SAP Event Stream Processor: Configuration and Administration Guide
# Table of Contents

## 1 Introduction

1.1 Tools for Administration and Monitoring ........................................... 4

## 2 Managing your Cluster

2.1 Clustering Architecture ........................................................................... 5
2.2 File and Directory Infrastructure ........................................................ 7
2.3 Get Started with a Cluster ...................................................................... 14
   2.3.1 Planning a Cluster .......................................................................... 14
   2.3.2 High Availability ........................................................................... 15
   2.3.3 Centralized Security ....................................................................... 19
   2.3.4 Cluster Configuration ..................................................................... 19
2.4 Deploying a Project to a Cluster ........................................................... 31
   2.4.1 Deploying a Project to a Remote Cluster from Studio .................... 32
   2.4.2 Project Deployment Options .......................................................... 33
   2.4.3 Active-Active Deployments ............................................................ 37
   2.4.4 Project Affinities ........................................................................... 38
   2.4.5 Processor Affinity ......................................................................... 39
   2.4.6 Cold Failover ................................................................................ 39
   2.4.7 Project Instances ........................................................................... 40
   2.4.8 Project Configurations ................................................................. 40
   2.4.9 Sample Project Configuration File ................................................ 64
   2.4.10 Accessing Project Contents using SAP HANA Smart Data Access ... 67
2.5 Deploying an Adapter to a Cluster ......................................................... 67
   2.5.1 Application Deployment Parameters ............................................. 68
   2.5.2 Sample Application Deployment Configuration File ....................... 70
   2.5.3 Adapter Cold Failover ................................................................... 71
   2.5.4 Adapter Affinity ............................................................................ 71
2.6 Administer a Cluster ............................................................................... 71
   2.6.1 Starting a Node or Cluster ............................................................ 71
   2.6.2 Stopping a Node or Cluster ........................................................... 73
   2.6.3 Logging ......................................................................................... 74
   2.6.4 Cluster Administrative Tool ........................................................ 78
   2.6.5 Safeguarding Your Data ............................................................... 82
   2.6.6 Memory Usage ............................................................................. 86
2.7 Monitor a Cluster ................................................................................... 87
   2.7.1 Monitoring a Project ....................................................................... 87
   2.7.2 Monitoring with SAP ESP Cockpit ................................................. 89
   2.7.3 Monitoring with Metadata Streams .............................................. 90

## 3 Configuring External Database Access .................................................. 103
3.1 Discovering a Service. .............................................................. 104
3.2 Loading a Data Service. .......................................................... 105
3.3 Importing a Data Service. ........................................................ 106
3.4 Adding a JDBC Connection to an External Database. ......................... 106
3.5 Adding an ODBC Connection to an External Database. ......................... 108
3.6 Adding an OCS Connection to an External Database. ......................... 109
3.7 Updating a Data Service. .......................................................... 110
3.8 Copying a Data Service. .......................................................... 111
3.9 Deleting a Data Service. .......................................................... 111
3.10 Configuring to Support HANA Failover. ........................................ 112
3.11 Set up an ODBC Driver Manager. ............................................... 113

4 Configuring ESP Server Processes to Run as Windows Services. ............... 114

5 Troubleshooting. .................................................................. 116
5.1 Login and Connection Problems. ............................................... 116
  5.1.1 Error: Invalid Login Credentials. ........................................... 116
  5.1.2 Prompted for Local Cluster Password. ....................................... 117
  5.1.3 Cannot Connect to the Cluster. ................................................... 117
  5.1.4 A Studio Project Does Not Run, Reports Login Failure. ................. 118
  5.1.5 A Utility Fails to Connect to the Server. ....................................... 118
  5.1.6 A Utility Fails to Connect to a Project. ........................................ 119
  5.1.7 Cannot Start ESP Cockpit Due to Configuration Errors. ................. 119
  5.1.8 An External Adapter Fails to Start. ............................................ 120
  5.1.9 An Adapter Fails to Connect to a Project. ..................................... 121
  5.2 Project Problems. ................................................................. 121
  5.2.1 A Project Fails to Restart. ....................................................... 121
  5.2.2 A Studio Project Does Not Run, Reports Login Failure. ................. 122
  5.2.3 A Utility Fails to Connect to a Project. ........................................ 122
  5.2.4 A Project Runs in the Wrong Cluster. .......................................... 122
  5.2.5 A Project Triggers Java Out-of-Memory Errors. ............................... 123
  5.2.6 A Legacy Project Fails to Compile or Run. .................................... 123
  5.2.7 Published Data Lost When Node Fails. ......................................... 124
  5.2.8 Cannot Publish or Subscribe to a Project that Seems to be Running. ... 125
  5.2.9 A Project Fails Repeatedly. ...................................................... 125
1 Introduction

The SAP Event Stream Processor: Configuration and Administration Guide describes tasks and concepts for managing SAP® Event Stream Processor.

1.1 Tools for Administration and Monitoring

Use the SAP ESP Cockpit to administer and monitor SAP Event Stream Processor.

The SAP ESP Cockpit is a Web-based tool for managing smart data streaming cluster nodes, projects, adapters, and other components. For more information, see the SAP Event Stream Processor: Cockpit Guide.
2 Managing your Cluster

An SAP Event Stream Processor cluster consists of one or more nodes—typically there is one node per host. Cluster nodes manage the ESP projects and adapters running on the ESP cluster. Clustering lets you run multiple projects simultaneously. The ESP clusters promote failure recovery and data redundancy.

2.1 Clustering Architecture

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

Manager Nodes and Controller Nodes

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.
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 restarts any failed projects on which cold failover is enabled. 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.

**Heartbeats and Shutting Down Stalled Projects**

Every deployed project broadcasts a heartbeat to 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 cold failover configured, the manager restarts the project.) For example, if your CPU utilization is at 100 percent, the project server may not be able to send heartbeats to the cluster manager, so the cluster manager stops the project. In multinode clusters, the manager responsible for monitoring a project might not be the manager through which the project is deployed.

**Shared Caches**

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.
Cluster Database

Prior to release ESP 5.1 SP08, the shared cache was file-based. Starting in ESP 5.1 SP08, this shared cache is stored in the cluster database. The cluster database maintains configuration information for all nodes in the cluster. It also holds policy configuration, data services (formerly stored in the service.xml file), and persists cluster cache data. There is one logical instance of the database per cluster. The cluster database lifecycle is distinct and separate from that of the cluster. If you set up the cluster database as a service, it can overlap or be completely separate.

Workspaces: Cluster and Studio

When you deploy a project, you assign it to a cluster workspace: a named, runtime, server-side construct that lets you group related projects, adapters, and data services and manage their permissions together. Because many of the objects you create and work with in ESP are accessed through cluster workspaces, cluster workspaces come up often—most references to workspaces in the documentation are to cluster workspaces.

Cluster workspaces are distinct from the Studio workspace, which is a single-user design-time location and a feature of the Eclipse platform on which ESP is built. The Studio workspace is a directory that stores your developing projects or pointers to them. See Eclipse documentation for more information.

2.2 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 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>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<adapter-base-directory>` | Parent of the working directories for adapter processes. The names of adapter working directories take this form:  
<p>|                       | <code>&lt;workspace-name&gt;.&lt;adapter-name&gt;.0</code>                                                             |
|                       | See also workspace, cluster [page 13] and project working directory [page 12].                  |
| Needs shared drive?   | Yes                                                                                             |
| Default Location      | Remote cluster: STREAMING_SHARED/cluster/adapters/ <code>&lt;cluster-name&gt;</code>                           |
|                       | The <code>&lt;adapter-base-directory&gt;</code> is defined in cluster configuration through ESP Cockpit.         |
|                       | Local cluster: <code>&lt;user's-home-dir&gt;/SybaseESP/5.1/workspace/adapters</code>                            |
|                       | You set the location of the workspace directory during installation. In Windows, <code>&lt;user's-home-dir&gt;</code> defaults to the My Documents folder. |
|                       | The <code>&lt;adapter-base-directory&gt;</code> for Studio is defined in Studio’s cluster configuration file, STREAMING_HOME/studio/clustercfg/studio.xml. |
| Base directory        | See <code>&lt;adapter-base-directory&gt;</code> [page 8] and <code>&lt;project-base-directory&gt;</code> [page 11].                |
| cluster.db            | The cluster database, which stores node and policy configurations and persistent data from the cluster cache. For details, see Cluster Configuration [page 19]. |
| Needs shared drive?   | No                                                                                              |
| Default Location      | STREAMING_HOME/cluster/config/&lt;cluster-name&gt;                                                    |
| cluster.key           | The cluster key file, which is used to encrypt SSL files and passwords in cluster configuration. |
| Needs shared drive?   | For security purposes, SAP does not recommend storing <code>cluster.key</code> on a shared drive. SAP recommends permissions settings that allow only the user who launches the node to access this file. |
| Default Location      | STREAMING_HOME/cluster/keys/&lt;cluster-name&gt;                                                       |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster.log</td>
<td>Cluster node log. Captures node-level and cluster-level errors and event messages; each node's log is unique to that node.</td>
</tr>
<tr>
<td></td>
<td><strong>Needs shared drive?</strong> Must not be shared</td>
</tr>
<tr>
<td></td>
<td><strong>Default Location</strong> STREAMING_HOME/cluster/config/&lt;cluster-name&gt;</td>
</tr>
<tr>
<td>cluster.log.properties</td>
<td>Cluster node log configuration file. cluster.log.properties is a log4j properties file.</td>
</tr>
<tr>
<td></td>
<td><strong>Needs shared drive?</strong> No</td>
</tr>
<tr>
<td></td>
<td><strong>Default Location</strong> STREAMING_HOME/cluster/config/&lt;cluster-name&gt;</td>
</tr>
<tr>
<td>STREAMING_HOME</td>
<td>Represents the Event Stream Processor installation directory. STREAMING_HOME is also an environment variable and a configuration file macro.</td>
</tr>
<tr>
<td></td>
<td><strong>Needs shared drive?</strong> Must not be shared</td>
</tr>
<tr>
<td></td>
<td><strong>Default Location</strong> Windows:C:\Sybase\ESP-5_1</td>
</tr>
<tr>
<td></td>
<td>UNIX:/opt/sybase/ESP-5_1</td>
</tr>
<tr>
<td>esp_metadata</td>
<td>Metadata directory that holds the metadata log store for the project if you are using the guaranteed delivery feature.</td>
</tr>
<tr>
<td></td>
<td><strong>Needs shared drive?</strong> Yes</td>
</tr>
<tr>
<td></td>
<td><strong>Default Location</strong> &lt;base-directory&gt;/&lt;workspace-name&gt;.&lt;project-name&gt;.&lt;instance-number&gt;/esp_metadata</td>
</tr>
<tr>
<td>STREAMING_SHARED</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td><strong>Needs shared drive?</strong> Yes</td>
</tr>
<tr>
<td></td>
<td><strong>Default Location</strong> STREAMING_HOME</td>
</tr>
<tr>
<td>esp_server.log</td>
<td>Project log. Captures errors and events in a running project. Resides in the working directory for the project.</td>
</tr>
<tr>
<td></td>
<td><strong>Needs shared drive?</strong> Optional; typically shared in a multinode cluster where the base directory is shared</td>
</tr>
<tr>
<td></td>
<td><strong>Location</strong> &lt;project-base-directory&gt;/&lt;workspace-name&gt;.&lt;project-name&gt;.&lt;instance-number&gt;/esp_server.log</td>
</tr>
<tr>
<td>examples</td>
<td>Examples directory. Contains the example configuration of a four node cluster.</td>
</tr>
<tr>
<td></td>
<td><strong>Needs shared drive?</strong> No</td>
</tr>
<tr>
<td></td>
<td><strong>Location</strong> STREAMING_HOME/cluster/examples</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Input files for projects</td>
<td>Anything a project needs at runtime—for example, input CSV files, ODBC drivers, or queuing 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: <code>&lt;project-base-directory&gt;/workspace-name&gt;.&lt;project-name&gt;.&lt;instance-number&gt;</code></td>
</tr>
<tr>
<td>Log store</td>
<td>Log store. Saves project input and output data in case of failover. Needs shared drive? Yes Default Location: <code>&lt;project-base-directory&gt;/workspace-name&gt;.&lt;project-name&gt;.&lt;instance-number&gt;/logstore-name</code></td>
</tr>
<tr>
<td>Node working directory</td>
<td>Directory from which the node is started. cluster.log.properties must reside in this directory; cluster.log and cluster.db, the cluster database, reside here by default. Not to be confused with the project working directory [page 12]. Needs shared drive? No Default Location: STREAMING_HOME/cluster/config/&lt;cluster-name&gt;</td>
</tr>
<tr>
<td><code>&lt;persistence-directory&gt;</code></td>
<td>Persistence directory. Saves project and workspace information when a node shuts down. Sets the persistence directory’s path in cluster configuration. All managers in a cluster must use the same persistence directory. Needs shared drive? Yes Default Location: STREAMING_SHARED/storage</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>&lt;project-base-directory&gt;</code></td>
<td>Parent of the working directories for project processes. Also holds cluster workspaces, project instances, and log stores. The names of project working directories take this form:</td>
</tr>
<tr>
<td></td>
<td>where <code>&lt;instance-number&gt;</code> can be 0 or 1 in an HA (dual-instance) project and is always 0 in a non-HA project.</td>
</tr>
<tr>
<td></td>
<td>See also workspace, cluster [page 13] and project working directory [page 12].</td>
</tr>
<tr>
<td>Needs shared drive?</td>
<td>Yes</td>
</tr>
<tr>
<td>Default Location</td>
<td>Remote cluster: STREAMING_SHARED/cluster/projects/ <code>&lt;cluster-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td>The <code>&lt;project-base-directory&gt;</code> is defined in cluster configuration through ESP Cockpit.</td>
</tr>
<tr>
<td></td>
<td>Local cluster: <code>&lt;user's-home-dir&gt;/SybaseESP/5.1/workspace/applications</code></td>
</tr>
<tr>
<td></td>
<td>You set the location of the workspace directory during installation. In Windows, <code>&lt;user's-home-dir&gt;</code> defaults to the <code>My Documents</code> folder.</td>
</tr>
<tr>
<td></td>
<td>The <code>&lt;project-base-directory&gt;</code> for Studio is defined in Studio’s cluster configuration file, <code>STREAMING_HOME/studio/clustercfg/studio.xml</code>.</td>
</tr>
<tr>
<td><code>&lt;project-name&gt;.ccl</code></td>
<td>Definition of a project in Continuous Computation Language. A project has one <code>&lt;project-name&gt;.ccl</code> file. When you edit the project in SAP Event Stream Processor Studio, you modify the CCL or an ancillary file. (The ancillary files store information about the visual presentation of the project in Studio.)</td>
</tr>
<tr>
<td>Needs shared drive?</td>
<td>N/A</td>
</tr>
<tr>
<td>Default Location</td>
<td>Local (Studio) cluster: <code>&lt;user's-home-dir&gt;/SybaseESP/5.1/workspace/&lt;project-name&gt;/</code></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| `<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>` |
| Project working directory | Contains files generated by the project. May also contain input files the project needs. Not to be confused with workspace, cluster [page 13] or node working directory [page 10].  
**Needs shared drive?** Optional; typically shared in a multinode cluster where the base directory is shared.  
**Default Location** `<project-base-directory>/<workspace-name>.<project-name>.<instance-number>` |
| security | Security directory. Stores Java keystores for RSA and policy files for access control. All components in a cluster are subject to the security rules defined in cluster configuration. 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** STREAMING_SHARED/security |
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Needs shared drive?</th>
<th>Default Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>server.key</td>
<td>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.</td>
<td></td>
<td>STREAMING_HOME/cluster/keys/&lt;cluster-name&gt;</td>
</tr>
<tr>
<td>server.crt</td>
<td>For security purposes, SAP does not recommend storing unencrypted SSL files on a shared drive.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>server.key.enc</td>
<td>Encrypted SSL files. ESP requires a set of encrypted SSL files (by default) or a set of unencrypted SSL files, but not both.</td>
<td></td>
<td>STREAMING_HOME/cluster/keys/&lt;cluster-name&gt;</td>
</tr>
<tr>
<td>server.crt.enc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdstreams.log</td>
<td>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.</td>
<td>Optional; typically shared in a multinode cluster where the base directory is shared</td>
<td>&lt;project-base-directory&gt;/&lt;workspace-name&gt;..&lt;project-name&gt;..&lt;instance-number&gt;/stdstreams.log</td>
</tr>
<tr>
<td>workspace, cluster (also workspace, streaming)</td>
<td>A named scope (similar to a directory) to which you deploy projects and their supporting files, adapters, and data services. Cluster workspaces give you the option to manage the permissions of related objects together. Every cluster has at least one workspace, and any workspace can contain projects, supporting files, adapters, and data services from multiple nodes. Contrast with workspace, Studio [page 13]. See also &lt;project-base-directory&gt; [page 11] and project working directory [page 12].</td>
<td>A shared drive is not possible—the cluster workspace is a logical rather than a physical location.</td>
<td>None</td>
</tr>
<tr>
<td>workspace, Studio</td>
<td>Directory where Studio stores project files. You set the location during installation. Contrast with workspace, cluster [page 13].</td>
<td>No</td>
<td>&lt;user's-home-dir&gt;/SybaseESP/5.1/workspace In Windows, &lt;user's-home-dir&gt; defaults to the My Documents folder.</td>
</tr>
</tbody>
</table>

In Windows, `<user's-home-dir>` defaults to the `My Documents` folder.
2.3  Get Started with a Cluster

Plan and configure an SAP Event Stream Processor cluster.

