* on page 63

Cluster Persistence, Caching, and Multicast

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

Persistence is enabled for clusters by default. Persistence saves 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, SAP recommends that you leave cluster persistence enabled.

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 12
- *Centralized Security* on page 15
- *Configuring a Cluster* on page 17

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, SAP BI, 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 12
- *Cluster Persistence, Caching, and Multicast* on page 15
- *File and Directory Infrastructure* on page 3
- *Deploying a Project to a Cluster* on page 63

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 SAP 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 11
- *Starting a Node or Cluster* on page 85

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 ssl="true">19011</Port>
    <AdminHost>dino</AdminHost>
    <AdminPort ssl="true">19111</AdminPort>
  </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 91 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}/cluster/keys/test-
name-1</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 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.

```
<Port ssl="true">19011</Port>
```

10. (Optional) Provide a separate Admin host name and Admin port value. Doing so allows you to distinguish between administrative and non-administrative users, and limit network access to specific administrative actions, which may be advantageous when you have firewalls in place.

```
<AdminHost>dino</AdminHost>
<AdminPort ssl="true">19111</AdminPort>
```

11. 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>
```

12. (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 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.

13. 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>
```

14. (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.

15. (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>
```

16. For manager nodes, enable or disable cluster persistence.

    By default, persistence is enabled. To disable it, set `<Persistence enabled="false">`. Cluster persistence lets the node 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, SAP recommends that you leave persistence enabled.

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

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

17. Move ESP files and directories that require a shared drive to a shared location so that other nodes in the cluster can access them. Set the path of the `ESP_SHARED` macro to this location.

    Note: See *File and Directory Infrastructure* on page 3 for sharing requirements.

18. 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 Persistence, Caching, and Multicast* on page 15
- *Cluster Administrative Tool* on page 92
- *File and Directory Infrastructure* on page 3
- *Enabling and Disabling SSL* on page 98

Configuring 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 63
- *File and Directory Infrastructure* on page 3

Authentication

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

Note: Linux provides finger print authentication, which ESPdoes not support. If this is enabled in the host and ESP is using native operating system authentication, ESP may shut down unexpectedly while authenticating users.

The type of server authentication you use is selected 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
  - SAP BI 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 SAP Sybase Event Stream Processor 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 SAP Sybase Event Stream Processor 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 SAP Sybase Event Stream Processor 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 SAP Sybase Event Stream Processor 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 37
- *Secure Sockets Layer (SSL) Connections* on page 50
- *Generating the Java Keystore* on page 51
- *Generating PEM Format Private Keys* on page 53
- *Encryption* on page 53

Configuring Kerberos Authentication

SAP 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 name takes the form `<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/nodes/<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 3. 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`

ESP_SERVICE_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 username.

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 username 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 username 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 name takes the form `<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 SAP 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/nodes/<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 the alias 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 28 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 28 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 53

User-Name-Password Authentication

Event Stream Processor provides several options for implementing user-Name-password authentication: LDAP, SAP BI, the operating system's native credential management system (native OS), and a native ESP preconfigured login 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 name takes the form `<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/nodes/<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 and all of the cluster managers.

Configuring SAP BI Authentication

To configure the server for SAP BI authentication, edit the cluster configuration file and the `csi_boe.xml` file.

During installation Event Stream Processor creates a cluster configuration file, called `node1.xml` by default. 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, the name of your cluster configuration file takes the form `<node-name>.xml`.

Event Stream Processor provides a default `csi_boe.xml` file in the `ESP_HOME/security` directory that you can use as a template and modify based on your specific SAP BI implementation.

If you selected SAP BI authentication at installation time, the installation program modified the `<node-name>.xml` file to refer to `csi_boe.xml`, and modified `csi_boe.xml` with the configuration information for SAP BI authentication. If you installed with a different authentication type, perform these steps to enable and configure SAP BI authentication:

1. Use a text editor to open the cluster configuration file, `ESP_HOME/cluster/nodes/<node-name>/<node-name>.xml`.
2. In the Security section, specify `csi_boe.xml` as the CSI configuration file. For example:

```
    <Security>
        <Csi>
            <!-- The File node is macro expanded by default. -->
            <!-- To disable expansion, set attribute expand="false". -->
          <!-- This node specifies the CSI configuration file to use. -->
          <!-- There are currently 5 distributed CSI config examples. -->
            <!-- csi_native_nt.xml uses a CSI supplied LoginModule that 
provides native OS authentication on Windows. -->
            <!-- csi_native_unix.xml uses a CSI supplied LoginModule that 
provides native OS authentication on Unix. -->
          <!-- csi_ldap.xml uses a CSI supplied LoginModule that provides 
LDAP authentication. -->
            <!-- csi_kerberos.xml uses an ESP supplied LoginModule that 
provides Kerberos authentication. -->
          <!-- csi_rsa.xml uses an ESP supplied LoginModule that provides 
RSA authentication. -->
          <!-- csi_boe.xml uses an ESP supplied LoginModule that provides 
SAP BI authentication. -->
            <File>${ESP_HOME}/security/csi_boe.xml</File>
            <!--Policy>${ESP_HOME}/security/policy.xml</Policy-->
        </Csi>
```

3. Save and close the cluster configuration file.
4. Go to the `ESP_HOME/security` folder and open the SAP BI configuration file, `csi_boe.xml` in a text editor.

```
<?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.esp.cluster.security.BoeLoginModule">
                <config:options name="cmsUri"
value="SAP_HOST_PLACEHOLDER:SAP_PORT_PLACEHOLDER"/>
                <config:options name="authenticationMethod"
value="secEnterprise"/>
        </config:authenticationProvider>
       <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:configuration>
```

5. Change `SAP_HOST_PLACEHOLDER` to the name of the host where you are running SAP BI.
6. Change `SAP_PORT_PLACEHOLDER` to the port to use when communicating with that host. Be careful not to remove the colon separating the host name and the port number.
7. Save and close the `csi_boe.xml` file.
8. Restart the server and all of the 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 name takes 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 authentication 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/nodes/<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`. For example:

```
<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)

If you selected the Native OS authentication option during installation, perform additional configuration to allow login using accounts on the host system.

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**

Automatic processes may reset privileges on `/etc/shadow`. If you cannot log in to ESP, check the privileges on `/etc/shadow` and re-add user accounts to the shadow group as necessary.

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 User-Name-Password Option

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

Note: SAP does not recommend using the preconfigured user-name-password option in a production environment; it is provided for use in development and testing. Preconfigured logins are file-based. Although you can encode the passwords in `csi_local.xml`, preconfigured logins can be compromised (intentionally or otherwise) by any user with access to that file.

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, the name of your cluster configuration file takes the form `<node-name>.xml`.

When preconfigured user-name-password is the active authentication method, the `<node-name>.xml` file refers to the `csi_local.xml` file, which contains the user names and passwords you configure. Event Stream Processor provides a default `csi_local.xml` file you can use as a basis for creating your own preconfigured user-name-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 encode the passwords, configure `csi_local.xml` to specify the user names and enter the encoded passwords, then modify the `<node_name>.xml` file to specify the preconfigured username-password option as the active authentication method.

1. Shut down the cluster.

If you make changes while the cluster is running, ESP will not apply them until you restart the cluster.

2. Create and encode 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 encode.
   c) The encoded password displays on screen. Copy the password to paste it into the `csi_local.xml` file.
3. 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 the `value` attribute to the desired user name.
   d) In the `<Configuration>` section, locate `<options name="password"` and change the `value` attribute by pasting the encoded password you copied from the CSI utility. Enclose the password in quotation marks.
   e) To add a second login, use a separate <authenticationProvider> element. Encode another password (see step *2* on page 35) and include controlFlag and name attributes with the values shown here:

```
<authenticationProvider controlFlag="sufficient"
name="com.sybase.security.core.PreConfiguredUserLoginModule">
  <options name="username" value="sybase2"/>
  <options name="password"
value="{SHA-256:gIQWZYOPQVM=}jqHtsTPcw8kGkZt1PQeveUAhQncAQhHXJBrjZAqTfk4
="/>
</authenticationProvider>
```

   f) Save and close the `csi_local.xml` file.
4. 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/nodes/<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.
5. Restart the cluster.

See also

- *Starting a Node or Cluster* on page 85
- *Stopping a Node or Cluster* on page 86

Configuring Cascading Authentication Methods

Configure Event Stream Processor to cascade through a list of authentication methods and use the first available method.