1. Planning a Cluster [page 14]
   Plan the number of manager and controller nodes for the cluster, and follow best practices for multiple node configuration.

2. High Availability [page 15]
   SAP Event Stream Processor supports a set of high availability features that promote failure recovery and data redundancy.

3. Centralized Security [page 19]
   Security options for Event Stream Processor are configured locally through the cluster.

4. Cluster Configuration [page 19]
   The cluster database maintains configuration information for all nodes in the cluster. It also holds policy configuration, and persists cluster cache data. There is one logical instance of the database per cluster.

2.3.1  Planning a Cluster

Plan the number of manager and controller nodes for the cluster, and follow 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.
- The number of projects a controller can support is partially dependent on machine size.
- 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 for cluster configuration through ESP Cockpit. See Configuring a Node [page 30] for more information. Node names should only contain numbers, letters, and underscores.
● 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 `STREAMING_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.

2.3.2 High Availability

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Where Set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters</td>
<td>Cluster configuration</td>
<td>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.)</td>
</tr>
<tr>
<td>Cold Failover</td>
<td>Project configuration</td>
<td>In cold failover, each 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.</td>
</tr>
<tr>
<td>Feature</td>
<td>Where Set</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Active-Active Mode</td>
<td>Project configuration</td>
<td>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.</td>
</tr>
<tr>
<td>Zero Data Loss</td>
<td>Project configuration</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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 stream or window receives all the data processed by that stream or window even if the client is not connected when the data is produced. GD is supported on streams and windows (not on delta streams) and each GD stream or window requires a log store.                                                                                                                                 |
<p>| Consistent recovery   | Project configuration | The consistent recovery feature can restore all the streams and 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. |</p>
<table>
<thead>
<tr>
<th>Feature</th>
<th>Where Set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto checkpoint</td>
<td>Project configuration</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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 delta stream.  
- 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. |

### 2.3.2.1 Configuring Cluster Database for High Availability

ESP uses a SQL Anywhere database for persistence for configurations and security policy. Set this cluster database in a mirroring configuration to provide high availability (HA).

#### Context

The cluster database maintains configuration information for all nodes in the cluster. It also holds policy configuration, data services (stored in the `service.xml` file in versions earlier than ESP 5.1 SP04), and persists cluster cache data. There is one logical instance of the database per cluster. The cluster database lifecycle is distinct and separate from that of the cluster.

The following steps show how to use database mirroring with a single installation of SQL Anywhere on a host. In practice, SAP recommends using three different hosts, each running a SQL Anywhere instance. These hosts are distinct and separate from the ESP boxes.
For more information on database mirrors, see the *Database Mirroring* section in the *SQL Anywhere* documentation.

**Procedure**

1. Create two new folders in `STREAMING_HOME/sqla` named `esp_db2` and `esp_db3`.
2. Start the SQL Anywhere server:
   ```
   STREAMING_HOME/sqla/Bin64> dbsrv16 -n mirror_server1 -x "tcpip(PORT=6871)" -su sql "STREAMING_HOME\sqla\esp_db\espdb.db" -xp on
   ```
   **Note**
   *STREAMING_HOME* refers to the environment variable. Write this as `%STREAMING_HOME%` in Windows and `$STREAMING_HOME` in Unix.
3. Start the `dbisql` tool:
   ```
   STREAMING_HOME/sqla/Bin64> dbisql -c "UID=DBA;PWD=SQL;SERVER=mirror_server1"
   ```
4. Execute the following commands in the `dbisql` tool:
   ```
   CREATE MIRROR SERVER mirror_server1
   AS PARTNER
   connection_string='SERVER=mirror_server1;host=localhost:6871'
   state_file='C:\Sybase\ESP-5_1\sqla\esp_db\server1.state';
   CREATE MIRROR SERVER mirror_server2
   AS PARTNER
   connection_string='SERVER=mirror_server2;host=localhost:6872'
   state_file='C:\Sybase\ESP-5_1\sqla\esp_db2\server2.state';
   CREATE MIRROR SERVER mirror_primary
   AS PRIMARY
   connection_string='SERVER=mirror_primary;host=localhost:6871,localhost:6982'
   CREATE MIRROR SERVER mirror_mirror
   AS MIRROR
   connection_string='SERVER=mirror_mirror;host=localhost:6871,localhost:6982'
   CREATE MIRROR SERVER esp_arbiter
   AS ARBITER
   connection_string='SERVER=esp_arbiter;HOST=localhost:6870';
   SET MIRROR OPTION authentication_string='abc';
   ```
5. From the command line:
   a. Copy database files from the first server:
      ```
      STREAMING_HOME/sqla/Bin64> dbbackup -c "UID=DBA;PWD=SQL;SERVER=mirror_server1;DBN=db" C:\Sybase\ESP-5_1\sqla\esp_db2
      ```
   b. Start the second server:
      ```
      STREAMING_HOME/sqla/Bin64>dbsrv16 -n mirror_server2 -x "tcpip(PORT=6872;DOROBROAD=no)" -su sql "C:\Sybase\ESP-5_1\sqla \esp_db2\espdb.db" -xp on
      ```
c. Start the third server as arbiter:

```
STREAMING_HOME/sqla/Bin64>dbsrv16 -n esp_arbiter -su sql -x
"tcpip(PORT=6870;DOBROAD=no" -xf "C:\Sybase\ESP-5_1\sqla
\esp_db3\arbiter.state" -xa "AUTH=abc;DBN=db"
```

### 2.3.3 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 cluster configuration through ESP Cockpit. See Configuring a Cluster [page 29] for more information. Event Stream Processor supports Kerberos, RSA, SAP BI, Pre-Configured User, native OS, SAP HANA, 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.

### 2.3.4 Cluster Configuration

The cluster database maintains configuration information for all nodes in the cluster. It also holds policy configuration, and persists cluster cache data. There is one logical instance of the database per cluster.

The ESP installer creates a cluster database during installation, and prompts you to enter a user name and password for this database. This cluster database initially contains the configuration for a single ESP node. Because the database holds the configuration for a cluster, changes in database performance or function will directly impact the cluster. For this reason, consider a high availability configuration for the database.

A cluster must be running before you can deploy projects to it. To start a cluster, start manager nodes first, then controller-only nodes, using the `streamingclusternode` utility. 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. You can also use `streamingclusternode` to deploy and retrieve cluster configuration files. For more information on `streamingclusternode` and its cluster options, see the SAP Event Stream Processor: Utilities Guide.
2.3.4.1 Role of the Installer in Cluster Configuration

ESP uses a database-based cluster configuration, rather than a file-based cluster configuration. This database hosts configuration information for the cluster and is known as the cluster database. Several necessary components for the database-based configuration are installed during installation.

Cluster Database

- When the ESP Server is installed, the installer creates the cluster database.
- The installer also deploys an initial cluster configuration consisting of a single node.

Passwords

- The installer prompts you for the cluster database user name and password. The credentials you define here do not have to belong to an actual user, but they are required for administrative-level access to this database.
- The installer prompts you for the cluster password – this is also used as the password for the SYS_STREAMING user which is used when logging into ESP Cockpit immediately after installing ESP.

File and Script Updates

- The installer updates settings in the cluster bootstrap file (cluster.cfg).
- The installer updates the script used to start the cluster database.
- The installer updates the script used to start the initial node.

2.3.4.2 Creating a Cluster Database

The ESP installer creates a cluster database during installation, and prompts you to enter a user name and password for this database. The cluster database created by the installer contains a single node. If you wish, you can also create your own cluster database, by following the cluster example under STREAMING_HOME\cluster\examples. However if you create a cluster database, you will need to deploy it, using streamingclusternode.

Procedure

1. Create the cluster database by running the following command and providing your own values for the parameters in brackets.
This step is required if you want to create a new database rather than using the database file created by the installer.

```
dbinit -dba <username>,<password> <database_file> -t <transaction_log_file>
```

where
- `<username>` is the initial administrator user name.
- `<password>` is the initial administrator user password.
- `<database_file>` is the name of the database file to be created.
- `<transaction_log_file>` is the name of the transaction log file for database.

For example:
```
dbinit -dba DBA,sql cluster_example.db -t cluster_example.log
```

2. Start the cluster database by running the following command, and providing your own values for the parameters in brackets.

```
dbsrv16 -n <server_name> <database_file> -n <database_name>
```

where
- `<server_name>` is the name of the database server.
- `<database_file>` is the name of the database file.
- `<database_name>` is the database name.

For example:
```
dbsrv16 -n cluster_example_myhost cluster_example.db -n cluster_example
```

This starts the database with a server name of `cluster_example_myhost` and a database name of `cluster_example`.

3. Create tables
   By following the `cluster_example` and making your own modifications, you have created a cluster database. You can now deploy or migrate cluster configuration data to the database, and then start the node. From there, you can use SAP ESP Cockpit to manage cluster configuration.

### 2.3.4.2.1 Using the Cluster Configuration Example

Deploy the example cluster configuration file `cluster_example.xml` to the cluster database and start the node.

**Procedure**

1. Start the cluster database by running the following command, providing your own values for the parameters in brackets.

```
dbsrv16 -n <server_name> <database_file>
```

where
1. <server name> is the name of your choice for the database server.
   <database_file> is the name of the database file.
   
   For example:
   
   dbsrvl6 -n cluster_example_myhost cluster_example.db
   
   This starts the database with a server name of cluster_example_myhost.

2. Modify the cluster bootstrap file cluster_example.cfg to provide database and cluster-related information.
   
   The following information is required:
   
   - jdbc-username - the database user name
   - jdbc-password - the database password
   - cluster-name - name of the cluster name
   - cluster-password - cluster password
   
   If the database is not running on the default port (2638), or on the local cluster, also modify the jdbc-url property by replacing the host and/or port value.

3. Change the default user password in the cluster configuration file cluster_example.xml. In the <Security> element, change the password option of the PreConfiguredUserLoginModule authenticator.
   
   a. To generate a password value in encoded form for this field, use the following command:

      ```
      $STREAMING_HOME/bin/streamingclusteradmin --encode_text
      Please enter text to be encoded:
      Please re-enter text to be encoded:
      {SHA-256:jvaPGO32Jjs=}8ALhhaxfCXD/5Jpkb/p9fT0F4uNomLZWA5gNj056Hh8=
      ```

   b. Then transfer the encoded value to cluster_example.xml.

      ```
      <Option name="password">{SHA-256:jvaPGO32Jjs=}8ALhhaxfCXD/5Jpkb/p9fT0F4uNomLZWA5gNj056Hh8=/Option>
      ```

4. (Optional) To configure cluster nodes for remote access, set the <STREAMING_HOSTNAME> macro for one or more nodes in the cluster configuration file to a hostname or IP address. For example:

   ```
   <Macro type="value" name="STREAMING_HOSTNAME">abc.sap.corp</Macro>
   ```

5. Deploy the modified cluster configuration file to the database by running the following command:

   ```
   $STREAMING_HOME/bin/streamingclusternode --config cluster_example.cfg --deploy --config-type file --file cluster_example.xml
   ```

6. Start the cluster node by running the start_node script located in the cluster/examples directory. You can specify the node to start by using the node name as the first argument for the start_node script. For example:

   ```
   $STREAMING_HOME/cluster/examples/start_node node1
   ```

   The properties defined in the cluster_example.cfg bootstrap file are used when starting the node. In the cluster_example.xml file, there are four nodes defined: node1, node2, node3, and node4.
2.3.4.2.1.1 Example Cluster Configuration File

The following example cluster configuration file is located at STREAMING_HOME/cluster/examples/cluster_example.xml.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Cluster>
  <Macros>
  </Macros>
  <SystemProperties/>
  <Manager/>
  <Controller>
    <ApplicationTypes>
      <ApplicationType enabled="true" name="ha_project">
        <Class>com.sybase.esp.cluster.plugins.apptypes.HaProject</Class>
        <StandardStreamLogging enabled="true"/>
        <Properties>
          <Property name="base-directory">${STREAMING_HOME}/cluster/examples/projects</Property>
          <Property name="streaming-home">${STREAMING_HOME}</Property>
          <Property name="hostname">${STREAMING_HOSTNAME}</Property>
          <Property name="ld-preload">${STREAMING_HOME}/lib/libjsig.so</Property>
          <Property name="debug-level">4</Property>
          <Property name="enable-uds">false</Property>
        </Properties>
      </ApplicationType>
      <ApplicationType enabled="true" name="project">
        <Class>com.sybase.esp.cluster.plugins.apptypes.Project</Class>
        <StandardStreamLogging enabled="true"/>
        <Properties>
          <Property name="base-directory">${STREAMING_HOME}/cluster/examples/projects</Property>
          <Property name="streaming-home">${STREAMING_HOME}</Property>
          <Property name="hostname">${STREAMING_HOSTNAME}</Property>
          <Property name="ld-preload">${STREAMING_HOME}/lib/libjsig.so</Property>
          <Property name="debug-level">4</Property>
        </Properties>
      </ApplicationType>
      <ApplicationType enabled="true" name="toolkit_adapter">
        <Class>com.sybase.esp.cluster.plugins.apptypes.FrameworkAdapter</Class>
        <StandardStreamLogging enabled="true"/>
        <Properties>
          <Property name="streaming-home">${STREAMING_HOME}</Property>
          <Property name="base-directory">${STREAMING_HOME}/cluster/adapters</Property>
        </Properties>
      </ApplicationType>
    </ApplicationTypes>
    <ServiceProvider>
      <ServiceTypes>
        <ServiceType name="discovery">
          <Class>com.sybase.esp.cluster.plugins.servicetypes.adapter.DiscoveryServiceImpl</Class>
          <StandardStreamLog enabled="true"/>
          <Properties>
            <Property name="base-directory">${STREAMING_HOME}/cluster/examples/discovery</Property>
          </Properties>
        </ServiceType>
      </ServiceTypes>
      <ServiceTypes/>
    </ServiceProvider>
    <Controller>
      <ApplicationTypes/>
    </Controller>
</Cluster>
```
<Property name="cnxml-path">${STREAMING_HOME}/lib/adapters</Property>

<Property name="streaming-home">${STREAMING_HOME}</Property>

<Property name="hostname">${STREAMING_HOSTNAME}</Property>

</Properties>
</ServiceType>
</ServiceTypes>
</ServiceProvider>
<Rpc>
<Ssl enabled="false"/>
</Rpc>

<Persistence enabled="false">
<Directory>${STREAMING_STORAGE}</Directory>
<Limited enabled="true"/>
</Limited>
</Persistence>

<Multicast enabled="false"/>
</Cache>

<Security>
<Authenticators>
<Authenticator>
<Provider>com.sybase.security.core.PreConfiguredUserLoginModule</Provider>
<Options>
<Option name="username">sybase</Option>
<Option name="password">[Replace with encoded password from the cluster admin tool]</Option>
</Options>
</Authenticator>
</Authenticators>
<Authorizer enabled="false"/>
<KeyStore>
<Type>JKS</Type>
<File>${STREAMING_HOME}/cluster/examples/cluster_example.jks</File>
>Password encrypted="true">
JxyT0wFwXmMy9Tzc+CL3+u11hI2buKvfg0ZEia4DU1B+/Qq2
</Password>
<KeyPassword encrypted="true">
JxyT0wFwXmMy9Tzc+CL3+u11hI2buKvfg0ZEia4DU1B+/Qq2
</KeyPassword>
<Algorithm>RSA</Algorithm>
</KeyStore>
</Security>

<Nodes>
<Node enabled="true" name="node1">
<Macros>
<Macro name="STREAMING_HOSTNAME" type="value">localhost</Macro>
</Macros>
<SystemProperties/>
<Manager enabled="true"/>
<Controller enabled="true"/>
<ApplicationTypes/>
</Controller>
<ServiceProvider enabled="true"/>
</ServiceProvider>
</Rpc>
<Host>${STREAMING_HOSTNAME}</Host>
</Port>19011</Port>
</Rpc>
<Cache>
<Host>${STREAMING_HOSTNAME}</Host>
</Port>19001</Port>
</Cache>
</Node>
<Node enabled="true" name="node2">
<Macros>
<Macro name="STREAMING_HOSTNAME" type="value">localhost</Macro>
</Macros>
<SystemProperties/>
<Manager enabled="true"/>
<Controller enabled="true"/>
<ApplicationTypes/>
</Controller>
<ServiceProvider enabled="true"/>
</ServiceProvider>
</Rpc>
<Host>${STREAMING_HOSTNAME}</Host>
</Port>19011</Port>
</Rpc>
<Cache>
<Host>${STREAMING_HOSTNAME}</Host>
</Port>19001</Port>
</Cache>
</Node>
<Macros>
  <Macro name="STREAMING_HOSTNAME" type="value">localhost</Macro>
</Macros>
<SystemProperties/>
<Manager enabled="true"/>
<Controller enabled="true">
  <ApplicationTypes/>
</Controller>
<Rpc>
  <Host>${STREAMING_HOSTNAME}</Host>
  <Port>19012</Port>
</Rpc>
<Cache>
  <Host>${STREAMING_HOSTNAME}</Host>
  <Port>19002</Port>
</Cache>
</Node>
<Node enabled="true" name="node3">
  <Macros>
    <Macro name="STREAMING_HOSTNAME" type="value">localhost</Macro>
  </Macros>
  <SystemProperties/>
  <Manager enabled="false"/>
  <Controller enabled="true">
    <ApplicationTypes/>
  </Controller>
  <Rpc>
    <Host>${STREAMING_HOSTNAME}</Host>
    <Port>19013</Port>
  </Rpc>
  <Cache>
    <Host>${STREAMING_HOSTNAME}</Host>
    <Port>19003</Port>
  </Cache>
</Node>
<Node enabled="true" name="node4">
  <Macros>
    <Macro name="STREAMING_HOSTNAME" type="value">localhost</Macro>
  </Macros>
  <SystemProperties/>
  <Manager enabled="false"/>
  <Controller enabled="true">
    <ApplicationTypes/>
  </Controller>
  <Rpc>
    <Host>${STREAMING_HOSTNAME}</Host>
    <Port>19014</Port>
  </Rpc>
  <Cache>
    <Host>${STREAMING_HOSTNAME}</Host>
    <Port>19004</Port>
  </Cache>
</Node>
</Nodes>
</Cluster>
2.3.4.3 Starting a Cluster Database

The ESP installer creates a cluster database during installation. However, you must still start the database before you can deploy and start a cluster.

Procedure

1. Start the cluster database by running the following command, and providing your own values for the parameters in brackets.

   start dbsrv16 -n <server_name> <database_file> -n <database_name>

   where
   ○ <server_name> is the name of the database server.
   ○ <database_file> is the name of the database file. If you want to run the database created by the installer, set <database_file> to $STREAMING_HOME/cluster/config/ESP1/esp_cluster.db
   ○ <database_name> is the database name.

   For example:

   start dbsrv16 -n cluster_example_myhost cluster_example.db -n cluster_example

   This starts the database with a server name of cluster_example_myhost and a database name of cluster_example.

   Note

   If you need to start the database in a different port, use the -x option. For example,

   start dbsrv16 -n my_server -x "tcpip(PORT=6872)" -su sql "C:\tmp\ASA\SQLA\Bin64\db2\db.db"

2. Modify the cluster bootstrap file cluster_example.cfg to provide database- and cluster-related information.

   The following information is required:
   ○ jdbc-username - the database user name
   ○ jdbc-password - the database password
   ○ cluster-name - the cluster name, used to name the cluster
   ○ cluster-password - the cluster password

   If the database is not running on the default port (2638), or on the local cluster, also modify the jdbc-url property by replacing the host and/or port value.

3. Start the cluster node by running the start_node script. You can specify the node to start by using the node name as the first argument for the start_node script. For example:

   $STREAMING_HOME/cluster/config/esp1/start_node node1
2.3.4.4 Stopping a Cluster Database

While there is a correct order for starting and stopping nodes and the cluster database, there is no need to stop the database when the nodes are stopped.

Procedure

1. If you do need to stop the cluster database,
   a. Stop all projects.
   b. Stop the cluster nodes, including the cluster manager.
   c. Stop the cluster database.
   Once the cluster database is restarted, the cluster manager should start normally.
2. If the cluster database is down for some reason,
   a. Manually terminate the projects and the cluster manager node.
   b. Restart the cluster database.
   c. Restart the cluster manager node.
   d. Restart all projects.

2.3.4.5 Starting a Cluster

Start a cluster by starting the cluster database, deploying the cluster configuration to the cluster database, and starting a node in the cluster. Starting a cluster is a prerequisite for running projects and logging into ESP Cockpit. In a test system, where you are running projects only on the Studio local cluster, you do not have to manually start a cluster.

Procedure

1. Set the ESP environment variables.
2. Start the cluster database.
3. Deploy the cluster configuration to the database.
4. Start a node in the cluster.
2.3.4.6  Cluster Persistence, Caching, and Multicast

Cluster persistence, caching, and multicast are stored in the cluster database.

Note

All nodes in a cluster read the configuration settings in the cluster database when they start up. Be careful to start all nodes in the cluster without changing these settings.

Persistence

When persistence is enabled, ESP will record which projects and workspaces were in existence when the cluster shuts down. By default, persistence is enabled and configured to record to the cluster database. You can use file-based persistence instead by changing the Persistence Type setting to directory and specifying the directory in the Persistence Directory setting.

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 hostname and port in the Connectivity tab for the node in ESP Cockpit.

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 its Managers section of the cluster configuration 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.
### 2.3.4.7 Configuring a Cluster

Edit cluster-wide macros, system properties, and attributes.

#### Prerequisites

- You are logged in to SAP ESP Cockpit.
- Your user account has the required read and admin permissions in ESP.

#### Procedure

1. In ESP, select the **EXPLORE** workset, then select ACTIONS Configure Cluster.
2. Select **ESP Cluster** in the browser.
3. Select the **Macros** tab to edit cluster-wide macros.
   a. To create a new macro, click the **Create Node Macro** button.
   b. Set the `<name>`, `<type>`, and `<value>` of the macro.
   c. Click **Apply**.
4. Select the **System Properties** tab to edit cluster-wide properties.
   a. To create a new property, click the **Create Node Property** button.
   b. Set the parameters of the property:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specify the name of the property.</td>
</tr>
<tr>
<td>Value</td>
<td>Specify the value of the property.</td>
</tr>
<tr>
<td>Expanded</td>
<td>Specify if the property uses expansion.</td>
</tr>
<tr>
<td>Encrypted</td>
<td>Specify if the property uses encryption.</td>
</tr>
</tbody>
</table>
   c. Click **Apply**.
5. Select the **Attributes** tab to edit the settings for the cluster:

   **Note**
   To see all available attributes, click **Show Advanced Setting**.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td>Specify whether the connection uses SSL.</td>
</tr>
<tr>
<td>Application Heartbeat Interval</td>
<td>Specify the time, in milliseconds, between a manager node’s heartbeat checks of a project for ESP.</td>
</tr>
</tbody>
</table>
### Attribute | Description
--- | ---
Multicast Manager | Specify whether this cluster uses multicast. If this option is not selected, manager nodes use direct connections.
Multicast Group | (If Multicast Manager is enabled) Specify the multicast group.
Multicast Port | (If Multicast Manager is enabled) Specify the multicast port.
Cluster Persistence | Specify whether this cluster uses persistence.
Persistence Type | (If Cluster Persistence is enabled) Specify the persistence type. Valid options are directory and database.
Persistence Directory | (If Cluster Persistence is enabled) Specify the path to the persistence directory.
Limit Persistence | Select this to enable limited persistence.
Limit Data Service Persistence | (If Limit Persistence is enabled) Select this to disable data service persistence.
Limit Application | (If Limit Persistence is enabled) Select this to disable application persistence.

a. Click **Apply**.

### 2.3.4.8 Configuring a Node

Edit parameters of an existing ESP node.

**Prerequisites**

- You are logged in to SAP ESP Cockpit.
- Your user account has the required read and admin permissions in ESP.

**Procedure**

1. In ESP, select the **EXPLORE** workset, then select ![Actions](Actions.png) **Configure Cluster**
2. Select the node you want to configure.
3. Select the **Macros** tab to view inherited cluster macros and node-specific macros.
   a. To create a new macro, click **Create Node Macro**.
   b. (Optional) To remove a node macro, select the node macro name and click **Remove Node Macro**.
   c. Set the `<name>`, `<type>`, and `<value>` of the macro.
   d. Click **Apply**.
4. Select the **System Properties** tab to view inherited cluster system properties and node-specific properties.
   a. To create a new property, click **Create Node Property**.
   b. (Optional) To remove an existing property, select the node system property name and click **Remove Node Property**.
   c. Set the parameters of the property:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specify the name of the property.</td>
</tr>
<tr>
<td>Value</td>
<td>Specify the value of the property.</td>
</tr>
<tr>
<td>Expanded</td>
<td>Specify if the property uses expansion.</td>
</tr>
<tr>
<td>Encrypted</td>
<td>Specify if the property uses encryption.</td>
</tr>
</tbody>
</table>

   d. Click **Apply**.

5. Select the **Connectivity** tab to edit connection settings:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Host Name</td>
<td>Specify the host name of the server.</td>
</tr>
<tr>
<td>Connection Port</td>
<td>Specify the connection port for the server.</td>
</tr>
<tr>
<td>Cache Host Name</td>
<td>Specify the host name of the cluster cache.</td>
</tr>
<tr>
<td>Cache Port</td>
<td>Specify the port for the cluster cache.</td>
</tr>
<tr>
<td>SSL</td>
<td>Specify whether or not the connection uses SSL.</td>
</tr>
</tbody>
</table>

   a. Click **Apply**.

6. Select the **Node Attributes** tab to edit the settings for the node:
   a. Specify whether the node is a manager and/or controller.
   b. If the node is a controller, select applications for it to control.
   c. If the node is a service provider, select **Service provider**. Available services are compiler and discovery.
   d. (Optional) To edit the default values for compiler and/or discovery, go to the left pane and expand the **Services** folder, then select a service and edit the values in the **Value** column.
   e. Click **Apply**.

### 2.4 Deploying a Project to a Cluster

To run the projects you have created, deploy them to a cluster.

### Context

You can deploy projects to a cluster using SAP ESP Cockpit, Studio, or the `streamingclusteradmin` command line utility.
A cluster must be running before you can deploy projects to it.

**Procedure**

1. Set project options using the Project Configuration view in 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:

   ```shell
   streamingclusteradmin --uri=esp[s]://localhost: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 `streamingclusteradmin` tool, Event Stream Processor prompts you for the password and hides it as you type, which improves security.

3. (Optional) Start the deployed project.

   ```shell
   streamingclusteradmin --uri=esp[s]://localhost:19011 --username=sybase --password=sybase --start_project --workspace-name=<workspace-name> --project-name=<project-name>
   ```

2.4.1 **Deploying a Project to a Remote Cluster from Studio**

The SAP Event Stream Processor Studio provides a graphic interface for connecting to a server.

**Context**

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

Procedure

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

2. If you do not see the Server View window, select **Window > Show View > Server View** while in the SAP ESP Run-Test perspective.

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

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.

5. Right-click on the target workspace and select **Load Project(s) into Workspace**.

### 2.4.2 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, 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 Configurations view in Studio.

<table>
<thead>
<tr>
<th>Project Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 |

*Note*: When you change options in a deployed project, use ESP Cockpit, Studio, or `streamingclusteradmin` to stop and remove the project from the node, then redeploy (add) the project.
<table>
<thead>
<tr>
<th>Project Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Monitor</td>
<td>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.</td>
</tr>
<tr>
<td>Java Classpath</td>
<td>Sets the Java classpath. Value is the path to the classpath file.</td>
</tr>
<tr>
<td>Java Max Heap (java-max-heap)</td>
<td>Sets the max Java heap for the project. Default value is 256 megabytes.</td>
</tr>
<tr>
<td>Bad Record File (bad-record-file)</td>
<td>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.</td>
</tr>
<tr>
<td>Utf8</td>
<td>Enables Utf8 functionality on the server. Default value is true; set to false to disable.</td>
</tr>
<tr>
<td>Web Service Enabled</td>
<td>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.</td>
</tr>
<tr>
<td>Optimize</td>
<td>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.</td>
</tr>
<tr>
<td>On Error Discard Record</td>
<td>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.</td>
</tr>
<tr>
<td>On Error Log</td>
<td>If set to true, any computation errors that occur will be logged in the error message. The default value is true.</td>
</tr>
<tr>
<td>Project Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Maximum Log File Size</td>
<td>Sets the maximum size of a log file before a new log file is started. The default setting is 1MB. Setting this parameter to “0” will enable an unrestricted/unlimited size for a single log file.</td>
</tr>
<tr>
<td>Note</td>
<td>Setting a higher file size is beneficial for projects that involve higher logging levels and retrieving any useful information. However, it is not recommended to have a higher limit for a prolonged period, as larger files will inevitably lead to performance and latency issues.</td>
</tr>
<tr>
<td>Maximum Number of Backup Log Files</td>
<td>Sets the maximum number of log files stored before the oldest files are completely overwritten for newer files. When there are multiple backup log files, files will be named esp_server-&lt;number&gt;.log, with the highest number marking the oldest file and the lowest file marking the most recent. If the log file size parameter is set to zero, setting this parameter is unnecessary.</td>
</tr>
<tr>
<td>Time Interval</td>
<td>Sets the constant interval expression that specifies the maximum age of rows in a window, in seconds. Default value is 1.</td>
</tr>
<tr>
<td>Precision</td>
<td>Sets decimal display characteristics for number characters in the project. Default value is 6.</td>
</tr>
<tr>
<td>Memory</td>
<td>Sets memory usage limits for the project in megabytes. Default is 0, meaning unlimited.</td>
</tr>
<tr>
<td>Command Port</td>
<td>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.</td>
</tr>
<tr>
<td>SQL Port</td>
<td>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.</td>
</tr>
<tr>
<td>Gateway Port</td>
<td>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.</td>
</tr>
<tr>
<td>Project Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 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 consistent 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).                                                                                                                                                                                                                               |
| 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. The minimum value is 32 and the maximum value is 2147483647 (2^{31}-1). 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: STREAMING_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.                                                                                                                                                                                                                                   |
| Memory Reserve (memory-reserve) | Specifies the amount of reserved memory, in megabytes, that you want released to shut a project for ESP down gracefully in the case that all available memory is used. The default value for this option is 0. The recommended value is 10. Only Linux and Solaris platforms support this function.                                                                                                                   |
2.4.3 **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.

Failover refers to restarting a process after it stops, whereas Active-Active refers to keeping a stand-by application already running to take over processing. Thus, if required, you could have an HA deployment `Project ha ="true" with Failover enable="false".`

---

**Example**

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.

```xml
<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>
        <Failover enable="false"/>
      </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>
```
2.4.4 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 node, but it can have a strong positive affinity for only one node.

- **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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>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.</td>
</tr>
<tr>
<td>Strength</td>
<td>Specify <em>Strong or weak</em>. 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.</td>
</tr>
<tr>
<td>Charge</td>
<td>Specify <em>Positive or negative</em>. 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.</td>
</tr>
</tbody>
</table>

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.
2.4.5 Processor Affinity

Set processor affinities in the CCR file to specify which processors a project can run on.

To set processor affinities through the CCR Project Configuration Editor, select the Advanced tab and click Add Processor Affinity. This creates an entry for the instance under Project Runtime. Select this entry and set the Processor Affinity parameter. When deploying a project in HA (active-active) mode, click Add Processor Affinity again to set affinities for an additional instance.

To set processor affinities in the CCR file through the Text Editor, add and set the <ProcessorAffinity> tag in the <Runtime> section. For example:

```
<ProcessorAffinity instance="0">1-5</ProcessorAffinity>
```

Here instance refers to the order of the instances as they appear in the <Deployment><Instances> section of the CCR file, beginning with 0. When deploying a project in HA (active-active) mode, add another <ProcessorAffinity> tag with the corresponding order number for instance.

When using the processor affinity option, make sure that, when the project is deployed to the cluster, it can find the specified processors. Adding a strong positive controller node affinity to the instance can help. If the nodes in an ESP cluster have the same processors or you would like to use a subset of processors across machines, this node affinity is not necessary.

Note

On Windows systems with more than 64 processors, the affinity mask must specify processors in a single processor group. The 64-bit versions of Windows 7, Windows Server 2008 R2, and later versions of Windows, support more than 64 logical processors on a single computer. Processor groups are not available on 32-bit versions of Windows.