A single authentication method can be a single point of failure. If you are using only LDAP authentication, for example, and your LDAP server goes down, users will no longer be able to authenticate using LDAP—so no one can log in. Cascading through two or more 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 file. Under normal circumstances, this will be the authentication method for your system. If that method becomes unavailable, the server tries to use the second method in the file, then the third, continuing down the list until it finds a working authentication provider. If any attempt is successful, users can continue authenticating (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>` or `<config:authenticationProvider>` section and paste it into `csi_all.xml`. Arrange the sections in the order you want the server to try them.
4. Make sure each `<authenticationProvider>` or `<config:authenticationProvider>` includes `controlFlag="sufficient"`, as in the first line of this LDAP example:

```
<config:authenticationProvider controlFlag="sufficient"
name="com.sybase.security.ldap.LDAPLoginModule">
  <config:options name="ServerType" value="openldap"/>
  <config:options name="ProviderURL" value="ldap://your-open-ldap-
host.your-domain.com:389"/>
  <config:options name="DefaultSearchBase" value="dc=your-
domain,dc=com"/>
  <config:options name="RoleSearchBase" value="dc=your-domain,dc=com"/>
  <config:options name="AuthenticationScope" value="subtree"/>
  <config:options name="RoleScope" value="subtree"/>
</config:authenticationProvider>
```

   When `controlFlag="sufficient"` is set on an authentication provider, you can log in once you have authenticated against that provider. Anyone who tries to log in using an authentication provider that lacks controlFlag="sufficient" must authenticate multiple times. Event Stream Processor cycles through the authentication providers in the order in which they appear in the file; you cannot log in until:
   - You authenticate against a provider whose definition includes `controlFlag="sufficient"`, or
   - You authenticate against every provider in the CSI file.
5. Save and close `csi_all.xml`.
6. Use a text editor to open the cluster configuration file, `ESP_HOME/cluster/nodes/<node-name>/<node-name>.xml`.
7. 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>
```

8. Save and close the file.
9. Restart the server and all of the 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 23
- *Secure Sockets Layer (SSL) Connections* on page 50
- *Generating the Java Keystore* on page 51
- *Generating PEM Format Private Keys* on page 53
- *Encryption* on page 53

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. When you use the *any role in a policy, no call is made to the security provider 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*.

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.

Policy Type	Resource	Action	Description
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.
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.

Policy Type	Resource	Action	Description
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> element.
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 46
- *Cluster Administrative Tool* on page 92

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 role checking for native OS or preconfigured logins in CSI files. (Role checking 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. To enable access control, edit the node's configuration file, `ESP_HOME/cluster/nodes/<nodename>/<node-name>.xml`. 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 access control is enabled, a login call from a client causes the security provider to authenticate the user. When the user tries to perform an action on a resource, the server determines whether 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-->
   ```

   When access control is disabled, the server performs no access control checking; any authenticated user can perform any action on any resource.

See also

- *Configuring Access Control* on page 41

Configure ESP for Sybase Control Center

Complete these configuration tasks if you plan to use Sybase Control Center to monitor or manage Event Stream Processor.

Configuring Policies for Monitoring and Administering Event Stream Processor

Edit the Project, Node, and Cluster policies in the `policy.xml` file to grant SCC monitoring and administrative access to a native OS, preconfigured login, LDAP, or SAP Business Intelligence (BI) group. Do not edit the `policy.xml` file if you are using Kerberos or RSA authentication for ESP as you are automatically granted SCC monitoring and administrative access.

The `policy.xml` file must be identical on every node in a cluster. In a multinode cluster where nodes are installed on different hosts, this is often accomplished by placing the ESP `security` directory on a shared drive. If your cluster's `policy.xml` file does not reside on a shared drive, make this change to the `policy.xml` for each node in the cluster.

1. Open `ESP-5_1\security\policy.xml`.
2. In the `<Role>` element, specify the group to which you wish to grant monitoring and administrative access and verify that the following is present:

   The sample below grants monitoring and administrative access to a group called "sybase".

```
<Policies>

        <Policy type="Project">
                <Subjects>
                        <Role>sybase</Role>
                </Subjects>
                <Resources>
                    <!--The group has "read" privileges for "any"
project resource, including meta-data streams and other project
streams-->
                        <Resource>*any</Resource>
                </Resources>
                <Actions>
                        <Action>read</Action>
                </Actions>
        </Policy>

        <Policy type="Node">
                <Subjects>
                        <Role>sybase</Role>
                </Subjects>
                <Resources>
                        <Resource>Node</Resource>
                </Resources>
                <Actions>
                        <Action>read</Action>
                        <Action>stop</Action>
                </Actions>
        </Policy>
        <Policy type="Cluster">
                <Subjects>
                        <Role>sybase</Role>
                </Subjects>
                <Resources>
                        <Resource>Security</Resource>
                        <Resource>Node</Resource>
                        <Resource>Workspace</Resource>
                        <Resource>Application</Resource>
                </Resources>
                <Actions>
                        <Action>read</Action>
                        <!--This privilege is required for write 
operations, such as reload policy, add workspace/project, and so 
on-->
                        <Action>write</Action>
                        <!--This privilege is required for stop 
operations-->
                        <Action>stop</Action>
                        <!--The privilege is required for start 
operations-->
                        <Action>start</Action>
                </Actions>
        </Policy>
</Policies>
```

To enable users within the group you specified in the `policy.xml` file to monitor and administer ESP, map this group to the espMonitorRole and espAdminRole roles.

Configuring Event Stream Processor for Monitoring

To enable users to monitor ESP node and cluster activity using Sybase Control Center, map a native OS, preconfigured login, LDAP, or SAP BI group to the espMonitorRole role.

The ESP node uses your corresponding authentication provider to determine which groups a user belongs to and then uses the `csi_role_mapping.xml` file to map these groups to the appropriate roles. This file is located in the `ESP-5_1\security` directory, which by default is installed in the `Sybase` directory. You may choose to map an existing group to espMonitorRole or create a new group.

1. Open `ESP-5_1\security\csi_role_mapping.xml`.

   The <Mapping> element represents a mapping for a logical role. The <LogicalName> element represents the role that SCC checks for this mapping. There can only be one <LogicalName> per mapping. Do not modify this element.
2. Set the LDAP group within the <MappedName> element. This element represents the group you wish to map to the logical role. You can include more than one <MappedName> element for a mapping.

   You can have the same <MappedName> element for two different <LogicalName> elements.

Here is an example of a mapping where the group "Administrators" maps to espMonitorRole:

```
<Mapping>
<LogicalName>espMonitorRole</LogicalName>
<MappedName>Administrators</MappedName>
</Mapping>
```

Here is an example of a mapping where several groups are mapped to espMonitorRole:

```
<Mapping>
<LogicalName>espMonitorRole</LogicalName>
<MappedName>IT</MappedName>
<MappedName>Developers</MappedName>
<MappedName>Operators</MappedName>
</Mapping>
```

Next

There are additional configuration tasks for Sybase Control Center; see the SCC online help for details.

Configuring Event Stream Processor for Administration

To enable users to start and stop ESP nodes, projects, and adapters from the Sybase Control Center Administration Console, and to view node and project log files, map a native OS, preconfigured login, LDAP, or SAP BI group to the espAdminRole role.

The ESP node uses your corresponding authentication provider to determine which groups a user belongs to and then uses the `csi_role_mapping.xml` file to map these groups to the appropriate roles. This file is located in the `ESP-5_1\security` directory, which by default is installed in the `Sybase` directory. You may choose to map an existing group to espAdminRole or create a new group.

1. Open `ESP-5_1\security\csi_role_mapping.xml`.

   The <Mapping> element represents a mapping for a logical role. The <LogicalName> element represents the role that SCC checks for this mapping. There can only be one <LogicalName> per mapping. Do not modify this element.
2. Set the LDAP group within the <MappedName> element. This element represents the group you wish to map to the logical role. You can include more than one <MappedName> element for a mapping.

   You can have the same <MappedName> element for two different <LogicalName> elements.

Here is an example of a mapping where the group "Administrators" maps to espAdminRole:

```
<Mapping>
<LogicalName>espAdminRole</LogicalName>
<MappedName>Administrators</MappedName>
</Mapping>
```

Here is an example of a mapping where several groups are mapped to espAdminRole:

```
<Mapping>
<LogicalName>espAdminRole</LogicalName>
<MappedName>IT</MappedName>
```

```
<MappedName>Developers</MappedName>
<MappedName>Operators</MappedName>
</Mapping>
```

Next
There are additional configuration tasks for Sybase Control Center; see the SCC online help for details.