On Solaris, only one processor needs to be specified. If more than one is specified, the first processor will be used.

2.4.6 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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failover</td>
<td>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.</td>
</tr>
</tbody>
</table>
### 2.4.7 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.

### 2.4.8 Project Configurations

A project configuration is an XML document that governs specific runtime properties of a project, including stream URI bindings, adapter properties, parameter values, and advanced deployment options.

Project configuration files are created and edited separately from the project they are attached to, and are identified by their `.ccr` file extension. View and edit project configuration files in the Project Explorer view in the SAP ESP Authoring perspective.

Configuration files maintain all run-time properties outside the CCL. Thus, you can maintain CCL and CCX files under version control, while varying run-time properties. This allows a project to be moved from a test environment to a production environment without modifying the CCL and CCX files.

By default, when a new project is created, a new project configuration file is also created. One project may have multiple configuration files attached to it, so you can manually create new project configurations.

---

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failures per interval</td>
<td>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.</td>
</tr>
<tr>
<td>Failure interval</td>
<td>(Optional) This specifies the time, in seconds, that makes up an interval. If left blank, the interval time is infinite.</td>
</tr>
</tbody>
</table>
2.4.8.1 Creating a Project Configuration

Create a project configuration and edit configuration properties. When you create a new project, a project configuration file is automatically generated. However, you can create additional project configuration files as follows:

Procedure

2. Select the folder in which to store the new configuration file, and assign it a file name.
3. Click Finish.
   You see the CCR Project Configuration Editor window.

Related Information

Opening an Existing Project Configuration [page 41]
   Open an existing project configuration file.

Project Configuration File Editor [page 42]
   Using the CCR Project Configuration File Editor you can select one of five categories of information and edit in the project configuration file.

Advanced Project Deployment Options [page 62]
   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, within Studio.

2.4.8.2 Opening an Existing Project Configuration

Open an existing project configuration file.

Context

By default, new projects create a project configuration so each project has at least one existing project configuration.
Procedure

1. Select **Window > Open Perspective > SAP ESP Authoring** or click the **SAP ESP Authoring** tab.
2. Select **Window > Show View > Project Explorer**.
3. Locate the project configuration file, which appears as `<projectname>.ccr`. Double-click to open the file.

Related Information

**Creating a Project Configuration** [page 41]
Create a project configuration and edit configuration properties. When you create a new project, a project configuration file is automatically generated. However, you can create additional project configuration files as follows:

**Project Configuration File Editor** [page 42]
Using the CCR Project Configuration File Editor you can select one of five categories of information and edit in the project configuration file.

**Advanced Project Deployment Options** [page 62]
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, within Studio.

2.4.8.3 Project Configuration File Editor

Using the CCR Project Configuration File Editor you can select one of five categories of information and edit in the project configuration file.

The CCR Project Configuration File Editor has five tabs, each one corresponding to one of the five categories of project configuration information.

Related Information

**Creating a Project Configuration** [page 41]
Create a project configuration and edit configuration properties. When you create a new project, a project configuration file is automatically generated. However, you can create additional project configuration files as follows:

**Opening an Existing Project Configuration** [page 41]
Open an existing project configuration file.

**Advanced Project Deployment Options** [page 62]
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, within Studio.
2.4.8.3.1 Editing Cluster Parameters in Project Configuration

Configure local or remote clusters that your project can bind to for receiving and providing data. These clusters can then be used when configuring bindings.

Procedure

1. In the CCR Project Configuration Editor window, select the Clusters tab.
2. Click the name of an existing cluster in the All Clusters pane to edit that cluster’s information or click Add to add a new cluster.
   The editor displays the Cluster Details pane.
3. Enter the requested information in the Cluster Details pane.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster URL</td>
<td>Enter the hostname of the cluster.</td>
</tr>
<tr>
<td>Type</td>
<td>Toggle between local (no server information necessary) and remote</td>
</tr>
<tr>
<td></td>
<td>(server information must be known) cluster connection options.</td>
</tr>
<tr>
<td>User Name</td>
<td>Enter a user name to use when logging in to the cluster.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter a password to use when logging in to the cluster.</td>
</tr>
</tbody>
</table>

4. (Optional) Click Encrypt after entering the user name or password.
   a. Fill in the required fields in the Content Encryption pane, including Cluster URI, comprised of your host name and port number (<HOST>:<PORT>) and credential fields.
   b. Click Encrypt.
      The editor redisplay Cluster Details pane with the field you chose to encrypt (either the user name or password) filled with randomized encryption characters.

   **Note**
   To reset the encryption, click Encrypt beside the appropriate field and click Reset when the Already Encrypted pop-up is displayed.

5. To add a remote cluster definition:
   a. In Cluster Details, select remote as the type.
   b. Right-click the cluster and select New > Cluster Manager.
   c. Configure each cluster node by selecting it and adding host and port information in the Cluster Manager Details pane. When the SSL protocol is enabled, the cluster hostname and port must include the https:// prefix. If you omit the https:// prefix, data cannot pass. When the SSL protocol is not enabled, the cluster hostname and port must include the http:// prefix. If you omit the http:// prefix, data cannot pass.
### Related Information

**Editing Bindings in Project Configuration** [page 44]
Configure input and output bindings to enable streams or windows in different projects to provide or receive data from one another.

**Editing Adapter Property Sets in Project Configuration** [page 51]
Use the CCR Project Configuration editor to configure adapter property sets in a project configuration file. Property sets are reusable sets of properties that are stored in the project configuration file. Using an adapter property set also allows you to move adapter configuration properties out of the CCL file and into the CCR file.

**Setting Parameters in Project Configuration** [page 52]
Edit existing parameter definition values and remove deleted parameters.

**Editing Advanced Options in Project Configuration** [page 53]
Modify project deployment properties, project options, and instances in a project configuration file.

**Datatypes** [page 59]
SAP Event Stream Processor supports integer, float, string, money, long, and timestamp datatypes for all of its components.

### 2.4.8.3.2 Editing Bindings in Project Configuration

Configure input and output bindings to enable streams or windows in different projects to provide or receive data from one another.

### Prerequisites

Verify that the streams or windows you want to bind have:

- Compatible schemas
- The same datatype for each field name
- The same column order
- The same number of columns

If you plan to bind to a project in another cluster, define the cluster of interest in the SAP ESP Run-Test perspective and add the cluster to the Cluster tab.

### Context

Configuring bindings is similar to attaching an input adapter to an input stream or window, but is more efficient as it directly connects the output of one project to the input of the other. Bindings connect projects to one another in the same way that adapters connect projects to outside data sources or destinations.

Bindings can be local, within the same cluster, or can connect projects in one cluster to projects in different clusters. You can configure bindings from either the source or the destination project—that is, you can choose to...
An input stream can receive data from different sources through multiple bindings; both input and output streams can provide data to different destinations through multiple bindings.

Bindings can convey data:

- From an output stream or window in the current project to an input stream or window in a remote project. This is called an output binding.
- From a stream or window in a remote project to an input stream or window in the current project. This is called an input binding; input is the default setting for a binding in the CCR file.
- From an input stream or window in one project to an input stream or window in another project. This is called an input-to-input binding. If you configure an input-to-input binding on the input stream or window that is providing the data, you must select the Output Binding type. (By default, an input stream or window assumes that any binding configured on it is an input binding.) However, if you configure an input-to-input binding on the input stream or window that is receiving the data, do not set Binding Type to Output Binding. For information on setting the Output parameter in the CCR file, see the SAP Event Stream Processor: Developer Guide.

Binding information is specified in the project configuration (CCR) file so that binding references may be changed at runtime, allowing the project to be used in multiple environments.

**Procedure**

1. In the CCR Project Configuration editor, select the **Bindings** tab.
2. To add a binding, click **Add**, or to display a list of available streams/windows, click **Discover**. You can create multiple bindings on a single stream or window.
3. To configure individual binding settings, use the **Binding Details** pane on the right side of the CCR Project Configuration editor.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Binding Type</strong></td>
<td>Property name as it appears in the ccr file: Output Type: boolean For most bindings, you need not set this option because it defaults to the correct value. Set this option to <strong>Output Binding</strong> only to configure an input stream or window in this project to send data to an input stream or window in a remote project. When you select <strong>Output Binding</strong> on an input stream, you tell the binding to publish (send data out) to the remote input stream. If you do not check the <strong>Output Binding</strong> box, the binding subscribes to data from the remote input stream because bindings on input streams receive data by default.</td>
</tr>
<tr>
<td><strong>Binding name</strong></td>
<td>Property name as it appears in the ccr file: BindingName Type: string (Optional) Apply a name to the binding.</td>
</tr>
<tr>
<td><strong>Local stream/window</strong></td>
<td>The value for this property is specified in the BindingName property in the ccr file Type: string Enter the name of the local stream or window (for example, localStream1) or click <strong>Discover</strong> to view and select from a list of streams/windows.</td>
</tr>
<tr>
<td><strong>Reconnect Interval</strong></td>
<td>Property name as it appears in the ccr file: ReconnectInterval Type: integer If the connection between the local and remote streams is lost, the project attempts to reconnect at the specified interval. To suppress all reconnection attempts, set Reconnect Interval to 0. Use positive whole number values to set the reconnection interval. Default interval is 5 seconds.</td>
</tr>
<tr>
<td><strong>Remote Stream properties</strong></td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Cluster</strong></td>
<td>Property name as it appears in the ccr file: Cluster</td>
</tr>
<tr>
<td></td>
<td>Type: string</td>
</tr>
<tr>
<td></td>
<td>Select the cluster that contains the project to bind to.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td>You can select only clusters listed on the Cluster tab of the SAP ESP Run-Test perspective.</td>
</tr>
<tr>
<td><strong>Remote stream/window</strong></td>
<td>Property name as it appears in the ccr file: RemoteStream</td>
</tr>
<tr>
<td></td>
<td>Type: string</td>
</tr>
<tr>
<td></td>
<td>Enter the name of the remote stream or window (for example, remoteStream1) or click Discover to view and select from a list of streams/windows.</td>
</tr>
<tr>
<td></td>
<td>If you use Discover, make sure the cluster and project you are binding to are both running.</td>
</tr>
<tr>
<td></td>
<td>If they are not, Discover cannot find their streams or windows.</td>
</tr>
<tr>
<td><strong>Workspace</strong></td>
<td>Property name as it appears in the ccr file: Workspace</td>
</tr>
<tr>
<td></td>
<td>Type: string</td>
</tr>
<tr>
<td></td>
<td>Enter the workspace name (for example, ws1) or click Discover to view and select from a list of workspaces.</td>
</tr>
<tr>
<td></td>
<td>If you use Discover, make sure the cluster and project you are binding to are both running.</td>
</tr>
<tr>
<td></td>
<td>If they are not, Discover cannot find their workspaces.</td>
</tr>
<tr>
<td><strong>Project</strong></td>
<td>Property name as it appears in the ccr file: Project</td>
</tr>
<tr>
<td></td>
<td>Type: string</td>
</tr>
<tr>
<td></td>
<td>Enter the project to access (for example, project1) or click Discover to view and select from a list of projects.</td>
</tr>
<tr>
<td></td>
<td>If you use Discover, make sure the cluster and project you are binding to are both running.</td>
</tr>
<tr>
<td></td>
<td>If they are not, Discover cannot find the project.</td>
</tr>
</tbody>
</table>

**Guarantee Delivery properties**
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Enable Guaranteed Delivery| Property name as it appears in the ccr file: EnableGD  
Type: boolean  
Enable GD for a binding to guarantee that if the connection between the binding and the remote stream is severed (by shutting down the project that contains the local stream, for example), all transactions that are supposed to be transmitted through the binding during its downtime are processed once the connection is re-established. |
|                           | **Note**  
When you enable GD on a binding, make sure:  
- The binding’s source data window is running in GD mode or GD mode with checkpoint.  
- The binding’s target data window is backed by a log store. |
| Enable Guaranteed Delivery Cache | Property name as it appears in the ccr file: EnableGDCache  
Type: string  
Enable this binding to cache data. When the source data window is in GD mode with checkpoint, the binding receives checkpoint messages indicating the last row of data that has been checkpointed by the window. If the binding is enabled for GD caching, it caches incoming transactions until it receives a checkpoint message from the source window. The checkpoint message triggers the binding to send to the target window all cached transactions up to the one indicated in the checkpoint message. The binding issues a GD commit to the source data window after releasing cached data.  
If GD caching is disabled, SAP ESP ignores checkpoint messages and the binding forwards data based on the Guaranteed Delivery Batch Size. SAP ESP ignores Enable Guaranteed Delivery Cache if the source data window is not in GD mode with checkpoint. |
| Guaranteed Delivery Name  | Property name as it appears in the ccr file: GDName  
Type: string  
Supply a unique name for the GD session (subscription) this binding establishes. |
| Guaranteed Delivery Batch Size | Property name as it appears in the ccr file: GDBatchSize  
Type: integer |
| Advanced Properties       | **Note**  
When you enable GD on a binding, make sure:  
- The binding’s source data window is running in GD mode or GD mode with checkpoint.  
- The binding’s target data window is backed by a log store. |
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Droppable** | Property name as it appears in the ccr file: Droppable  
Type: `boolean`  
(For input bindings only) If the reader cannot keep up, the connection to the bound stream/window is dropped and attempts to reconnect. The default value is false.                                                                                                                                                                                                                   |
| **Keep Base** | Property name as it appears in the ccr file: KeepBase  
Type: `boolean`  
(For input bindings only) Set this property to true to receive the initial contents of the stream, as well as the updates. The default value is true.                                                                                                                                                                                                              |
| **Lossy**   | Property name as it appears in the ccr file: Lossy  
Type: `boolean`  
(For input bindings only) If set to true, the binding is lossy and if the binding cannot keep up (meaning when the first project cannot send information to the second project), some data may be lost. If set to false, the binding is not lossy and cannot lose data if it cannot keep up.  
Setting this property to false may have negative performance impact. The default value is false.                                                                                                                                                                                                                                                                 |
| **Mirror**  | Property name as it appears in the ccr file: Mirror  
Type: `boolean`  
(For input bindings only) If set to true, the input binding stream/window keeps the same data as the remote stream/window to which it is bound and clears the data when the remote stream data is cleared. If set to false, the local binding stream/window does not check the status of the remote stream/window data and always keeps the local input binding stream/window data. The default value is false.                                                                                                                                 |
| **Only Base** | Property name as it appears in the ccr file: OnlyBase  
Type: `boolean`  
(For input and output bindings) Set to true to receive only the initial contents of the stream. The default value is false.                                                                                                                                                                                                                                                                             |
| **Base Drain Timeout** | Property name as it appears in the ccr file: BaseDrainTimeout  
Type: `integer`  
(For input bindings only) The maximum time, in milliseconds, to receive all base data for the connected stream before the connected remote project forces a disconnection. The default value is 0.                                                                                                                                                                                                                      |
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Pulse   | Property name as it appears in the ccr file: Pulse  
Type: integer  
(For input bindings only) Specify a period, in seconds, in which the remote stream/window pulses to check for updates. Use a non-zero value to enable this property. The default value is 0. |
| Queue Size | Property name as it appears in the ccr file: QueueSize  
Type: integer  
(For input bindings only) The maximum number of records to queue up before the binding connection is dropped. The default value is 0. |
| Query  | Property name as it appears in the ccr file: Query  
Type: string  
(For input bindings only) A string which specifies a valid SELECT SQL statement. |

4. To remove a binding, select it and click Remove.

**Related Information**

- **Editing Cluster Parameters in Project Configuration** [page 43]
  Configure local or remote clusters that your project can bind to for receiving and providing data. These clusters can then be used when configuring bindings.

- **Editing Adapter Property Sets in Project Configuration** [page 51]
  Use the CCR Project Configuration editor to configure adapter property sets in a project configuration file. Property sets are reusable sets of properties that are stored in the project configuration file. Using an adapter property set also allows you to move adapter configuration properties out of the CCL file and into the CCR file.

- **Setting Parameters in Project Configuration** [page 52]
  Edit existing parameter definition values and remove deleted parameters.

- **Editing Advanced Options in Project Configuration** [page 53]
  Modify project deployment properties, project options, and instances in a project configuration file.

- **Datatypes** [page 59]
  SAP Event Stream Processor supports integer, float, string, money, long, and timestamp datatypes for all of its components.
2.4.8.3.3 Editing Adapter Property Sets in Project Configuration

Use the CCR Project Configuration editor to configure adapter property sets in a project configuration file. Property sets are reusable sets of properties that are stored in the project configuration file. Using an adapter property set also allows you to move adapter configuration properties out of the CCL file and into the CCR file.