Recovering Administrative Access

If you lose administrative access rights to SAP 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 SAP 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 SAP 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 23
- *Access Control* on page 37
- *Generating the Java Keystore* on page 51
- *Generating PEM Format Private Keys* on page 53
- *Encryption* on page 53

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 23
- *Access Control* on page 37
- *Secure Sockets Layer (SSL) Connections* on page 50
- *Generating PEM Format Private Keys* on page 53
- *Encryption* on page 53

Generating PEM Format Private Keys

Convert the Java keytool to generate privacy enhanced mail (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 23
- *Access Control* on page 37
- *Secure Sockets Layer (SSL) Connections* on page 50
- *Generating the Java Keystore* on page 51
- *Encryption* on page 53

Encryption

ESP provides two utilities for encryption, **esp_cluster_admin** and **esp_encrypt**. These utilities support password encryption for internal adapter, service, cluster, and project configuration files. Event Stream Processor also provides the `encrypt.sh` and `encrypt.bat` scripts for encrypting passwords in external adapter configuration files.

Utilities

The **esp_cluster_admin** utility encrypts passwords in project, internal adapter, and service configuration files. Call it from a command line:

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

The **esp_encrypt** utility encrypts passwords in the cluster and for the web services provider, as well as cluster SSL files. Call it from a command line:

```
$ESP_HOME/bin/esp_encrypt [options...]
```

For more information on the utilities and their supported commands, see the *Utilities Guide*.

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.

The script or utility you use for encryption depends on the element you're encrypting, as described in this table.

Encrypt...	In the file...	Using the utility or script...
Cache password	<node-name>.xml	esp_encrypt
Keystore password	<node-name>.xml	esp_encrypt
Key password	<node-name>.xml	esp_encrypt
Project configuration file (CCR) password	<project-name>.ccr	esp_cluster_admin
Adapter CNXML file password	<adapter>.cnxml	esp_cluster_admin
Database service configuration file password	service.xml	esp_cluster_admin
SSL files (server.key and server.crt)	<node-name>.xml (referenced only)	esp_encrypt
Java external adapter configuration file password		encrypt.sh or encrypt.bat
Adapter configuration files	adapter.xml; adapter_config.xml	encrypt.sh or encrypt.bat

Encrypt...	In the file...	Using the utility or script...
Web services provider keystore password	esp_wsp.xml	esp_encrypt

See also

- *Authentication* on page 23
- *Access Control* on page 37
- *Secure Sockets Layer (SSL) Connections* on page 50
- *Generating the Java Keystore* on page 51
- *Generating PEM Format Private Keys* on page 53

Encrypting Passwords in the Cluster Node Configuration File

Use the **esp_encrypt** executable to encrypt passwords in a cluster node configuration file (`<node-name>.xml`) for new nodes, or to re-encrypt existing values.

During installation, ESP encrypts passwords in `<node-name>.xml` for the cache, keystore, and key elements. Encrypt passwords in `<node-name>.xml` only when you configure a new node or cluster (before you start it), or when you need to re-encrypt a password or property using a new key file.

The key password element is optional. While the keystore password locks the entire keystore, the key password only locks a specific key within the store. If the key password is not specified, ESP uses the same password for both elements.

1. Shut down the affected node or, in some cases, the entire cluster:

Before you...	Shut down...
Encrypt a password in `<node-name>.xml` that has not changed (password value is already in the file)	The node
Change and encrypt a password or keypassword in the Cache or Keystore section of `<node-name>.xml`	All nodes in the cluster

Note: If you choose to encrypt passwords in the Keystore element of a new node, first configure the Keystore section of `<node-name>.xml`. The Type, File, and Password elements in Keystore require values. A default value is provided for Type, but you must fill in File and Password values.

2. Use a text editor to open the cluster node configuration file:

   ESP_HOME/cluster/nodes/*<node-name>*/*<node-name>*.xml

3. If you are encrypting a password that is already in the cluster node configuration file, copy the password you want to encrypt.

In the following section of a sample cluster node configuration file, the keystore password is "Pass1234".

```
<Security>
      .
      .
      .
      <Cipher>
          <File>ESP_HOME/cluster/keys/<cluster-name>/cluster.key</File>
      </Cipher>
      <Keystore>
           <Type>JKS</Type>
           <File>ESP_HOME\security\keystore_rsa.jks</File>
           <Password prompt="true">Pass1234</Password>
           <Algorithm>RSA</Algorithm>
      </Keystore>
      .
      .
      .
</Security>
```

4. Note the value in the **Cipher** element. This is the cluster key file required to encrypt passwords. If the **Cipher** element does not exist:
   a) Create a cluster key. From a command line, navigate to `ESP_HOME/bin` and launch the **esp_encrypt** executable using the **--create-key** option:

      ```
      esp_encrypt --create-key cluster.key
      ```

      The command writes a new key to the file `cluster.key`.
   b) Add the **Cipher** element to `<node-name>.xml` using the format in step *3* on page 55.
5. From a command line, navigate to `ESP_HOME/bin` and launch the **esp_encrypt** executable using the **--encrypt** option:

   ```
   esp_encrypt --encrypt <key-file> --text <text>
   ```

   If you enter the **--text** value successfully, the **esp_encrypt** executable writes the encrypted text to the display.
6. Copy and paste the encrypted text from the utility into the cluster node configuration file you opened in step *2*. Replace the original value in the **Password** or **KeyPassword** parameters for the **Keystore** or **Cache** elements with the encrypted text.
7. Ensure that the `encrypted` attribute in each password that receives encrypted text is set to `encrypted="true"`.

   This attribute ensures that the server recognizes the password as encrypted text and decrypts it at runtime. If the attribute is not set to true, the server does not recognize the password as encrypted text and tries to process the password without decrypting it, resulting in errors.
8. Save and close the cluster node configuration file.

See also

- *Stopping a Node or Cluster* on page 86

Encrypting Passwords for Configuration Files

Encrypt passwords within project configuration (CCR) files, adapter CNXML files, and database service configuration files using the **esp_cluster_admin** utility to avoid displaying sensitive data in plain text.

Prerequisites

Configure and start your cluster.

Task

Modify the adapter `.cnxml` and the database service configuration file only during project environment setup.

1. Use a text editor to open the desired configuration file:

   In the local (Studio) cluster: `<user's-home-dir>/SybaseESP/5.1/workspace/<project-name>/<project-name>.ccr`

   `ESP_HOME/lib/adapters/<adapter>.cnxml`

   `ESP_HOME/bin/service.xml`
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` utility using the **--encrypt_text** command. This command requires host and port information as well as credentials for the ESP server. For example, where `<Pass1234>` is the password you want to encrypt, the syntax is:

   ```
   esp_cluster_admin --uri=esp[s]://<host>:<port> --
   username=<username> --password=<password> --encrypt_text --
   text=<Pass1234>
   ```

   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.

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

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.
5. Change the `encrypted="false"` attribute for the **Password** parameter to `encrypted="true"`.

   The `encrypted` 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 tries to process the password without decrypting it, resulting in errors.
6. Save and close the configuration file.

Encrypting Passwords for the ESP Web Services Provider

Use the **esp_encrypt** executable to encrypt the keystore password for the ESP Web Services Provider.

During installation, ESP encrypts the keystore password in `esp_wsp.xml`. Encrypt the keystore password only when you configure a new node or cluster (before you start it), or when you need to re-encrypt a password using a new key file.

1. Shut down all nodes in the cluster.
2. Use a text editor to open the ESP Web Services Provider configuration file:

   `ESP_HOME/wsp/esp_wsp.xml`
3. Copy the keystore password. If the keystore password is not in the configuration file, add the `password` parameter to the keystore element and set it to `"true"`.

   In the following section of a sample cluster node configuration file, the keystore password is "Pass1234".

```
<Security>
      <Keystore>
           <Type>JKS</Type>
           <File>keystore.jks</File>
           <Password prompt="true">Pass1234</Password>
      </Keystore>
      <Cipher>
            <File>ESP_HOME/wsp/wsp.key</File>
      </Cipher>
</Security>
```

4. Note the value in the **Cipher** element. This is the cluster key file required to encrypt passwords. If the **Cipher** element does not exist:

   a) Create a cluster key. From a command line, navigate to `ESP_HOME/bin` and launch the **esp_encrypt** executable using the **--create-key** option:

   ```
   esp_encrypt --create-key wsp.key
   ```

The command writes a new key to the file `wsp.key`.

b) Add the **Cipher** element to `<node-name>.xml` using the format in step *3* on page 58

5. From a command line, navigate to `ESP_HOME/bin` and launch the **esp_encrypt** executable using the **--encrypt** option:

```
esp_encrypt --encrypt <key-file> --text <text>
```

   If you enter the **--text** value successfully, the **esp_encrypt** executable writes the encrypted text to the display.