Context

Property sets appear in a tree format, and individual property definitions are shown as children to property sets.

Procedure

1. In the CCR Project Configuration editor, select the **Adapter Properties** tab.
2. (Optional) To create a list of adapter property sets that correspond to the **ATTACH ADAPTER** statements in the main CCL file for the project, click **Add from CCL**.
3. To create a new adapter property set, click **Add**.
4. In the Property Set Details pane, define a name for the property set.
5. To add a new property to a property set, right-click the set and select **New > Property**.

   **Note**
   You can add as many property items to a property set as required.

6. To configure a property:
   a. In the Property Details pane, define a name for the property.
   b. Enter a value for the property.
7. (Optional) To encrypt the property value:
   a. Select the property value and click **Encrypt**.
   b. Enter the required fields, including Cluster URI and credential fields.
   c. Click **Encrypt**.
      The value, and related fields, are filled with randomized encryption characters.

   **Note**
   To reset the encryption, click **Encrypt** beside the appropriate field. Change the values, as appropriate, then click **Reset**.

8. To remove items from the All Adapter Properties list:
   ○ Right-click a property set and select **Remove**, or
   ○ Right-click a property and select **Delete**.
Related Information

Editing Cluster Parameters in Project Configuration [page 43]
Configure local or remote clusters that your project can bind to for receiving and providing data. These clusters can then be used when configuring bindings.

Editing Bindings in Project Configuration [page 44]
Configure input and output bindings to enable streams or windows in different projects to provide or receive data from one another.

Setting Parameters in Project Configuration [page 52]
Edit existing parameter definition values and remove deleted parameters.

Editing Advanced Options in Project Configuration [page 53]
Modify project deployment properties, project options, and instances in a project configuration file.

Datatypes [page 59]
SAP Event Stream Processor supports integer, float, string, money, long, and timestamp datatypes for all of its components.

2.4.8.3.4 Setting Parameters in Project Configuration

Edit existing parameter definition values and remove deleted parameters.

Context

The list of parameter definitions is automatically populated based on parameters within any CCL documents in the project folder. You can change parameter definition values in the CCR editor. You can also remove parameters if the definition has been deleted from the CCL document.

Procedure

1. Select the Parameters tab in the Project Configuration editor.
2. To modify a parameter value, click the parameter and change the value in the Parameter Details pane.

   Note
   You cannot modify the parameter Name field.
   You cannot use functions within parameter value fields. The Project Configuration editor only accepts simple values that adhere to the standards set by ESP datatypes. See Datatypes for more information.

3. To remove deleted parameter definitions from the list, select Remove, which is located at the top of the list.
i Note
A parameter definition marked as (removed) has been deleted from the original CCL file and can be removed from the parameter definition list.

Related Information

Editing Cluster Parameters in Project Configuration [page 43]
Configure local or remote clusters that your project can bind to for receiving and providing data. These clusters can then be used when configuring bindings.

Editing Bindings in Project Configuration [page 44]
Configure input and output bindings to enable streams or windows in different projects to provide or receive data from one another.

Editing Adapter Property Sets in Project Configuration [page 51]
Use the CCR Project Configuration editor to configure adapter property sets in a project configuration file. Property sets are reusable sets of properties that are stored in the project configuration file. Using an adapter property set also allows you to move adapter configuration properties out of the CCL file and into the CCR file.

Editing Advanced Options in Project Configuration [page 53]
Modify project deployment properties, project options, and instances in a project configuration file.

Datatypes [page 59]
SAP Event Stream Processor supports integer, float, string, money, long, and timestamp datatypes for all of its components.

### 2.4.8.3.5 Editing Advanced Options in Project Configuration

Modify project deployment properties, project options, and instances in a project configuration file.

**Procedure**

1. In the CCR Project Configuration editor, select the **Advanced** tab.
2. If no project deployment item exists, select **Add**.
3. Choose a project deployment type from the Project Deployment Details window. The options are:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-HA</td>
<td>Non-HA deployments create one project option item and one instance item as children under the project deployment item.</td>
</tr>
</tbody>
</table>
HA

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HA</td>
<td>HA deployments create one project option item and two instance items as children under the project deployment item. HA provides for hot project failover between instances.</td>
</tr>
</tbody>
</table>

4. Options are shown in the main Project Deployment Node as Project Deployment Details. To add a deployment option to your project, click the check box to enable it, then set a value for it or accept the default. The following table describes these options and provides information on their settings.

<table>
<thead>
<tr>
<th>Project Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug Level (debug-level)</td>
<td>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</td>
</tr>
<tr>
<td>Performance Monitor Refresh Interval (time-granularity)</td>
<td>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.</td>
</tr>
<tr>
<td>Java Classpath</td>
<td>Sets the Java classpath. Value is the path to the classpath file.</td>
</tr>
<tr>
<td>Java Max Heap (java-max-heap)</td>
<td>Sets the max Java heap for the project. Default value is 256 megabytes.</td>
</tr>
<tr>
<td>Bad Record File (bad-record-file)</td>
<td>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: &lt;base-directory&gt;/&lt;workspace-name&gt;.&lt;project-name&gt;.&lt;instance-number&gt; &lt;base-directory&gt; is a property defined in the cluster configuration: ○ In ESP Cockpit cluster configuration, under Services ○ In the local (Studio) cluster: STREAMING_HOME/studio/clustercfg/studio.xml ○ In a remote cluster: STREAMING_HOME/cluster/config/&lt;cluster-name&gt;/cluster.xml</td>
</tr>
<tr>
<td>Utf8</td>
<td>Enables Utf8 functionality on the server. Default value is true; set to false to disable.</td>
</tr>
<tr>
<td>Project Option</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Web Service Enabled</td>
<td>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.</td>
</tr>
<tr>
<td>Optimize</td>
<td>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.</td>
</tr>
<tr>
<td>On Error Discard Record</td>
<td>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.</td>
</tr>
<tr>
<td><img src="https://example.com" alt="Note" /></td>
<td>If the computation of a key column fails, the record will be discarded regardless of this option.</td>
</tr>
<tr>
<td>On Error Log</td>
<td>If set to true, any computation errors that occur will be logged in the error message. The default value is true.</td>
</tr>
<tr>
<td>Maximum Log File Size</td>
<td>Sets the maximum size of a log file before a new log file is started. The default setting is 1MB.</td>
</tr>
<tr>
<td><img src="https://example.com" alt="Note" /></td>
<td>Setting this parameter to “0” will enable an unrestricted/unlimited size for a single log file.</td>
</tr>
<tr>
<td>Maximum Number of Backup Log Files</td>
<td>Sets the maximum number of log files stored before the oldest files are completely overwritten for newer files. When there are multiple backup log files, files will be named esp_server-&lt;number&gt;.log, with the highest number marking the oldest file and the lowest file marking the most recent.</td>
</tr>
<tr>
<td><img src="https://example.com" alt="Note" /></td>
<td>Setting a higher file size is beneficial for projects that involve higher logging levels and retrieving any useful information. However, it is not recommended to have a higher limit for a prolonged period, as larger files will inevitably lead to performance and latency issues.</td>
</tr>
<tr>
<td>Time Interval</td>
<td>Sets the constant interval expression that specifies the maximum age of rows in a window, in seconds. Default value is 1.</td>
</tr>
<tr>
<td>Precision</td>
<td>Sets decimal display characteristics for number characters in the project. Default value is 6.</td>
</tr>
<tr>
<td>Memory</td>
<td>Sets memory usage limits for the project in megabytes. Default is 0, meaning unlimited.</td>
</tr>
<tr>
<td>Project Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Command Port</td>
<td>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.</td>
</tr>
<tr>
<td>SQL Port</td>
<td>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.</td>
</tr>
<tr>
<td>Gateway Port</td>
<td>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.</td>
</tr>
<tr>
<td>Consistent Recovery</td>
<td>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</td>
</tr>
</tbody>
</table>
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 consistent 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).

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. The minimum value is 32 and the maximum value is 2147483647 (2^{31}−1). 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: STREAMING_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.

Memory Reserve (memory-reserve)

Specifies the amount of reserved memory, in megabytes, that you want released to shut a project for ESP down gracefully in the case that all available memory is used. The default value for this option is 0. The recommended value is 10. Only Linux and Solaris platforms support this function.

5. To configure an option item, complete these fields:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Select from the list of available options shown in the above table.</td>
</tr>
<tr>
<td>Value</td>
<td>Enter a value for the property option.</td>
</tr>
</tbody>
</table>

i Note

To return options to their default settings, click Reset All.
6. To add an affinity under the instance item, right-click the instance item and select **New > affinity**. Complete these fields:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name for the affinity item.</td>
</tr>
<tr>
<td>Strength</td>
<td>Select a strength level.</td>
</tr>
<tr>
<td>Type</td>
<td>Select a type. (for example, controller).</td>
</tr>
<tr>
<td>Charge</td>
<td>Select a charge.</td>
</tr>
</tbody>
</table>

7. (Optional) To add a processor affinity to an instance:
   a. Click **Add Processor Affinity**. This creates an entry for the primary instance under **Project Runtime**.
   b. Select the instance.
   c. In the Processor Affinity option, enter a list of processors for the instance to run on. This list is comma-delimited and accepts dashes to denote range. For example, 0,1,3-5.
   d. To add a processor affinity to a secondary instance, repeat steps a-c.

8. To remove items from the All Advanced Configurations list:
   - Select a project deployment item and click **Remove**.
   - Right-click an option or affinity item and select **Delete**.

**Related Information**

**Editing Cluster Parameters in Project Configuration** [page 43]
Configure local or remote clusters that your project can bind to for receiving and providing data. These clusters can then be used when configuring bindings.

**Editing Bindings in Project Configuration** [page 44]
Configure input and output bindings to enable streams or windows in different projects to provide or receive data from one another.

**Editing Adapter Property Sets in Project Configuration** [page 51]
Use the CCR Project Configuration editor to configure adapter property sets in a project configuration file. Property sets are reusable sets of properties that are stored in the project configuration file. Using an adapter property set also allows you to move adapter configuration properties out of the CCL file and into the CCR file.

**Setting Parameters in Project Configuration** [page 52]
Edit existing parameter definition values and remove deleted parameters.

**Datatypes** [page 59]
SAP Event Stream Processor supports integer, float, string, money, long, and timestamp datatypes for all of its components.
## 2.4.8.3.6 Datatypes

SAP Event Stream Processor supports integer, float, string, money, long, and timestamp datatypes for all of its components.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
</table>
| bigdatetime | Timestamp with microsecond precision. The default format is YYYY-MM-DDTHH:MM:SS:SSSSSS. All numeric datatypes are implicitly cast to bigdatetime. The rules for conversion vary for some datatypes:  
- All boolean, integer, and long values are converted in their original format to bigdatetime.  
- Only the whole-number portions of money(n) and float values are converted to bigdatetime. Use the cast function to convert money(n) and float values to bigdatetime with precision.  
- All seconddate values are multiplied by 1000000 and converted to microseconds to satisfy bigdatetime format.  
- All msdate values are multiplied by 1000 and converted to microseconds to satisfy bigdatetime format. |
<p>| bigint     | An alias for long. |
| binary     | Represents a raw binary buffer. Maximum length of value is platform-dependent, with a size limit of 2 gigabytes. NULL characters are permitted. |
| boolean    | Value is true or false. The format for values outside of the allowed range for boolean is 0/1/false/true/y/n/on/off/yes/no, which is case-insensitive. |
| seconddate | Date with second precision. The default format is YYYY-MM-DDTHH:MM:SS. |
| decimal    | Used to represent numbers that contain decimal points. Accepts two mandatory parameters, precision and scale, which determine the range of values that can be stored in a decimal field. precision specifies the total number (from 1 to 34) of digits that can be stored. scale specifies the number of digits (from 0 to precision) that can be stored to the right of the decimal point. The value 88.999p10s3 would have a decimal datatype of (10,3), which means the value has a decimal precision of 10 and a decimal scale of 3. |
| double     | A 64-bit numeric floating point with double precision. The range of allowed values is approximately -10^{308} through +10^{308}. Equivalent to float. |
| float      | A 64-bit numeric floating point with double precision. The range of allowed values is approximately -10^{308} through +10^{308}. Equivalent to double. |</p>
<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>integer</strong></td>
<td>A signed 32-bit integer. The range of allowed values is -2147483648 to +2147483647 ((-2^{31}) to (2^{31}-1)). Constant values that fall outside of this range are automatically processed as long datatypes. To initialize a variable, parameter, or column with a value of -2147483648, specify (-2147483647) -1 to avoid CCL compiler errors.</td>
</tr>
</tbody>
</table>
| **interval** | A signed 64-bit integer that represents the number of microseconds between two timestamps. Specify an interval using multiple units in space-separated format, for example, “5 Days 3 hours 15 Minutes”. External data that is sent to an interval column is assumed to be in microseconds. Unit specification is not supported for interval values converted to or from string data. When an interval is specified, the given interval must fit in a 64-bit integer (long) when it is converted to the appropriate number of microseconds. For each interval unit, the maximum allowed values that fit in a long when converted to microseconds are:  
  - MICROSECONDS (MICROSECOND, MICROS): +/- 9223372036854775807  
  - MILLISECONDS (MILLISECOND, MILLIS): +/- 9223372036854775  
  - SECONDS(SECOND, SEC): +/- 9223372036854  
  - MINUTES(MINUTE, MIN): +/- 153722867280  
  - HOURS(HOUR,HR): +/- 2562047788  
  - DAYS(DAY): +/- 106751991  
The values in parentheses are alternate names for an interval unit. When the maximum value for a unit is specified, no other unit can be specified or it causes an overflow. Each unit can be specified only once. |
<p>| <strong>long</strong> | A signed 64-bit integer. The range of allowed values is -9223372036854775808 to +9223372036854775807 ((-2^{63}) to (2^{63}-1)). To initialize a variable, parameter, or column with a value of -9223372036854775808, specify (-9223372036854775807) -1 to avoid CCL compiler errors. |
| <strong>money</strong> | A legacy datatype maintained for backward compatibility. It is a signed 64-bit integer that supports 4 digits after the decimal point. Currency symbols and commas are not supported in the input data stream. |</p>
<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>money(n)</td>
<td>A signed 64-bit numerical value that supports varying scale, from 1 to 15 digits after the decimal point. Currency symbols and commas are not supported in the input data stream, however, decimal points are. The supported range of values change, depending on the specified scale.</td>
</tr>
<tr>
<td></td>
<td><code>money(1)</code>: -922337203685477580.8 to 922337203685477580.7</td>
</tr>
<tr>
<td></td>
<td><code>money(2)</code>: -92233720368547758.08 to 922337203685477580.7</td>
</tr>
<tr>
<td></td>
<td><code>money(3)</code>: -9223372036854775.808 to 9223372036854775.807</td>
</tr>
<tr>
<td></td>
<td><code>money(4)</code>: -922337203685477.5808 to 922337203685477.5807</td>
</tr>
<tr>
<td></td>
<td><code>money(5)</code>: -92233720368547.75808 to 92233720368547.75807</td>
</tr>
<tr>
<td></td>
<td><code>money(6)</code>: -92233720368547.75808 to 92233720368547.75807</td>
</tr>
<tr>
<td></td>
<td><code>money(7)</code>: -9223372036854775.808 to 9223372036854775.807</td>
</tr>
<tr>
<td></td>
<td><code>money(8)</code>: -9223372036854775.808 to 9223372036854775.807</td>
</tr>
<tr>
<td></td>
<td><code>money(9)</code>: -9223372036.854775808 to 9223372036.854775807</td>
</tr>
<tr>
<td></td>
<td><code>money(10)</code>: -922337203.6854775808 to 922337203.6854775807</td>
</tr>
<tr>
<td></td>
<td><code>money(11)</code>: -922337203.6854775808 to 922337203.6854775807</td>
</tr>
<tr>
<td></td>
<td><code>money(12)</code>: -922337203.6854775808 to 922337203.6854775807</td>
</tr>
<tr>
<td></td>
<td><code>money(13)</code>: -922337203.6854775808 to 922337203.6854775807</td>
</tr>
<tr>
<td></td>
<td><code>money(14)</code>: -922337203.6854775808 to 922337203.6854775807</td>
</tr>
<tr>
<td></td>
<td><code>money(15)</code>: -922337203.6854775808 to 922337203.6854775807</td>
</tr>
</tbody>
</table>

To initialize a variable, parameter, or column with a value of -92,233.72036854775807, specify (-9...7) -1 to avoid CCL compiler errors.

Specify explicit scale for money constants with Dn syntax, where n represents the scale. For example, 100.1234567D7, 100.12345D5.