6. Copy and paste the encrypted text from the utility into the cluster node configuration file you opened in step *2*. Replace the original password in the **Password** parameter for the **Keystore** element with the encrypted text.
7. Ensure that the `encrypted` attribute in the password parameter is set to `encrypted="true"`.

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

8. Save and close the ESP Web Services Provider configuration file.

Encrypting SSL Files

Use the **esp_encrypt** executable to encrypt secure sockets layer (SSL) files (`server.key` and `server.crt`) for new nodes, or to re-encrypt existing files.

During installation, ESP encrypts SSL files and references them in `<node-name>.xml`. To indicate that they are encrypted, the files gain the `.enc` extension, becoming `server.key.enc` and `server.crt.enc`. Encrypt SSL files only when you configure a new node or cluster (before you start it), or when you need to re-encrypt SSL files using a new key file. By default, ESP looks for encrypted and unencrypted SSL files in `ESP_HOME/cluster/keys/<cluster-name>`.

The ESP installer provides only encrypted SSL files. To configure SSL files for a new cluster, either:

- Use OpenSSL or a similar toolkit to generate your own `server.key` and `server.crt` in privacy enhanced mail (PEM) format, or;
- Copy existing SSL files to the new cluster, then use a new cluster key file to re-encrypt the files.

1. Shut down all nodes in the cluster.
2. Use a text editor to open the cluster node configuration file:

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

3. Note the value in the **Cipher** element. This is the cluster key file required to encrypt SSL files. If the **Cipher** element does not exist:

a) Create a cluster key. From a command line, navigate to `ESP_HOME/bin` and launch the **esp_encrypt** executable using the **--create-key** option:

```
esp_encrypt --create-key cluster.key
```

The command writes a new key to the file `cluster.key`.

b) Add the **Cipher** element to `<node-name>.xml using the following format:`

```
<Security>
      .
      .
      .
      <Cipher>
          <File>ESP_HOME/cluster/keys/<cluster-name>/cluster.key</File>
      </Cipher>
      .
      .
      .
</Security>
```

Although each cluster key can encrypt a specific file multiple times, it encrypts the file the same way every time. Since ESP encrypts files during installation, create a new key file to re-encrypt SSL files for a new cluster.

4. From a command line, navigate to `ESP_HOME/bin` and launch the **esp_encrypt** executable. Do one of the following to encrypt either the `server.key` file or the `server.crt` file:

a) To encrypt an SSL file for the first time within `<node-name>.xml,` use the **--encrypt** option with the cluster key file:

```
esp_encrypt --encrypt <key-file> --file <server.key>
```

b) To re-encrypt an SSL file within `<node-name>.xml`, create a new `cluster.key` file (see step *3* on page 59). Then, use the **--re-encrypt** option with the cluster key file:

```
esp_encrypt --re-encrypt <old-key-file> <new-key-file> --file
<server.key.enc>
```

Note: esp_encrypt works on a file with any name, allowing you to keep multiple copies of your SSL files. At runtime, however, ESP looks for SSL files with these names:

```
server.key
server.crt
server.key.enc
server.crt.enc
```

The setting of the ssl-key-file-encrypted property (see step *5* on page 61) determines whether ESP looks for SSL files with or without the `.enc` extension.

The SSL files gain the `.enc` extension, marking them as encrypted.

5. In Controller|ApplicationTypes in `<node-name>.xml`, ensure that the ssl-key-file-encrypted property in project and ha_project ApplicationType elements is set to true.

   This attribute ensures that the server recognizes the file as encrypted and decrypts it at runtime. If the attribute is not set to true, the server does not recognize the file as encrypted and tries to process the file without decrypting it, resulting in errors.
6. Ensure that the ssl-key-file property points to the location of the encrypted SSL files:

   ESP_HOME/cluster/keys/<cluster-name>
7. Save and close the cluster node configuration file.

See also

- *Enabling and Disabling SSL* on page 98

Encrypting Passwords for Java External Adapters

Use an independent keystore to encrypt passwords in external adapter configuration files, and to tell Event Stream Processor to decrypt encrypted values at runtime.

Prerequisites

Set the JAVA_HOME environment variable. Event Stream Processor supports SAP JVM 7.1.011.

Task

Java external adapter configuration files contain an encryption algorithm that Event Stream Processor uses to authorize decryption.

1. Use any text editor to open the desired external adapter configuration file.
2. Call the `encrypt.sh` (UNIX) or the `encrypt.bat` (Windows) 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 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` (UNIX) or `decrypt.bat` (Windows) script to validate encrypted text. To run the decrypt command against the encrypted text, call the `decrypt` script and provide the same credentials you provided for the `encrypt` 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 Event Stream Processor recognizes the password as encrypted text and is able to decrypt the password at runtime. If the attribute is set to false, ESP does not recognize the password as encrypted text and 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

Authentication Type	Required Value
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.

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[s]://<host>:19011 --
   username=<username> --password=<password>
   > add project <workspace-name>/<project-name> <project-name>.ccx
   <project-name>.ccr
   >
   ```

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

- *Configuring Security* on page 22
- *Configuring External Database Access* on page 73
- *High Availability* on page 12
- *Centralized Security* on page 15
- *File and Directory Infrastructure* on page 3

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 (debug-level)	Sets 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

Project Option	Description
Performance Monitor Refresh Interval (time-granularity)	Defines the performance monitor refresh interval, or time granularity, within the project. This option specifies, in seconds, how often the set of performance records—one per stream and one per gateway connection—is obtained from the running Event Stream Processor. By default, the performance monitor refresh interval is set to 5. Set this option to 0 to disable monitoring; this also optimizes performance.
Java Classpath	Sets the Java classpath. Value is the path to the classpath file.
Java Max Heap (java-max-heap)	Sets the max Java heap for the project. Default value is 256 megabytes.
Bad Record File (bad-record-file)	Saves bad records to a file that you specify. When this option is omitted (the default), bad records are discarded. Default file name is `esp_bad_record_file`. If the value is a file name with no path, ESP places the file in a default location: `<base-directory>/<workspace-name>.<project-name>.<instance-number>` where `<base-directory>` is a property defined in the cluster configuration file: • In the local (Studio) cluster: `ESP_HOME/studio/clustercfg/localnode.xml` • In a remote cluster: `ESP_HOME/cluster/nodes/<node-name>/<node-name>.xml`
Utf8	Enables 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. For example, when set to true, you don't receive output if a window gets updates for a key but the window's column value does not change. You receive output only when the column value changes. When set to false, you receive output with every update regardless of whether the column value changes. Default value is true.

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	Sets the constant interval expression that specifies the maximum age of rows in a window, in seconds. Default value is 1.
Precision	Sets decimal display characteristics for number characters in the project. Default value is 6.
Memory	Sets memory usage limits for the project in megabytes. Default is 0, meaning unlimited.
Command Port	Sets an explicit command port number. Change the value 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 65535.
SQL Port	Sets an explicit SQL port number. Change the value 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 65534.

Project Option	Description
Gateway Port	Sets an explicit gateway port number. Change the value 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 65533.
Consistent Recovery	If set to true, the project runs in consistent recovery mode. Any window that you have assigned to a log store will be recoverable to the last checkpointed state. Related windows do not have to be assigned to a single log store because checkpointing of multiple log stores is done together. If the server or connection fails in the middle of checkpointing a log store, the server restarts the project at the last successful checkpointed state. If you are not using log stores, consistent recovery mode does not take effect. Defaults: For new projects created in Studio: True For existing projects: False
Auto Checkpoint	Sets the maximum number of input transactions, across all input streams and windows, that will be processed before the server issues a checkpoint. A checkpoint may occur before the maximum number of transactions if the server deems it necessary, or if the server is in conistent recovery mode and the publisher issues a commit. If you are not using log stores, Auto Checkpoint does not take effect. When you use Auto Checkpoint without consistent recovery mode, if the server fails in the middle of checkpointing a log store, the server does not roll back to the last successful checkpointed state on restart. If you set the value to 1, a checkpoint is done after every input transaction. This means that only the last record (which would not be completely processed and checkpointed) has the potential of not being recovered. If you set the value higher than 1, you have the potential for higher data loss, but will have better system performance. If you set the value to 0, Auto Checkpoint is not activated. Checkpoints occur only as the server determines, or if a publisher issues a commit. Default value is 0 (Auto Checkpoint disabled).

Project Option	Description
Meta Store Size	In order to persist state across server restarts, the server creates a metadata log store. This store holds the information (tracked by the ESP_GD_Sessions metadata stream) regarding the amount of data each guaranteed delivery subscriber for this project has read for a given stream. Default store size is 64 MB.
Meta Store Directory	You can change the location of the metadata log store. The default location is a subdirectory of the project working directory: `ESP_HOME/cluster/projects/<cluster-name>/<workspace-name>.<project-name>.<instance-number>/esp_metadata` The `esp_metadata` directory must be located on a shared disk accessible to all nodes in the cluster.

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

- *High Availability* on page 12

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.

Affinities

Set controller and instance affinities in the CCR file 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 recovered, the project restart fails and you must restart manually. A weak positive affinity causes the project starts on the preferred controller if possible, but if that controller is unavailable, it may start on another available controller.
Charge	Specify **Positive** or **negative**. If positive, the project runs (for a strong affinity) or prefers to run (for a weak affinity) on the named controller. If negative, the project does not run (or prefers not to run) on the named 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.

Cold Failover

Enable or disable cold failover for projects and set failover options.

A project fails when it does not run properly or stops running properly. If cold 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.

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 a deployment with failover enabled and affinities configured on both instances of an active-active (HA) 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>
```