Implicit conversion between `money(n)` types is not supported because there is a risk of losing range or scale. Perform the cast function to work with money types that have different scale.

<table>
<thead>
<tr>
<th>string</th>
<th>Variable-length character string, with byte values encoded in UTF-8. Maximum string length is platform-dependent, with a size limit of 2 gigabytes. This size limit is reduced proportionally by the size of other content in the row, including the header.</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>Stores the time of day as a two-byte field having a range of 00:00:00 to 23:59:59. The default format is HH24:MM:SS.</td>
</tr>
<tr>
<td>msdate</td>
<td>A timestamp with millisecond precision. The default format is YYYY-MM-DDTHH:MM:SS:SSS.</td>
</tr>
</tbody>
</table>
Related Information

Editing Cluster Parameters in Project Configuration [page 43]
Configure local or remote clusters that your project can bind to for receiving and providing data. These clusters can then be used when configuring bindings.

Editing Bindings in Project Configuration [page 44]
Configure input and output bindings to enable streams or windows in different projects to provide or receive data from one another.

Editing Adapter Property Sets in Project Configuration [page 51]
Use the CCR Project Configuration editor to configure adapter property sets in a project configuration file. Property sets are reusable sets of properties that are stored in the project configuration file. Using an adapter property set also allows you to move adapter configuration properties out of the CCL file and into the CCR file.

Setting Parameters in Project Configuration [page 52]
Edit existing parameter definition values and remove deleted parameters.

Editing Advanced Options in Project Configuration [page 53]
Modify project deployment properties, project options, and instances in a project configuration file.

2.4.8.4 Advanced 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, within Studio.

Active-Active Deployments

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.

Project Options

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.
Instances

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.

Failover

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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failover</td>
<td>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.</td>
</tr>
<tr>
<td>Failures per interval</td>
<td>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.</td>
</tr>
<tr>
<td>Failure interval</td>
<td>(Optional) This specifies the time, in seconds, that makes up an interval. If left blank, the interval time is infinite.</td>
</tr>
</tbody>
</table>

Affinities

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 node, but it can have a strong positive affinity for only one node.
- 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 to start 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.

---

**Processor Affinity**

Processor Affinity lets you set a project to only run on specified processors. In HA (active-active) mode, processor affinities can be set separately for each instance.

---

**Related Information**

- **Creating a Project Configuration** [page 41]
  
  Create a project configuration and edit configuration properties. When you create a new project, a project configuration file is automatically generated. However, you can create additional project configuration files as follows:

- **Opening an Existing Project Configuration** [page 41]
  
  Open an existing project configuration file.

- **Project Configuration File Editor** [page 42]
  
  Using the CCR Project Configuration File Editor you can select one of five categories of information and edit in the project configuration file.

---

**2.4.9 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 SAP Event Stream Processor: Studio Users Guide. For information on adapters and adapter property sets, see the SAP Event Stream Processor: Studio Users Guide and the SAP Event Stream Processor: Adapters Guide.

```xml
<?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>
      </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>
</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.

```xml
<Deployment>
  <Project ha="true">
    <Options>
      <Option name="debug-level" value="1"/>
    </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>
        <Failover enable="false"/>
      </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>
```
2.4.10 Accessing Project Contents using SAP HANA Smart Data Access

Using the Smart Data Access feature in SAP HANA, you can connect to a project for ESP and query the contents of its windows, using the content as a data source for SAP HANA.

Context

Accessing the contents of a project for ESP using Smart Data Access involves:

Procedure

1. Installing the ESP ODBC driver on the SAP HANA machines you will be using with ESP. For information, see the ESP Installation Guide under Post Installation > Installing the ODBC Driver for SAP HANA Smart Data Access.
2. On the SAP HANA machine, create a remote source over the ESP window.
3. On the SAP HANA machine, create a virtual table over the remote source.
   For detailed information, see the topics related to ESP in SAP HANA Administration Guide > Administration of SAP HANA Data Provisioning Technologies > SAP HANA Data Provisioning Technologies > About SAP HANA Smart Data Access.

Related Information

SAP HANA Administration Guide

2.5 Deploying an Adapter to a Cluster

To manage an adapter using a cluster, add the adapter to the cluster using the streamingclusteradmin command line utility. You can only do this for adapters developed using the adapter toolkit (either preconfigured or custom). For a complete list of adapters developed using the adapter toolkit, see Preconfigured Adapters Included with the Adapter Toolkit in the SAP Event Stream Processor: Building Custom Adapters.

Prerequisites

Configure one of the toolkit adapters.
Context

You can start, stop, and remove an adapter from a cluster using SAP ESP Cockpit, SAP Event Stream Processor Studio, or the `streamingclusteradmin` command line utility.

Procedure

1. Add the adapter to the cluster:

   ```
   streamingclusteradmin --uri=esp[s]://localhost:19011 --username=user --password=pass --add_adapter --workspace-name=<workspacename> --adapter-name=<adaptername> --adapter-type=toolkit_adapter --arc=/config/adapter_config.xml
   ```

   **Note**

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

2. (Optional) Start the adapter:

   ```
   streamingclusteradmin --uri=esp[s]://localhost:19011 --username=user --password=pass --start_adapter --workspace-name=<workspacename> --adapter-name=<adaptername>
   ```

2.5.1 Application Deployment Parameters

Application deployment parameters specify whether failover or affinity are enabled on the application being deployed to the cluster. Manually create an application deployment (XML) configuration file to set these options. Each application that you wish to deploy to the cluster, such as an adapter, requires its own configuration file.

The ESP installation directory does not include a sample application deployment configuration file. See Sample Application Deployment Configuration File for sample file content.

These options are only valid for adapters built using the adapter toolkit. For a full list of these adapters, see Preconfigured Adapters Included with the Adapter Toolkit in the SAP Event Stream Processor: Building Custom Adapters guide.

<table>
<thead>
<tr>
<th>XML Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationDeployment</td>
<td>(Required) Root element containing all elements below.</td>
</tr>
<tr>
<td>Instances</td>
<td>(Required) Section containing the Instance element and its sub elements.</td>
</tr>
<tr>
<td>XML Element</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Instance     | Type: string  
(Required) Specify the application instance. Contains two attributes, name, which specifies the application name, and type, which specifies the application instance type. The only valid value for the type attribute is toolkit_adapter.  
This element contains the Failover and Affinities elements below. |
| Failover     | Type: boolean  
(Required) Specify whether you want to enable failover for the adapter. If disabled, adapter restarts are not permitted. If enabled, the FailureInterval and FailurePerInterval elements are accessible and adapter restarts are permitted.  
Contains the FailureInterval and FailurePerInterval elements. |
| FailureInterval | Type: integer  
(Required) Specify the time, in seconds, that makes up an interval.  
If failover is enabled, do not leave this value blank. If failover is enabled and this value is empty, the adapter cannot be successfully added to the cluster. |
| FailurePerInterval | Type: integer  
(Required) Specify the number of restarts the application can attempt within a given interval. This count resets to zero if you restart the application manually or if failures are dropped from the list because they are older than the size of the interval. |
<p>| Affinities   | (Optional) Section containing the Affinity element. |</p>
<table>
<thead>
<tr>
<th>XML Element</th>
<th>Description</th>
</tr>
</thead>
</table>
| Affinity    | Type: string  
(Optional) Specify a controller affinity which determines which controller node the application can run on. The only valid value is controller.  
This element contains four attributes, charge, strength, type, and value. If you have specified an affinity, also specify values for these attributes.  
Charge can be either positive or negative. If positive, the application runs on the controller node. If negative, the application does not run on the controller node.  
Strength can be either strong or weak. Strong requires the application to run only on a specific controller node. If you have strong, positive affinity set for a controller node, and that node fails, the failover process tries to restart the application on that node. If the node has not recovered, the application restart fails and you must restart manually.  
Type specifies the affinity type. The only valid value for this attribute is controller.  
Value specifies the value of the affinity. For example, if the affinity type is controller, its value should be the controller node name for which the affinity is set. |

### 2.5.2 Sample Application Deployment Configuration File

Example content of an application deployment configuration file.

```xml
<?xml version="1.0" encoding="utf-8"?>
<ApplicationDeployment xmlns="http://www.sybase.com/esp/application/2014/01/">
  <Instances>
    <Instance name="adapter1" type="toolkit_adapter">
      <Failover enable="true">
        <FailureInterval>1</FailureInterval>
        <FailuresPerInterval>1</FailuresPerInterval>
      </Failover>
      <Affinities>
        <Affinity charge="negative" strength="strong" type="controller" value="node2"/>
      </Affinities>
    </Instance>
  </Instances>
</ApplicationDeployment>
```
2.5.3 Adapter Cold Failover

An adapter fails when it does not run properly or stops running properly. If failover is enabled, the cluster restarts the adapter if it is killed or stopped abnormally.

Failover typically results in a restart, though a strong positive affinity to a node that is not available can prevent an adapter from restarting. Restarts can be limited based on failure intervals and restarts per interval.

Enable failover in the application deployment configuration file. See Application Deployment Parameters [page 68] for detailed information on configuring failover for adapters managed by a cluster.

2.5.4 Adapter Affinity

Adapters in cluster-managed mode support controller affinities, which lets you set which controller node the adapter can run on. Each adapter allows a single controller affinity.

Set adapter affinity in the application deployment configuration file. See Application Deployment Parameters [page 68] for detailed information on configuring adapters managed by a cluster.

2.6 Administer a Cluster

Attending to such items as logging, data backup, and data restoration will ensure that your cluster is administered properly.

2.6.1 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.

Context

SAP recommends starting a node using ESP Cockpit. For more information, see the ESP Cockpit documentation.
A cluster must be running before you can deploy projects to it. To start a cluster, start manager nodes first, then controller-only nodes. The cluster database, containing the cluster configuration, must be running. 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.

**Procedure**

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

   ```bash
   cd %STREAMING_HOME%/cluster/config/<cluster-name>
   %STREAMING_HOME%/bin/streamingclusternode --config <config-file>.cfg
   ```

2. To start a node on a UNIX host, execute:

   ```bash
   cd $STREAMING_HOME/cluster/config/<cluster-name>
   $STREAMING_HOME/bin/streamingclusternode --config <config-file>.cfg
   ```

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:

   ```bash
   streamingclusteradmin --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 `streamingclusteradmin` tool, Event Stream Processor prompts you for the password and hides it as you type, which improves security.

4. (Optional) In `streamingclusteradmin`'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:

   ```bash
   add project <workspace-name>/<project-name> <project-name>.ccx [<project-name>.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:

   ```bash
   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.

### 2.6.2 Stopping a Node or Cluster

Shut down the nodes in a cluster.

#### Prerequisites

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

There is no need to stop the cluster database when stopping a node or cluster. If you do need to stop a cluster database, see Stopping a Cluster Database [page 27].

#### Context

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.

#### Procedure

1. From the Windows command line, execute:

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

   On a Linux or Solaris system, execute:

   ```
   cd $STREAMING_HOME/cluster/nodes/<node_name> streamingclusteradmin
   --uri=esp[s]://<host>:19011 --username=<name> --password=<pass> --stop_node <node_name>
   ```

   **Note**

   These examples use the authentication syntax for a username/password authentication provider such as LDAP or native OS. For RSA authentication, use this command:

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

SAP Event Stream Processor produces log files for projects, as well as the cluster. A cluster node may contain multiple projects, each with its own project log file.

In Event Stream Processor, projects run on local and remote clusters. Only remote clusters generate cluster log files. However, projects log files are generated regardless of whether the node is local or remote.

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

2.6.3.1 Troubleshooting Log Files

ESP logs installation results, errors, and warnings from various components in different log files. Review these logs to help troubleshoot issues. If you require technical support, your representative may request that you send information from one or more of these logs.

This file contains a summary of the ESP installation results.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>esp_suite.log</td>
<td>SAP Event Stream Processor; includes a summary of installation results</td>
</tr>
<tr>
<td>conn_lang.log</td>
<td>Open Client connectivity language modules</td>
</tr>
<tr>
<td>dbilib.log</td>
<td>Open Client DB-Library</td>
</tr>
<tr>
<td>esp_excel_install.log</td>
<td>SAP Event Stream Processor Add-In for Microsoft Excel</td>
</tr>
<tr>
<td>esp_framework_install.log</td>
<td>Adapter Toolkit</td>
</tr>
<tr>
<td>esp_http_install.log</td>
<td>HTTP Output Adapter</td>
</tr>
<tr>
<td>esp_logfile_input_install.log</td>
<td>Logfile Input Adapter</td>
</tr>
<tr>
<td>esp_odbc_install.log</td>
<td>ODBC Driver</td>
</tr>
</tbody>
</table>
2.6.3.2 Cluster Log Configuration File

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

Each cluster has a cluster log configuration file, which resides in the cluster’s working directory:

`STREAMING_HOME/cluster/config/<cluster-name>/cluster.log.properties`

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

A sample `cluster.log.properties` file:

```properties
com.sybase.esp.cluster.logfile=cluster.log
log4j.rootLogger=info, Log
log4j.logger.com.sybase.esp=info
log4j.logger.com.sybase.esp.cluster.applications=info
log4j.additivity.com.sybase.esp.cluster.applications=false

log4j.appenders.Log=org.apache.log4j.RollingFileAppender
log4j.appenders.Log.File=${com.sybase.esp.cluster.logfile}
log4j.appenders.Log.MaxFileSize=1MB
log4j.appenders.Log.MaxBackupIndex=5
#log4j.appenders.Log.layout.ConversionPattern=%d{MMM dd yyyy HH:mm:ss.SSS} %p %c - %m%n
#log4j.appenders.Log.layout.ConversionPattern=%d{MMM dd yyyy HH:mm:ss.SSS} %p %c[1] - %m%n
```
In this sample configuration, the `com.sybase.esp.cluster.logfile` property in the first line specifies that the cluster log is written to the log file `cluster.log`, which is located by default in the cluster working directory, `STREAMING_HOME/cluster/config/<cluster-name>`. If you want to write the cluster log elsewhere, specify a path relative to the cluster 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.

The cluster log file 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.ConversionPattern=%d{MMM dd yyyy HH:mm:ss.SSS} %p - %m%n
```

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

### 2.6.3.3 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 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 `STREAMING_HOME\SybaseESP\5.1\workspace\<workspace-name>\.<project-name>\.<instance-number>`.

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

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

To modify logging levels for a project at runtime, use `streamingprojectclient`:

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

Log level changes made with `streamingprojectclient` 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.

You can modify the size and number of log files stored in a given project using the `logfile-size` and `logfile-depth` properties in the project configuration (ccr) file. You can edit the ccr file manually, or by using the Project Configuration editor in Studio. For more information, see the SAP Event Stream Processor: Studio Users Guide.

### 2.6.3.3.1 Logging Level

Logging levels range from 0 to 7, and represent a decreasing order of severity. For example, the higher the log error, the less severe the issue. The higher you set the log level, the more information you receive as errors up to and including that log level are reported. The default logging level for projects is 4.

You can set logging levels:

- In cluster configuration. Logging levels in cluster configuration 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 `streamingprojectclient` at runtime.

<table>
<thead>
<tr>
<th>Name</th>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG_EMERG</td>
<td>0</td>
<td>system is unusable</td>
</tr>
<tr>
<td>LOG_ALERT</td>
<td>1</td>
<td>action must be taken immediately</td>
</tr>
<tr>
<td>LOG_CRIT</td>
<td>2</td>
<td>critical conditions</td>
</tr>
<tr>
<td>LOG_ERR</td>
<td>3</td>
<td>error conditions</td>
</tr>
</tbody>
</table>
### 2.6.4 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 SAP Event Stream Processor Studio and in SAP ESP Cockpit.

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:

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

To connect to the cluster manager with ticket Kerberos authentication:

```
streamingclusteradmin --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:

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

> **Note**
> If you omit the password parameter when you call the `streamingclusteradmin` 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:

```
streamingclusteradmin --uri=esp[s]://cluster_server:19011 --username=me --password=sybase > get managers
```
These command line mode examples demonstrate the use of some of the parameters and commands:

```
streamingclusteradmin --uri=esp[s]://cluster_server:19011 --username=me --password=sybase --get_managers
streamingclusteradmin --uri=esp[s]://cluster_server:19011 --username=me --password=sybase --get_workspaces
streamingclusteradmin --uri=esp[s]://cluster_server:19011 --username=me --password=sybase --get_projects
```

Table 3: **streamingclusteradmin Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive mode: get managers</td>
<td>Returns the host-name:rpc-port pairs for the managers in the cluster.</td>
</tr>
<tr>
<td>Command line mode: --get_managers</td>
<td></td>
</tr>
<tr>
<td>Interactive mode: get controllers</td>
<td>Returns the list of controllers in the cluster.</td>
</tr>
<tr>
<td>Command line mode: --get_controllers</td>
<td></td>
</tr>
<tr>
<td>Interactive mode: get workspaces</td>
<td>Returns the names of the workspaces in the cluster.</td>
</tr>
<tr>
<td>Command line mode: --get_workspaces</td>
<td></td>
</tr>
<tr>
<td>Interactive mode: get projects</td>
<td>Returns the list of projects, with their state.</td>
</tr>
<tr>
<td>Command line mode: --get_projects</td>
<td></td>
</tr>
<tr>
<td>Interactive mode: get project &lt;workspace-name&gt;/&lt;project-name&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>Command line mode: --get_projectdetail --workspace-name=&lt;workspace-name&gt; --project-name=&lt;project-name&gt;</td>
<td></td>
</tr>
<tr>
<td>Interactive mode: get streams &lt;workspace-name&gt;/&lt;project-name&gt;</td>
<td>Returns the streams associated with a workspace.</td>
</tr>
<tr>
<td>Command line mode: --get_streams --workspace-name=&lt;workspace-name&gt; --project-name=&lt;project-name&gt;</td>
<td></td>
</tr>
<tr>
<td>Interactive mode: get schema &lt;workspace-name&gt;/&lt;project-name&gt; &lt;stream-name&gt;</td>
<td>Returns the schema of the specified stream.</td>
</tr>
<tr>
<td>Command line mode: --get_schema --workspace-name=&lt;workspace-name&gt; --project-name=&lt;project-name&gt; --stream-name=&lt;stream-name&gt;</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Function</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> add workspace &lt;workspace-name&gt; [--ignore-error]</td>
<td>Adds a workspace. Use the optional ignore-error argument to add the workspace even when doing so causes a workspace error.</td>
</tr>
<tr>
<td><strong>Command line mode:</strong> --add_workspace --workspace-name=workspace-name [--ignore-error]</td>
<td></td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> add project &lt;workspace-name&gt;/&lt;project-name&gt; &lt;project-name&gt;.ccx [&lt;project-name&gt;.ccr]</td>
<td>Adds a project. &lt;project-name&gt;.ccx is the compiled project file. Specify the path to the file. &lt;project-name&gt;.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. &lt;project-name&gt;.ccr and &lt;project-name&gt;.ccx are always located in the same directory.</td>
</tr>
<tr>
<td><strong>Command line mode:</strong> --add_project --workspace-name=workspace-name --project-name=project-name --ccx=project-name.ccx [--ccr=project-name.ccr]</td>
<td></td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> remove workspace &lt;workspace-name&gt; [--ignore-error]</td>
<td>Removes a workspace. Use the optional ignore-error argument to remove the workspace even when doing so causes a workspace error.</td>
</tr>
<tr>
<td><strong>Command line mode:</strong> --remove_workspace --workspace-name=workspace-name [--ignore-error]</td>
<td></td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> remove project &lt;workspace-name&gt;/&lt;project-name&gt;</td>
<td>Removes a project. Prerequisite: Stop the project. You cannot remove a running project.</td>
</tr>
<tr>
<td><strong>Command line mode:</strong> --remove_project --workspace-name=workspace-name --project-name=project-name</td>
<td></td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> start project &lt;workspace-name&gt;/&lt;project-name&gt; [timeout (sec)] [instance-index]</td>
<td>Starts the project. If the project is added with a strong controller affinity and that controller is not available, start-up fails. &lt;timeout-in-seconds&gt; 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.</td>
</tr>
<tr>
<td><strong>Command line mode:</strong> --start_project --workspace-name=workspace-name --project-name=project-name [--timeout=&lt;timeout-in-seconds&gt;] [--instance-index=&lt;instance-index&gt;]</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Function</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> stop project <code>&lt;workspace-name&gt;</code>/ <code>&lt;project-name&gt;</code> [timeout (sec)] [instance-index]</td>
<td>Stops the project. &lt;timeout-in-seconds&gt; 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 <code>get_project</code> or <code>get_project_detail</code> to determine whether and where the instances are running.</td>
</tr>
<tr>
<td><strong>Command line mode:</strong> `--stop_project --workspace-name=&lt;workspace-name&gt; --project-name=&lt;project-name&gt; [--timeout=&lt;timeout-in-seconds&gt;] [--instance-index=&lt;instance-index&gt;]</td>
<td></td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> stop node <code>&lt;node-name&gt;</code></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Command line mode:</strong> <code>--stop_node --node-name=&lt;node-name&gt;</code></td>
<td></td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> encrypt <code>&lt;clear-text&gt;</code></td>
<td>Encrypts plain text data. Use this command to encrypt passwords in configuration files.</td>
</tr>
<tr>
<td><strong>Command line mode:</strong> <code>--encrypt_text --text=&lt;clear-text&gt;</code></td>
<td></td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> deploykey <code>&lt;new-username&gt;</code> <code>&lt;keystore&gt;</code> <code>&lt;storepass&gt;</code> <code>&lt;key-alias&gt;</code> [&lt;storetype&gt;]</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Command line mode:</strong> <code>--deploy_key --new-user=&lt;new-username&gt; --keystore=&lt;keystore&gt; --storepass=&lt;storepass&gt; --key-alias=&lt;key-alias&gt; [--storetype=&lt;storetype&gt;]</code></td>
<td></td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> connect</td>
<td>Connect or reconnect a project to a cluster. This command is in interactive mode only.</td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> quit or exit</td>
<td>Logs you out of interactive mode. To reaccess the utility, provide your user name and password.</td>
</tr>
<tr>
<td><strong>Interactive mode:</strong> help</td>
<td>Displays a plain-text description of the <code>streamingclusteradmin</code> utility’s commands and usage information.</td>
</tr>
<tr>
<td><strong>Command line mode:</strong> <code>--help</code></td>
<td></td>
</tr>
</tbody>
</table>
### 2.6.5 Safeguarding Your Data

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

#### Context

SAP recommends that you:

#### Procedure

- 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 Studio to browse your source folder and make changes in the source files.
- Perform regular backups of project data, including log stores.
- Configure and monitor the log file size. Having the size set too low will result in files automatically overwriting themselves, resulting in a loss of data; having it set too high will eventually flood a server’s hard drive with logging data. For more information on log file size, see the SAP Event Stream Processor: Studio Users Guide.

### 2.6.5.1 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
- The `cluster.log.properties` file in `$STREAMING_HOME/cluster/config/<cluster-name>`
- Security configuration files in `$STREAMING_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, `/SybaseESP/5.1/workspace` unless you overrode the default when installing ESP
- The `cluster database` file, `$STREAMING_HOME/cluster/config/<cluster-name>/esp_cluster.db`
- The `cluster.log.properties` file in `%STREAMING_HOME%/cluster/config/<cluster-name>`
• security configuration files in %STREAMING_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.

### 2.6.5.1.1 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.

**Procedure**

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

   ```
   $STREAMING_HOME/bin/streamingclusteradmin --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 `streamingclusteradmin` 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 `streamingclusteradmin` 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>`.

### 2.6.5.1.2 Performing an Online Backup

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

#### Context

**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.

#### Procedure

1. Use the `streamingprojectclient` utility in the command prompt to create a backup copy of log store files. For example:

   ```bash
   $STREAMING_HOME/bin/streamingprojectclient -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.

### 2.6.5.1.3 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.
2.6.5.2 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
- the cluster.log.properties file in $STREAMING_HOME/cluster/config/<cluster-name>
- security configuration files in $STREAMING_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, /SybaseESP/5.1/workspace unless you overrode the default when installing ESP
- the cluster database file, $STREAMING_HOME/cluster/config/<cluster-name>/esp_cluster.db
- the cluster.log.properties file in %STREAMING_HOME%/cluster/config/<cluster-name>
- security configuration files in %STREAMING_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

2.6.5.2.1 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.

Procedure

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:
   ```bash
   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.

### 2.6.5.2.2 Restoring Backup Files on Windows Systems

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

#### Procedure

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.

   ```
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.

### 2.6.6 Memory Usage

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.

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 default location is:

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

Use the memory-reserve option to specify the amount of reserved memory, in megabytes, that you want released to shut a project for ESP down gracefully in the case that all available memory is used. The default value for this option is 0. The recommended value is 10. Only Linux and Solaris platforms support this function.
2.7 Monitor a Cluster

You can monitor a cluster using various tools.

2.7.1 Monitoring a Project

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

The `streamingmonitor` 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 Event Stream Processor Studio set this project option. For more information on configuring projects in SAP Event Stream Processor Studio, see the *SAP Event Stream Processor: Studio Users Guide*.

**Note**

The `streamingprojectclient` 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:

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

For more information on the `streamingmonitor` tool, see the *SAP Event Stream Processor: Utilities Guide*.

2.7.1.1 Monitoring Project Status

Use the `--get_projects` command for the `streamingclusteradmin` utility to monitor the status of your ongoing projects.

The `streamingclusteradmin --get_projects` command lists all available projects independent of a workspace. The project status is provided with each project reported. To show the statuses of all currently existing projects using Command Line Mode, enter:

```
streamingclusteradmin --uri=esps://<host>://<port> --username=<user name> --password=<password> --get_projects
```
Status Types and Definitions

For each project, `--get_projects` provides two types of statuses:

<table>
<thead>
<tr>
<th>Status Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requested</td>
<td>The desired status that the project is attempting to achieve.</td>
</tr>
<tr>
<td>Current</td>
<td>The actual status at the time of the report.</td>
</tr>
</tbody>
</table>

The requested status can be one of the following:

<table>
<thead>
<tr>
<th>Requested Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Started-Running</td>
<td>The project is fully functional.</td>
</tr>
<tr>
<td>Stopped</td>
<td>The project is stopped entirely.</td>
</tr>
</tbody>
</table>

When the project is requested to start or stop, the requested status becomes started-running or stopped, respectively. The project then goes through a cycle of starting up or shutting down. The cycle is not instantaneous.

The current status can be one of the following:

<table>
<thead>
<tr>
<th>Current Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting</td>
<td>The project has been requested to start, but the process has not started yet. The requested status is started-running.</td>
</tr>
<tr>
<td>Started</td>
<td>The controller has received the request and is about to start the project. The requested status is started-running.</td>
</tr>
<tr>
<td>Started-Initializing</td>
<td>The project process has started and is in contact with the cluster, but it is not yet functional. The requested status is started-running.</td>
</tr>
<tr>
<td>Started-Running</td>
<td>The project is now fully functional. The requested status is started-running.</td>
</tr>
<tr>
<td>Stopping</td>
<td>The project has been requested to stop, but the shutdown cycle is not complete yet. The requested is will be stopped.</td>
</tr>
<tr>
<td>Stopped</td>
<td>The project is stopped is stopped entirely. The requests status is stopped.</td>
</tr>
<tr>
<td>Failed</td>
<td>The project achieved started-running status, but it has suddenly stopped without a stop request. The requested status is started-running.</td>
</tr>
</tbody>
</table>
## Current Status

<table>
<thead>
<tr>
<th>Current Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start-Failed</td>
<td>An attempt was made to start the project, but the project never achieved started-running status. The requested status is started-running.</td>
</tr>
</tbody>
</table>

Use the `--get_projects` command in either Command Line Mode or Interactive Mode. For further instruction, consult the Server Executables section of the SAP Event Stream Processor: Utilities Guide.

### Investigating a Failed Status

If your project has failed for any reason, search the project base directory and examine the `esp_server.log`, `stdstreams.log`, and `cluster.log` of that project for any errors reported. Project logs are stored in different directories depending on whether the project is deployed on a local or remote cluster. For more information on log files, consult the File and Directory Infrastructure and Project Logging sections of the SAP Event Stream Processor: Configuration and Administration Guide. For information on troubleshooting project errors, consult the Troubleshoot A Cluster section of the SAP Event Stream Processor: Configuration and Administration Guide.

### 2.7.2 Monitoring with SAP ESP Cockpit

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

The ESP Cockpit architecture allows multiple administrators using Web clients to monitor and control all the Event Stream Processor components in an enterprise through one or more ESP Cockpit servers. ESP Cockpit 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 ESP Cockpit 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.

2.7.3 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 streamingsubscribe 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.