Configuring 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.
- To connect to an SAP HANA database, you must have an SAP HANA ODBC client installed. SAP recommends that you use the latest version of the SAP HANA ODBC client available, but it must be at least version 1.0.67.
- Specify service definitions in the `service.xml` file for your desired database connections.

If you make changes to the `service.xml` file while the ESP Server is running, the Server does not recognize the changes until you restart it. If you are running on the local cluster, which the ESP Studio starts on its own, restart the ESP Studio as well.

Adapter	Supported Drivers	Supported Databases
SAP® Sybase® Adaptive Server® Enterprise (ASE) Output Adapter	Open Client™	SAP Sybase ASE
Database Input and Output Adapters	JDBC	• SAP Sybase ASE • IBM DB2 • Oracle • Kx Systems KDB+ • Microsoft SQL Server • SAP HANA®
Database Input and Output Adapters	ODBC	• SAP Sybase ASE • SAP® Sybase® IQ • SQL Anywhere® • IBM DB2 • Oracle • Microsoft SQL Server • TimesTen • MySQL 5.x • PostgreSQL • SAP HANA
SAP HANA Adapter	ODBC	SAP HANA
SAP Sybase IQ	ODBC	SAP 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 63

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 2.3.0 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`.

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>
```

If you make changes to the `service.xml` file while the ESP Server is running, the Server does not recognize the changes until you restart it. If you are running on the local cluster, which the ESP Studio starts on its own, restart the ESP Studio as well.

See also

- *Linking to Your ODBC Driver Manager Library* on page 83

Configuring a JDBC Connection to an External Database

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

Prerequisites

- Back up `ESP_HOME/bin/service.xml`.
- Open `service.xml` using an xml-compatible text editor. You cannot edit `service.xml` using ESP Studio.

Task

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
SAP Sybase ASE	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>` If you are using the HANA JDBC driver, the generic JDBC driver plugin, and want the connect string URL to support SSL encryption settings, specify the **DriverManager** parameter described in the step below.

9. (For the HANA JDBC driver and generic JDBC driver plugin only) Set the **DriverManager** parameter to `true` to enable the HANA JDBC driver to connect to HANA using SSL encryption.

Next

If you make changes to the `service.xml` file while the ESP Server is running, the Server does not recognize the changes until you restart it. If you are running on the local cluster, which the ESP Studio starts on its own, restart the ESP Studio as well.

Configuring an ODBC Connection to an External Database

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

Prerequisites

- Back up `ESP_HOME/bin/service.xml`.
- Open `service.xml` using an xml-compatible text editor. You cannot edit `service.xml` using ESP Studio.

Task

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: For Windows, the **DriverLibrary** parameter can only be set to esp_db_odbc_lib. For Linux and Solaris, there are 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 and Solaris, 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.

Next

If you make changes to the `service.xml` file while the ESP Server is running, the Server does not recognize the changes until you restart it. If you are running on the local cluster, which the ESP Studio starts on its own, restart the ESP Studio as well.

Configuring an Open Client Connection to a Database

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

Prerequisites

- Back up `ESP_HOME/bin/service.xml`.
- Open `service.xml` using an xml-compatible text editor. You cannot edit `service.xml` using ESP Studio.

Task

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 SAP Sybase ASE 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.

Next

If you make changes to the `service.xml` file while the ESP Server is running, the Server does not recognize the changes until you restart it. If you are running on the local cluster, which the ESP Studio starts on its own, restart the ESP Studio as well.

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 SAP Sybase ASE 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 SAP Sybase ASE database over JDBC; specify the SAP Sybase ASE 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>
```

Here is an example of connecting to the SAP HANA database using a HANA JDBC driver and generic JDBC driver plugin with SSL encryption:

```
<Service Name="HANA_Generic_JDBC_Service" Type="DB">
    <Parameter Name="DriverLibrary">esp_db_jdbc_lib</Parameter>
    <Parameter Name="DataSource">com/sap/db/jdbcext/DataSourceSAP</
Parameter>
    <Parameter Name="ConnectString">host:port://
HANA_DB_SERVER_HOST:PORT_NUMBER/?
encrypt=true&amp;validateCertificate=true&amp;hostNameInCertificate
=HANA_DB_SERVER_HOST&amp;keyStore=KEYSTORE_FILE&amp;keyStoreType=JK
S&amp;keyStorePassword=KEYTSTORE_PASSWORD&amp;trustStore=KEYSTORE_F
ILE&amp;trustStoreType=JKS&amp;trustStorePassword=KEYTSTORE_PASSWOR
D</Parameter>
    <Parameter Name="DriverManager">true</Parameter>
    <Parameter Name="User">USERNAME</Parameter>
    <Parameter Name="Password">PASSWORD</Parameter>
</Service>
```

To connect using SSL encryption, ensure the `DriverManager` parameter is specified and set to true. If you specified a relative keystore filepath, place the keystore file under the base project directory.

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 75

CHAPTER 3 Administer a Cluster

Starting a Node or Cluster

Start a cluster by starting its nodes. This task does not apply to the local (Studio) cluster.

Prerequisites

Add nodes to your cluster and configure them.

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.

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. SAP recommends that you start each node from a separate working directory.

1. To start a node on a Windows host, execute:

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

2. To start a node on a UNIX host,

   - Execute this to start the node in the foreground:

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

   - Execute this to start the node in the background:

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

     where `esp-node-console.out` is a file that collects console output. Specifying a console output file is optional because most console output, which is minimal for Event Stream Processor, is also recorded in `cluster.log`. (You can configure logging in `cluster.log.properties`.) Some users prefer to direct console output to `/dev/null` when running a node in the background.

3. (Optional) Use the cluster administration tool to start and manage projects. Execute a command of this form to enter the tool's interactive mode:

```
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. Use the URI protocol `esps` when the cluster is SSL-enabled. 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.

---

4. (Optional) In **esp_cluster_admin**'s interactive mode,
   - Enter **get projects** to display a list of projects on this node.
   - To deploy a project to this node, enter a command of this form:

     ```
     add project <workspace-name>/
     <project-name> <project-name>.ccx [<projectname>.
     ccr]
     ```

     where `<project-name>.ccx` is the path to the compiled project file and `<project-name>.ccr` is the path to the project's runtime configuration file. Include the CCR file for an HA (active-active) project or a project with affinities.
   - To start a project, enter a command of this form:

     ```
     start project <workspace-name>/
     <project-name> [<instance-index>]
     ```

     where <instance-index> specifies, for an HA project, which of the two instances (0 or 1) you want to start.

See also

- *Adding a Node to a Cluster* on page 16
- *Configuring a Cluster* on page 17

Stopping a Node or Cluster

Shut down the nodes in a cluster.

Prerequisites

SAP 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]://<host>: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]://<host>: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 a server, also known as a cluster manager node.

By default after installation, a local cluster manager connection is already defined in the SAP Sybase ESP 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 SAP Sybase ESP Run-Test perspective.

View cluster setup details for SAP Sybase Event Stream Processor Studio in: `$ESP_HOME/studio/clustercfg/localnode.xml`. Do not modify this file.

1. If you are not already in the SAP Sybase ESP Run-Test perspective, click the **SAP Sybase ESP Run-Test** tab at the top of the window, or select **Window** > **Open Perspective** > **SAP Sybase ESP 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** causes 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 SAP Sybase Event Stream Processor 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.

Each node in the cluster has a cluster node log configuration file, which resides in the node's working directory, from which the node is typically started:

Windows: `%ESP_HOME%\cluster\nodes\<node-name>\cluster.log.properties`

Linux/Solaris: `$ESP_HOME/cluster/nodes/<node-name>/cluster.log.properties`.

Ensure that the cluster node log configuration file is located in the same directory as that node's configuration file, `<node-name>.xml`.

A sample `cluster.log.properties` file:

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

log4j.rootLogger=info, Log

log4j.logger.com.sybase.esp=info
log4j.logger.com.sybase.esp.cluster.loggers.security.Audit=info,
AuditLog
log4j.logger.com.sybase.esp.cluster.applications=info
```

```
log4j.additivity.com.sybase.esp.cluster.security.Audit=false
log4j.additivity.com.sybase.esp.cluster.applications=false

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

log4j.appender.AuditLog=org.apache.log4j.RollingFileAppender
log4j.appender.AuditLog.File=audit.log
log4j.appender.AuditLog.MaxFileSize=1MB
log4j.appender.AuditLog.MaxBackupIndex=5
log4j.appender.AuditLog.layout=org.apache.log4j.EnhancedPatternLayo
ut
log4j.appender.AuditLog.layout.ConversionPattern=%d{MMM dd yyyy
HH:mm:ss.SSS} %p - %m%n

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

In this sample configuration, the com.sybase.esp.cluster.logfile property in the first line specifies that the cluster node log is written to the log file `cluster.log`, which is located by default in the node working directory, `ESP_HOME/cluster/nodes/<node-name>`. If you want to write the cluster node log elsewhere, specify a path relative to the node working directory.

Note: You can also set the log file location directly in the appender that writes the log file. In the example above, change the value of `log4j.appender.A.File` to the desired path and file name, where the path is relative to the node working directory.

`cluster.log` is configured by default 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.

`.com.sybase.esp.cluster.loggers.security.Audit` is available as a unique logger which logs significant security audit events, including user login and log out, user authorization and user session expiration. Specifying a special appender for this logger will allow these events to be written to a separate log file. The lines in the sample configuration file that pertain to isolating the security audit events into a separate file are:

```
log4j.logger.com.sybase.esp.cluster.loggers.security.Audit=info,
AuditLog

log4j.additivity.com.sybase.esp.cluster.security.Audit=false

log4j.appender.AuditLog=org.apache.log4j.RollingFileAppender
log4j.appender.AuditLog.File=audit.log
log4j.appender.AuditLog.MaxFileSize=1MB
log4j.appender.AuditLog.MaxBackupIndex=5
log4j.appender.AuditLog.layout=org.apache.log4j.EnhancedPatternLayo
ut
log4j.appender.AuditLog.layout.ConversionPattern=%d{MMM dd yyyy
HH:mm:ss.SSS} %p - %m%n
```

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 SAP Sybase Event Stream Processor 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 enter 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. To use interactive mode, the password you use to connect to the cluster manager must begin with an alphabetic character.

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=esps://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 4. 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.
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.

Command	Function
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.
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.

Command	Function
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.
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.

Command	Function
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 17
- *Configuring Access Control* on page 41

Safeguarding Your Data

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

SAP 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. SAP 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 **SAP Sybase ESP 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 **SAP Sybase ESP 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*.

Enabling and Disabling SSL

Turn secure sockets layer (SSL) on or off at the node or project level.

SSL is enabled for both nodes and projects by default.

When SSL is enabled at the project level, gateway communication is encrypted. Project-level SSL imposes no user requirements. When SSL is enabled at the node level, all incoming URLs must use HTTPS and URIs must use ESPS. For example, to enter the cluster administration tool's interactive mode:

With SSL enabled: `esp_cluster_admin --uri=esps://myhost:19011 --username=mylogin`

With SSL disabled: `esp_cluster_admin --uri=esp://myhost:19011 --username=mylogin`

The SSL settings for both the node and the project level are in the cluster configuration file, `<node-name>.xml`.

1. Open `ESP_HOME/cluster/nodes/<node-name>/<node-name>.xml` .
2. To enable or disable SSL for the node, set the ssl attribute on the RPC|Port element to true (to enable) or false (to disable). For example:
   ```
   <Port ssl="false">19011</Port>
   ```
3. To enable SSL for a project, set the ssl-key-file property on the project and ha_project ApplicationType elements in Controller|ApplicationTypes to point to the directory that

holds the SSL files, `ESP_HOME/cluster/keys/<cluster-name>`. See the example at the end of this task.

To disable SSL for a project, omit ssl-key-file or leave its value empty.

4. To flag the project’s SSL files as encrypted so Event Stream Processor decrypts them at runtime, set the ssl-key-file-encrypted property to true on project and ha_project ApplicationType elements in Controller|ApplicationTypes. See the example at the end of this task.

   If you set ssl-key-file-encrypted to false when the SSL files are encrypted, or to true when the SSL files are not encrypted, the project does not run.

Example: SSL Properties in `<node-name>.xml`

This example shows SSL properties in the ApplicationTypes element, which is part of the Controller section of `<node-name>.xml`. The ApplicationType shown here is project. The only difference in an ha_project example would be to change `project` to `ha_project` in the second line: `<ApplicationType name="ha_project" enabled="true">`

```
<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">${ESP_HOME}/cluster/projects/Your-
Cluster-Name</Property>
      <Property name="ssl-key-file">${ESP_HOME}/cluster/keys/Your-
Cluster-Name</Property>
      <Property name="ssl-key-file-encrypted">true</Property>
    </Properties>
  </ApplicationType>
</ApplicationTypes>
```

See also

- *Encrypting SSL Files* on page 59

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

SAP 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 Backup

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.

Task

1. If you are backing up a project, use the following command to stop it:

```
$ESP_HOME/bin/esp_cluster_admin --uri=esp[s]://<cluster_server>:
19011 --username=<username> --password=<password> --stop_project
<workspace-name>/<project-name> [<instance-number>]
```

Replace 19011 with your cluster cache number. For *`username`* and *`password`*, enter your user credentials. Use the URI protocol esps for clusters that are SSL-enabled; for clusters that are not, use 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.

2. If you are backing up a cluster, use **esp_cluster_admin** to stop the nodes.
3. Back up the `.ccl` files, the `.ccr` files, and the log stores for each project. On Linux and Solaris systems, use the **tar** system utility. On Windows systems, use the **pkzip** freeware utility or equivalent. To find the individual log stores in a project, follow the path `<base-directory>/<workspace-name>.<project-name>.<instance-number>.`

See also

- *Stopping a Node or Cluster* on page 86
- *Cluster Administrative Tool* on page 92

Performing an Online Backup

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 of your cluster cache. 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. Shut down the entity whose files you are restoring.
    - If you are restoring a project, use the cluster administration tool to stop the project.

- If you are restoring a cluster, use the cluster administration tool to stop each node in the cluster.

2. Extract the files to restore from the backup file.
   For example, to extract files from `backup.tar`, use:

```
cd \
tar -xvf backup.tar
```

3. If the files you want to restore were created using the online backup process, rename all `.bak` files to `.log` files.
4. Restart the project or cluster.

See also

- *Stopping a Node or Cluster* on page 86
- *Starting a Node or Cluster* on page 85
- *Cluster Administrative Tool* on page 92

Restoring Backup Files on Windows Systems

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

1. Shut down the entity whose files you are restoring.
   - If you are restoring a project, use the cluster administration tool to stop the project.
   - If you are restoring a cluster, use the cluster administration tool to stop each node in the cluster.
2. Extract the files to restore from the backup file.
   For example, to extract files from `backup.zip` use:

```
c:
cd \
pkunzip backup.zip
```

3. If the files you want to restore were created using the online backup process, rename all `.bak` files to `.log` files.
4. Restart the project or cluster.

See also

- *Stopping a Node or Cluster* on page 86
- *Starting a Node or Cluster* on page 85
- *Cluster Administrative Tool* on page 92

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"/>
```

The CCR file resides in the local (Studio) cluster; the default location is

```
<user's-home-dir>/SybaseESP/5.1/workspace/<projectname>/
<project-name>.ccr
```

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 is disabled for all projects. Users may choose to leave the time-granularity project option disabled when monitoring is not required, to increase performance. The `.ccr` file and SAP Sybase Event Stream Processor Studio set this project option. For more information on configuring projects in SAP Sybase Event Stream Processor 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 SAP 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 SAP 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://help.sap.com/database*.

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.

Several adapters have been created using the adapter toolkit and consist of various modules (transporters, formatters, ESPPublisher or ESPSubscriber). See *Preconfigured Adapters Included with the Adapter Toolkit* in the *Building Custom Adapters* guide for a full list of these adapters. Note that:

- Output transporter modules that run in streaming mode do not report statistics to the _ESP_Adapter_Statistics metadata stream.
- Output transporter modules that run in row mode report statistics to the _ESP_Adapter_Statistics metadata stream.
- Input transporter modules that run in either streaming or row mode report statistics to the _ESP_Adapter_Statistics metadata stream.

See *Transporters Currently Available from SAP* and *Formatters Currently Available from SAP* in the *Building Custom Adapters* guide to determine in which mode a module operates. See *Adapter Controller Parameters* for the adapter of your choice in the *Adapters Guide* for details on tuning the frequency that the adapter reports its statistics to the _ESP_Adapter_Statistics metadata 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.

Column	Type	Description
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.

Column	Type	Description
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.
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.

Column	Type	Description
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.
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.
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.

Column	Type	Description
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_GD_Sessions

Contains information about registered guaranteed delivery sessions that may be active or inactive. In a guaranteed delivery session, a guaranteed delivery window transmits event information to a registered guaranteed delivery subscriber.

The _ESP_GD_Sessions metadata stream tracks all the guaranteed delivery sessions for a project. You can monitor the streams being subscribed to in a given guaranteed delivery sesssion, the client handle associated with a session and the last sequence number committed for a given GD session/stream combination.

When a project contains at least one stream that supports guaranteed delivery, Event Stream Processor stores _ESP_GD_Sessions in a metadata log store so the stream can be recovered after a restart. If a project contains no GD streams, _ESP_GD_Sessions is stored in a memory store, but it is not used.

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

Column	Type	Description
gd_key	`string`	The automatically generated key for the guaranteed delivery session. The gd_key is unique for a given gd_name/user_name combination.
stream_name	`string`	The name of the stream this guaranteed delivery session subscribes to.
user_name	`string`	The user associated with this guaranteed delivery session.
gd_name	`string`	The name of this guaranteed delivery session.

Column	Type	Description
sequence_no	`long`	The sequence number of the last event committed from stream_name in this guaranteed delivery session. A value of 0 indicates that no commits have been issued.
client_handle	`long`	The active client handle associated with this guaranteed delivery session. A value of -1 indicates that there are no active clients for this guaranteed delivery session. For active connectors, the value is 0.
last_update	`bigdate-time`	The last time this stream entry was updated.

The key for this stream consists of the gd_key plus the stream_name—that is, there is one stream entry per gd_key/stream_name pair.

_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.
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.
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".

Key	Value	Stream	Description
REQUES-TEXIT	None	None	A request to shut down the project received.
EXIT	None	None	All the user streams have exited and the project is about to shut down.

_ESP_Streams

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

Column	Type	Description
user_name	`string`	Hardcoded as "user".
stream_name	`string`	The name of the stream described by this row.
handle	`long`	The stream's numeric ID.
type	`string`	The type of the stream: "stream", "deltastream", "window", or "metadata".
visibility	`string`	The visibility of the stream: "input", "output", "local", or "in-termediate".
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.
gdmode	integer	Indicates whether guaranteed delivery is enabled for the stream: • 0 - does not support guaranteed delivery • 1 - supports guaranteed delivery • 2 - supports guaranteed delivery with check points Guaranteed delivery with check points is available only when the server is running in consistent recovery mode or the auto check point option is enabled.

_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.
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.

Column	Type	Description
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.

Column	Type	Description
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 sub-scription.

_ESP_Tables

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

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

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

CHAPTER 5 Troubleshoot a Cluster

Resolve problems with SAP Sybase Event Stream Processor.

Tip: If a problem arises, you can often find the cause in the logs. ESP writes two project-level logs and a cluster/node-level log. For the names and locations of the logs, see *File and Directory Infrastructure* on page 3.

Login and Connection Problems

Solve problems with authentication and with connecting to servers or projects.

Error: Invalid Login Credentials

Problem: You cannot log in; when you try, SAP Sybase Event Stream Processor returns an error:

```
[error] security : Authentication failure:Invalid login credentials
```

Solution: Check for the following:

- Are the user ID and password valid and spelled correctly?
- Is your authentication method set up correctly? ESP supports a variety of authentication providers, all of which require configuration. To identify the configuration method or methods enabled on this node, open `ESP_HOME/cluster/nodes/<node-name>/<node-name>.xml` and look in the Security | Csi | File element. The file specified here is called the CSI file.

If the <File> Value Is...	The Authentication Method Is...
`csi_boe.xml`	SAP BI
`csi_kerberos.xml`	Ticket-based Kerberos
`csi_ldap.xml`	LDAP
`csi_local.xml`	ESP native account (preconfigured login)
`csi_native_nt.xml`	Windows
`csi_native_unix.xml`	UNIX (Linus or Solaris)
`csi_rsa.xml`	RSA

If the <File> Value Is...	The Authentication Method Is...
<any other name>	Cascading (two or more of the methods above). There is no default CSI file name when the node uses cascading authentication. The file might be called `csi_all.xml`, `csi_cascade.xml`, or something else. Read the file to see which methods are enabled.

- If you are using UNIX authentication, have you configured a pluggable authentication module?
- If you are using multiple cascading authentication methods, have you set `controlFlag="sufficient"` on each `<authenticationProvider>` or `<config:authenticationProvider>` element in the CSI file? This flag allows you to authenticate using only one authentication provider. Anyone who tries to log in using an authentication provider that lacks `controlFlag="sufficient"` must authenticate multiple times. ESP cycles through the authentication providers in the order in which they appear in the file; you cannot log in until:
  - You authenticate against a provider whose definition includes `controlFlag="sufficient"`, or
  - You authenticate against every provider in the CSI file.

See also

- *Authentication* on page 23
- *Configuring a Pluggable Authentication Module (PAM)* on page 33
- *Configuring Cascading Authentication Methods* on page 36

A Studio Project Does Not Run, Reports Login Failure

Problem: Attempts to run an SAP Sybase Event Stream Processor Studio project in a remote cluster fail with these errors:

```
Failed to connect to server "esp://localhost:19011".
Reason: "Failed to login server"
```

Studio reports these errors when it has an SSL mismatch with the ESP server: either SSL is enabled on the server and Studio's connection definition for that server does not include SSL, or the connection definition does include SSL but SSL is not enabled on the server.

Solution: Correct the connection definition in Studio. For details, look for instructions on creating and modifying remote cluster connections in the *Studio Users Guide*.

A Utility Fails to Connect to the Server

Problem: Using a command-line utility such as **esp_client** or **esp_subscribe**, you fail to connect to the ESP server.

The command might return this message when it fails:

```
Couldn't connect to server, XML-RPC Fault(-504)
```

Utilities affected by this issue include **esp_client**, **esp_cnc**, **esp_convert**, **esp_kdbin**, **esp_kdbout**, **esp_query**, **esp_subscribe**, and **esp_upload**.

Solution: Check the following:

- Are you using the correct host name, port number, and login details in the command?
- Is the server up and reachable? Try a ping.
- Are you using **-e** correctly? If SSL is enabled on the server, use the utility’s **-e** flag. For example:

  ```
  > esp_client -c your_user_name:your_password
  -p localhost:51011/your_workspace/your_project -e
  esp_client> loglevel 7
  esp_client> stop
  esp_client> quit
  ```

If SSL is not enabled on the server, do not use **-e**—the connection also fails if **-e** is present when it should not be.

For details on command syntax, see the *Utilities Guide*.

A Utility Fails to Connect to a Project

Problem: Using a command-line utility such as **esp_client**, you cannot connect to a project.

For example:

```
% esp_client -c your_user_name:your_password -p localhost:
51011/your_workspace/your_project
ASAP_loginToCluster( your_user_name ) failed, status = -1
```

Solution: Check for the following:

- Are the user ID and password valid and spelled correctly?
- Are the workspace name and project name correct?
- If access control is enabled, does this user have permissions that allow access to the project?
- Is the project running? You can use **esp_cluster_admin** to find out and to start the project if necessary.

See also

- *Cluster Administrative Tool* on page 92
- *Access Control* on page 37

Project Problems

Solve problems with projects.

A Project Fails to Restart

Problem: A project or project instance fails to restart when its node goes down.

If the project has a strong positive affinity to the downed controller node, it tries to restart on that controller, fails because the node is down, and gives up—a project does not make repeated restart attempts when no appropriate controller is available.

Solution: Change your project’s controller affinity to weak so the project can use other controllers.

See also

- *Affinities* on page 69

A Studio Project Does Not Run, Reports Login Failure

Problem: Attempts to run an SAP Sybase Event Stream Processor Studio project in a remote cluster fail with these errors:

```
Failed to connect to server "esp://localhost:19011".
Reason: "Failed to login server"
```

Studio reports these errors when it has an SSL mismatch with the ESP server: either SSL is enabled on the server and Studio’s connection definition for that server does not include SSL, or the connection definition does include SSL but SSL is not enabled on the server.

Solution: Correct the connection definition in Studio. For details, look for instructions on creating and modifying remote cluster connections in the *Studio Users Guide*.

A Utility Fails to Connect to a Project

Problem: Using a command-line utility such as **esp_client**, you cannot connect to a project.

For example:

```
% esp_client -c your_user_name:your_password -p localhost:
51011/your_workspace/your_project
ASAP_loginToCluster( your_user_name ) failed, status = -1
```

Solution: Check for the following:

- Are the user ID and password valid and spelled correctly?
- Are the workspace name and project name correct?
- If access control is enabled, does this user have permissions that allow access to the project?
- Is the project running? You can use **esp_cluster_admin** to find out and to start the project if necessary.

See also

- *Cluster Administrative Tool* on page 92

- *Access Control* on page 37

A Project Runs in the Wrong Cluster

Problem: One or more projects are running on nodes that you thought belonged to another SAP Sybase Event Stream Processor cluster.

Clusters merge when members of two or more clusters on the same network:

- Have overlapping lists of manager nodes in their `<node-name>.xml` files, and
- Use the same values for the Name and Password elements (most often the default values) in the Cache section of their `<node-name>.xml` files. (A cluster is defined by its cache.)

In this Cache section, Name is set to the default value:

```
<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>
</Cache>
```

Solution: Configure a unique name for each cluster. To separate two clusters, give all the nodes in cluster A the same value in the Cache | Name element of their `<node-name>.xml` files —cluster_A, for example. All the nodes in cluster B must likewise share a Cache | Name value—cluster_B, for example. It can be anything but cluster_A. It is best to also assign a unique Cache | Password value to each cluster, and to ensure that the lists of manager nodes in the Managers section do no overlap across clusters. (The Managers section should include the same manager nodes in every member of a given cluster.)

See also

- *Configuring a Cluster* on page 17

A Project Triggers Java Out-of-Memory Errors

Problem: When a project runs, you see out-of-memory errors from the Java virtual machine.

Solution: Modify the project configuration (CCR) file to increase the heap size for the project's Java virtual machine.

See also

- *Memory Usage* on page 104

A Legacy Project Fails to Compile or Run

Problem: A legacy project does not compile or does not run. You might see an error saying there is no such adapter as platform_in or platform_out.

As of Event Stream Processor version 5.1.04, the Platform Input and Platform Output adapters are no longer installed.

Solution: If your project uses platform_in or platform_out to create stream or window bindings in the CCL file, you must remove the bindings from the CCL and recreate them in the CCR file.

1. Remove ATTACH ADAPTER statements for the deprecated adapters from the project:
   - `ATTACH INPUT ADAPTER platform_in`
   - `ATTACH OUTPUT ADAPTER platform_out`

   You can do this in Studio using either the visual editor (right-click the adapter element and select **Delete**) or the CCL text editor (delete the statement). The default location of the CCL file on the Studio machine is `<user's-home-dir>/SybaseESP/5.1/workspace/<project-name>/<project-name>.ccl`.
2. Recreate each binding in the project configuration (CCR) file. See the *Studio Users Guide* for information on editing the project configuration. The default location of the CCR file on the Studio machine is `<user's-home-dir>/SybaseESP/5.1/workspace/<project-name>/<project-name>.ccr`.
3. If your old CCL file references adapter parameters that do not appear on the Bindings tab of the Project Configuration File Editor in Studio, call technical support to report the problem.

Published Data Lost When Node Fails

Problem: A subscriber receiving data through a stream does not receive all the data sent by the publisher before the ESP node shuts down unexpectedly.

Solution 1: Provide the Subscriber with a Window with a Long Retention Time

Replace the stream feeding the subscriber with a window configured with a retention time long enough to allow the subscriber to reconnect when the node comes back up or (if the node is in active-active HA mode) fails over to its secondary instance. For example, if it takes the subscriber a minute to reconnect, you might configure a 2-minute retention time to ensure that no data is lost.

Note: The subscriber must filter out any delete operations coming from the window and view only inserts.

See the *Programmers Guide* or the *Studio Users Guide* for information on data retention policies for windows.

Solution 2: Provide the Subscriber with a Guaranteed Delivery Window

Replace the stream feeding the subscriber with a window on which guaranteed delivery (GD) is enabled.

Guaranteed delivery (GD) uses log stores to ensure that a GD subscriber registered with a GD window receives all the data processed by that window even if the client is not connected when the data is produced. GD is supported only on windows (not on streams or delta streams) and each GD window requires a log store.

Note: SAP does not recommend using GD-enabled windows on a node configured for active-active HA mode. The shared disk requirements for GD log stores are not compatible with the continuous synchronization that enables an active-active primary instance to fail over quickly to its secondary instance.

For a window assigned to a memory store, a stream, or a delta stream, consider using persistent subscribe pattern (PSP) instead of GD.

See the *Programmers Guide* or the *Studio Users Guide* for information on guaranteed delivery. See the *Studio Users Guide* for information on PSP.

A Project Fails Repeatedly

Problem: One or more projects in a cluster repeatedly shut down unexpectedly.

Solution: Consider increasing the heartbeat timeout value the cluster uses to ensure that projects are running. Note, however, that problems related to the heartbeat timeout setting are unusual; it should rarely be necessary to change it. The default project heartbeat timeout is 20000 milliseconds (20 seconds).

1. To confirm that the heartbeat timeout is the problem, check the project log: `ESP_HOME/cluster/projects/<cluster-name>/<workspacename>.<project-name>.<instance-number>/esp_server.log`

   Or in a Studio cluster: `<user's-home-dir>/SybaseESP/5.1/workspace/<workspacename>.<project-name>.<instance-number>/esp_server.log`).

   Look for a 722014 message that includes a last contact delta value. The message looks similar to this:

   ```
    2013-02-22 01:20:55.036 | 12611 | container | [SP-2-722014]
   (5741.829)
        sp(12589) Manager.heartbeatApplication() asked to stop. Last
   contact delta=20568
   ```

   The delta value is the time in milliseconds between the final contacts between the project and the cluster. If the delta value is close to or larger than the heartbeat timeout value, try increasing the heartbeat timeout value.

2. In an editor, open the node configuration file, `ESP_HOME/cluster/nodes/<nodename>/<node-name>.xml`.
3. Replace the Manager element with this code:

   ```
   <Manager enabled="true">
     <!-- The ApplicationHeartbeatTimeout node is optional -->
     <!-- The first Manager in the cluster determines the value 
   cluster-wide -->
     <!-- The value is in milliseconds -->
     <ApplicationHeartbeatTimeout>20000</
   ApplicationHeartbeatTimeout>
   </Manager>
   ```

   Note: This is the top-level Manager element, not a Manager element in the Cache | Managers section.
4. Replace the default value of ApplicationHeartbeatTimeout, 20000, with a value larger than the last contact delta found in your log. For example, to increase the timeout to 30 seconds, enter 30000.
5. Copy the new Manager section into the `<node-name>.xml` file for every manager node in the cluster.
6. Stop and restart the cluster, shutting down controller-only nodes first, then manager nodes, and starting all the manager nodes before the controller-only nodes.

See also

- *Clustering Architecture* on page 1
- *Stopping a Node or Cluster* on page 86
- *Starting a Node or Cluster* on page 85

Index