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

For details on streamingsubscribe, see the SAP Event Stream Processor: 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
```

2.7.3.1 _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 SAP Event Stream Processor: Building Custom Adapters 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 SAP Event Stream Processor: Building Custom Adapters to determine in which mode a module operates. See Adapter
Controller Parameters for the adapter of your choice in the SAP Event Stream Processor: 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 SAP Event Stream Processor: Adapters Guide.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapter_name</td>
<td>string</td>
<td>A unique name of the adapter instance.</td>
</tr>
<tr>
<td>stat_name</td>
<td>string</td>
<td>The name of an adapter statistic, as defined by the adapter.</td>
</tr>
<tr>
<td>last_update</td>
<td>bigdatetime</td>
<td>The time that the statistic was last updated.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>The value of the statistic (converted to a string).</td>
</tr>
</tbody>
</table>

2.7.3.2 _ESP_Clients

Contains information about all the currently active gateway client connections.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>long</td>
<td>A unique integer ID of the connection.</td>
</tr>
<tr>
<td>user_name</td>
<td>string</td>
<td>The user name to log in to the connection, shown once the user is authenticated.</td>
</tr>
<tr>
<td>IP</td>
<td>string</td>
<td>The address of the client machine.</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>The symbolic host name of the client machine, if available. If not available, host is the IP address of the client machine.</td>
</tr>
<tr>
<td>port</td>
<td>integer</td>
<td>The TCP port number from which the connection originates.</td>
</tr>
<tr>
<td>login_time</td>
<td>msdate</td>
<td>The time the server accepts (but does not authenticate) the connec tion, in GMT.</td>
</tr>
<tr>
<td>conn_tag</td>
<td>string</td>
<td>The user-set symbolic connection tag name. If not set by the user, conn_tag is NULL.</td>
</tr>
</tbody>
</table>

2.7.3.3 _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.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>long</td>
<td>A unique integer ID of the connection.</td>
</tr>
<tr>
<td>Column</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>user_name</td>
<td>string</td>
<td>The user name provided by the client during connection establishment. Shown once the user is authenticated.</td>
</tr>
<tr>
<td>IP</td>
<td>string</td>
<td>The IP address of the client machine, as a string.</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>The symbolic host name of the client machine, if available. If not available, host is the IP address of the client machine.</td>
</tr>
<tr>
<td>port</td>
<td>integer</td>
<td>The TCP port number from which the connection originates.</td>
</tr>
<tr>
<td>login_time</td>
<td>msdate</td>
<td>The time the server accepts (but does not authenticate) the connection.</td>
</tr>
<tr>
<td>conn_tag</td>
<td>string</td>
<td>The user-set symbolic connection tag name. If not set by the user, conn_tag is NULL.</td>
</tr>
<tr>
<td>cpu_pct</td>
<td>float</td>
<td>Total CPU usage for the client thread, as a percentage of a single CPU core.</td>
</tr>
<tr>
<td>last_update</td>
<td>secondate</td>
<td>The time of the current update.</td>
</tr>
<tr>
<td>subscribed</td>
<td>integer</td>
<td>The status of a subscription to a stream: 1 if subscribed; 0 if not subscribed, indicating a publisher.</td>
</tr>
<tr>
<td>sub_trans_per_sec</td>
<td>float</td>
<td>The client’s performance, in transactions per second, received by the client since the last update.</td>
</tr>
<tr>
<td>sub_rows_per_sec</td>
<td>float</td>
<td>The client’s performance, in data rows per second, received by the client since the last update.</td>
</tr>
<tr>
<td>sub_inc_trans</td>
<td>long</td>
<td>The number of transactions, envelopes, or messages received by the client since the last update.</td>
</tr>
<tr>
<td>sub_inc_rows</td>
<td>long</td>
<td>The number of data rows received by the client since the last update.</td>
</tr>
<tr>
<td>sub_total_trans</td>
<td>long</td>
<td>The total number of transactions, envelopes, or messages received by the client. Transactions, envelopes, and messages still in the queue are not counted.</td>
</tr>
<tr>
<td>sub_total_rows</td>
<td>long</td>
<td>The total number of data rows received by the client. Rows still in the queue are not counted.</td>
</tr>
<tr>
<td>sub_dropped_rows</td>
<td>long</td>
<td>The total number of data rows dropped in the gateway because they were not read quickly enough by the client. For lossy subscriptions.</td>
</tr>
<tr>
<td>sub_accum_size</td>
<td>integer</td>
<td>The current number of rows collected in the accumulator to be sent in the next pulse. For pulsed subscriptions.</td>
</tr>
<tr>
<td>sub_queue</td>
<td>integer</td>
<td>The number of rows queued for transmission to the client.</td>
</tr>
<tr>
<td>sub_queue_fill_pct</td>
<td>float</td>
<td>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.</td>
</tr>
<tr>
<td>Column</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sub_work_queue</td>
<td>integer</td>
<td>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.</td>
</tr>
<tr>
<td>pub_trans_per_sec</td>
<td>float</td>
<td>The client’s performance, in transactions per second, sent by the client since the last update. Envelopes and any service messages count as transactions.</td>
</tr>
<tr>
<td>pub_rows_per_sec</td>
<td>float</td>
<td>The client’s performance, in data rows per second, sent by the client since the last update.</td>
</tr>
<tr>
<td>pub_inc_trans</td>
<td>long</td>
<td>The number of transactions, envelopes, or messages sent by the client since the last update.</td>
</tr>
<tr>
<td>pub_inc_rows</td>
<td>long</td>
<td>The number of data rows sent by the client since the last update.</td>
</tr>
<tr>
<td>pub_total_trans</td>
<td>long</td>
<td>The total number of transactions, envelopes, or messages sent by the client. Does not include the transactions/envelopes/messages in the queue.</td>
</tr>
<tr>
<td>pub_total_rows</td>
<td>long</td>
<td>The total number of data rows sent by the client. Does not include the rows in the queue.</td>
</tr>
<tr>
<td>pub_stream_id</td>
<td>long</td>
<td>The numeric ID of the stream to which the client is trying to currently publish data. Typically, pub_stream_id is -1, meaning that data is not currently being published to the stream. Since the server publishes data in very short bursts, the monitor will usually not report any value other than -1.</td>
</tr>
<tr>
<td>node_cpu_pct</td>
<td>float</td>
<td>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.</td>
</tr>
<tr>
<td>node_cpu_pct_system</td>
<td>float</td>
<td>System CPU usage for the client, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>node_cpu_pct_user</td>
<td>float</td>
<td>User CPU usage for the client, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>cpu_time</td>
<td>interval</td>
<td>Total CPU time since the creation of the client, in microseconds. Total CPU time equals system CPU time plus user CPU time.</td>
</tr>
<tr>
<td>cpu_time_system</td>
<td>interval</td>
<td>Total system CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>cpu_time_user</td>
<td>interval</td>
<td>Total user CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>time_since_start</td>
<td>interval</td>
<td>Duration of lapsed real time since the creation of the client thread.</td>
</tr>
</tbody>
</table>
2.7.3.4 _ESP_Clockupdates

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

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>string</td>
<td>The type of the update, currently &quot;CLOCK&quot;.</td>
</tr>
<tr>
<td>Rate</td>
<td>float</td>
<td>The rate of the logical clock relative to the real time.</td>
</tr>
<tr>
<td>Time</td>
<td>float</td>
<td>The current time in seconds since the UNIX epoch.</td>
</tr>
<tr>
<td>Real</td>
<td>integer</td>
<td>The real time flag. 1 if the logical clock matches the system time and 0 if the times do not match.</td>
</tr>
<tr>
<td>stop_depth</td>
<td>integer</td>
<td>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.</td>
</tr>
<tr>
<td>max_sleep</td>
<td>integer</td>
<td>The time, in real milliseconds, that guarantees all sleepers discover changes in the physical clock rate or time.</td>
</tr>
</tbody>
</table>

2.7.3.5 _ESP_Columns

Contains information about all columns of all streams.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usename</td>
<td>string</td>
<td>Hard-coded as &quot;user&quot;.</td>
</tr>
<tr>
<td>relname</td>
<td>string</td>
<td>The name of the stream that contains columns described by this row.</td>
</tr>
<tr>
<td>attname</td>
<td>string</td>
<td>The name of the column described by this row.</td>
</tr>
<tr>
<td>atttypid</td>
<td>integer</td>
<td>The internal PostgreSQL value representing the type of this column. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For integer – 23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For long – 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For money – 701</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For float – 701</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For date – 1114</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For timestamp – 1114</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For string – 1043</td>
</tr>
<tr>
<td>attnum</td>
<td>integer</td>
<td>The position of this column in the schema, starting from 0.</td>
</tr>
</tbody>
</table>
### 2.7.3.6 _ESP_Config

Contains the current CCX of the project.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>string</td>
<td>Hard-coded as &quot;XML&quot;.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>The text of the current CCX.</td>
</tr>
</tbody>
</table>

### 2.7.3.7 _ESP_Connectors

Contains information about all internal adapters defined in the project.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>The name of the adapter, as defined in the project.</td>
</tr>
<tr>
<td>stream</td>
<td>string</td>
<td>The name of the stream on which the adapter is defined.</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>The adapter type defined in the ATTACH ADAPTER statement.</td>
</tr>
<tr>
<td>input</td>
<td>integer</td>
<td>Values are 1 for InConnection or 0 for OutConnection.</td>
</tr>
<tr>
<td>ingroup</td>
<td>string</td>
<td>The StartUp group where this connector belongs.</td>
</tr>
<tr>
<td>state</td>
<td>string</td>
<td>The state of the adapter, one of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● READY – ready to be started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● INITIAL – performing start-up and initialization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● CONTINUOUS – continuously receiving real-time data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● IDLE – currently not receiving data but attempting to re-connect the to the data source or link.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● DONE – no remaining input or output data; the adapter is about to exit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● DEAD – the adapter thread exited. The adapter remains in this state until explicitly requested to restart.</td>
</tr>
<tr>
<td>total_rows</td>
<td>long</td>
<td>The total number of data records recognized in the input data.</td>
</tr>
<tr>
<td>good_rows</td>
<td>long</td>
<td>The number of data records successfully processed.</td>
</tr>
<tr>
<td>bad_rows</td>
<td>long</td>
<td>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.</td>
</tr>
<tr>
<td>last_error_time</td>
<td>seconddate</td>
<td>The time that the error occurred in YYYY-MM-DD hh:mm:ss format.</td>
</tr>
<tr>
<td>last_error_msg</td>
<td>string</td>
<td>The complete text of the error message as written to the log.</td>
</tr>
</tbody>
</table>
### Column Type Description

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>latency</td>
<td>interval</td>
<td>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.</td>
</tr>
</tbody>
</table>

#### 2.7.3.8 _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 session, 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.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gd_key</td>
<td>string</td>
<td>The automatically generated key for the guaranteed delivery session. The gd_key is unique for a given gd_name/user_name combination.</td>
</tr>
<tr>
<td>stream_name</td>
<td>string</td>
<td>The name of the stream this guaranteed delivery session subscribes to.</td>
</tr>
<tr>
<td>user_name</td>
<td>string</td>
<td>The user associated with this guaranteed delivery session.</td>
</tr>
<tr>
<td>gd_name</td>
<td>string</td>
<td>The name of this guaranteed delivery session.</td>
</tr>
<tr>
<td>sequence_no</td>
<td>long</td>
<td>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.</td>
</tr>
<tr>
<td>client_handle</td>
<td>long</td>
<td>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.</td>
</tr>
<tr>
<td>last_update</td>
<td>bigdatatime</td>
<td>The last time this stream entry was updated.</td>
</tr>
</tbody>
</table>

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.
2.7.3.9  _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.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>string</td>
<td>The name of the stream owning the column described by this row.</td>
</tr>
<tr>
<td>field</td>
<td>string</td>
<td>The name of the column described by this row.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>The internal PostgreSQL value representing the type of this column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● For integer – 23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● For long – 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● For money – 701</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● For float – 701</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● For date – 1114</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● For timestamp – 1114</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● For string – 1043</td>
</tr>
</tbody>
</table>

2.7.3.10  _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.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>project_name</td>
<td>string</td>
<td>Currently hard-coded to the word “project”.</td>
</tr>
<tr>
<td>node_cpu_pct</td>
<td>float</td>
<td>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.</td>
</tr>
<tr>
<td>node_cpu_pct_system</td>
<td>float</td>
<td>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.</td>
</tr>
<tr>
<td>Column</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>node_cpu_pct_user</td>
<td>float</td>
<td>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.</td>
</tr>
<tr>
<td>cpu_time</td>
<td>interval</td>
<td>Total CPU time for the project, in microseconds. Total CPU time is equal to the system CPU time plus the user CPU time.</td>
</tr>
<tr>
<td>cpu_time_system</td>
<td>interval</td>
<td>Total system CPU time for the project, in microseconds.</td>
</tr>
<tr>
<td>cpu_time_user</td>
<td>interval</td>
<td>Total user CPU time for the project, in microseconds.</td>
</tr>
<tr>
<td>time_since_start</td>
<td>interval</td>
<td>Duration of lapsed real time since the project was started, in microseconds.</td>
</tr>
<tr>
<td>startmem_usage_vm</td>
<td>long</td>
<td>Total amount of virtual memory, in bytes, used by the project at the time of the update.</td>
</tr>
<tr>
<td>mem_usage_rss</td>
<td>long</td>
<td>Total amount of system memory (RSS), in bytes, used by the project at the time of the update.</td>
</tr>
<tr>
<td>num_threads</td>
<td>integer</td>
<td>Total number of threads used by the project at the time of the update.</td>
</tr>
<tr>
<td>last_update</td>
<td>bigdatet ime</td>
<td>Time of the current update.</td>
</tr>
</tbody>
</table>

### 2.7.3.11 _ESP_RunUpdates_

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

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>string</td>
<td>The type of the update. See table below.</td>
</tr>
<tr>
<td>value</td>
<td>integer</td>
<td>A number associated with the update, determined by the Key column.</td>
</tr>
<tr>
<td>stream</td>
<td>string</td>
<td>The name of the stream if the update notifies an event related to an individual stream; otherwise, stream NULL.</td>
</tr>
<tr>
<td>info</td>
<td>string</td>
<td>Additional information associated with the update. Its format depends on the type of the update.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Stream</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACE</td>
<td>0 or 1</td>
<td>None</td>
<td>Enabled (1) or disabled (0).</td>
</tr>
<tr>
<td>RUN</td>
<td>0 or 1</td>
<td>None</td>
<td>Event Stream Processor paused (0) or running (1).</td>
</tr>
</tbody>
</table>
### 2.7.3.12  _ESP_Streams

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

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_name</td>
<td>string</td>
<td>Hardcoded as &quot;user&quot;.</td>
</tr>
<tr>
<td>stream_name</td>
<td>string</td>
<td>The name of the stream described by this row.</td>
</tr>
<tr>
<td>handle</td>
<td>long</td>
<td>The stream’s numeric ID.</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>The type of the stream: &quot;stream&quot;, &quot;deltastream&quot;, &quot;window&quot;, or &quot;metadata&quot;.</td>
</tr>
<tr>
<td>visibility</td>
<td>string</td>
<td>The visibility of the stream: &quot;input&quot;, &quot;output&quot;, &quot;local&quot;, or &quot;intermediate&quot;.</td>
</tr>
<tr>
<td>target</td>
<td>string</td>
<td>The name of the target stream for streams with &quot;intermediate&quot; visibility value. For streams with all other visibility values, target is the same as the stream_name column.</td>
</tr>
<tr>
<td>gd_support</td>
<td>integer</td>
<td>Indicates whether guaranteed delivery is enabled for the stream or window:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0 - does not support guaranteed delivery</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1 - supports guaranteed delivery with no checkpoint messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2 - supports guaranteed delivery with checkpoint messages</td>
</tr>
</tbody>
</table>

Guaranteed delivery with checkpoints is available only when the server is running in consistent recovery mode or the auto checkpoint option is enabled.
2.7.3.13  _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.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream</td>
<td>string</td>
<td>The name of the stream.</td>
</tr>
<tr>
<td>target</td>
<td>string</td>
<td>The name of the target element.</td>
</tr>
<tr>
<td>cpu_pct</td>
<td>float</td>
<td>Total CPU usage for the stream thread, as a percentage of a single CPU core.</td>
</tr>
<tr>
<td>trans_per_sec</td>
<td>float</td>
<td>The stream’s performance, in transactions per second, since the last update.</td>
</tr>
<tr>
<td>rows_per_sec</td>
<td>float</td>
<td>The stream’s performance, in rows per second, since the last update.</td>
</tr>
<tr>
<td>inc_trans</td>
<td>long</td>
<td>The number of transactions processed by the server since the last update.</td>
</tr>
<tr>
<td>inc_rows</td>
<td>long</td>
<td>The number of rows processed by the server since the last update.</td>
</tr>
<tr>
<td>queue</td>
<td>integer</td>
<td>The current input queue size.</td>
</tr>
<tr>
<td>store_rows</td>
<td>long</td>
<td>The current number of records in the stream’s store.</td>
</tr>
<tr>
<td>last_update</td>
<td>seconddate</td>
<td>The time of the current update.</td>
</tr>
<tr>
<td>sequence</td>
<td>long</td>
<td>The sequence number of the current update.</td>
</tr>
<tr>
<td>posting_to_client</td>
<td>long</td>
<td>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.</td>
</tr>
<tr>
<td>node_cpu_pct</td>
<td>float</td>
<td>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.</td>
</tr>
<tr>
<td>node_cpu_pct_system</td>
<td>float</td>
<td>System CPU usage for the stream, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>node_cpu_pct_user</td>
<td>float</td>
<td>User CPU usage for the stream, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>cpu_time</td>
<td>interval</td>
<td>Total CPU time since the creation of the stream, in microseconds. Total CPU time equals system CPU time plus user CPU time.</td>
</tr>
<tr>
<td>cpu_time_system</td>
<td>interval</td>
<td>Total system CPU time, in microseconds, since the creation of the stream.</td>
</tr>
</tbody>
</table>
### 2.7.3.14 _ESP_Streams_Topology

Contains pairs of names for streams that are directly connected.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>src_stream</td>
<td>string</td>
<td>The name of the source stream.</td>
</tr>
<tr>
<td>dst_stream</td>
<td>string</td>
<td>The name of the destination stream.</td>
</tr>
</tbody>
</table>

### 2.7.3.15 _ESP_Subscriptions

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

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream_handle</td>
<td>long</td>
<td>The handle the server assigns to the subscribed stream.</td>
</tr>
<tr>
<td>conn_handle</td>
<td>long</td>
<td>The handle the server assigns to the subscribed connection.</td>
</tr>
</tbody>
</table>

### 2.7.3.16 _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.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream_handle</td>
<td>long</td>
<td>The handle of the stream the server subscribes to.</td>
</tr>
<tr>
<td>conn_handle</td>
<td>long</td>
<td>The handle of the connection the server subscribes to.</td>
</tr>
<tr>
<td>stream_name</td>
<td>string</td>
<td>The name of the stream.</td>
</tr>
<tr>
<td>stream_user</td>
<td>string</td>
<td>The user name of the owner of the stream.</td>
</tr>
<tr>
<td>subscriber_user</td>
<td>string</td>
<td>The login name of the user account that owns the subscription.</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>The IP address of the client machine.</td>
</tr>
</tbody>
</table>
### _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.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>relname</td>
<td>string</td>
<td>The name of the stream described by this row.</td>
</tr>
<tr>
<td>user name</td>
<td>string</td>
<td>Hard-coded as &quot;user&quot;.</td>
</tr>
<tr>
<td>remarks</td>
<td>string</td>
<td>The stream’s numeric ID, a decimal number as an ASCII string.</td>
</tr>
</tbody>
</table>
3 Configuring External Database Access

Use the Data Services View in the SAP ESP Authoring perspective of SAP Event Stream Processor Studio to manage database service definitions for local and remote clusters. Create a separate data service for every external database you want the SAP Event Stream Processor server to connect to.

The ESP server accesses external databases by using data service definitions which are managed and stored in separate databases for each cluster. Use the Data Services View to add, edit or delete database service definitions for local and remote clusters. You can also migrate database services defined in the service.xml file of previous releases of ESP, using the service.xml migration command of the streamingclusterutil command line utility.

Click the SAP ESP Authoring tab at the top of the Studio main window to see the Data Services View. If the Data Services View is not visible, from the Studio main menu select Window Show View Other SAP Event Stream Processor Data Services. The Data Services View is automatically populated with the connections you create to remote servers in the Server View, which is visible in the SAP ESP Run-Test perspective.

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 ASE database through JDBC and to SQL Server through ODBC. At run time, the adapter obtains a connection from the database manager based on the properties in the data service, and executes queries over it.

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 1.0.73.

<table>
<thead>
<tr>
<th>Adapter</th>
<th>Supported Drivers</th>
<th>Supported Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP® Adaptive Server® Enterprise (ASE) Output Adapter</td>
<td>Open Client™</td>
<td>SAP ASE</td>
</tr>
</tbody>
</table>
| Database Input and Output Adapters | JDBC | • SAP ASE  
• IBM DB2  
• Oracle  
• Kx Systems KDB+  
• Microsoft SQL Server  
• SAP HANA® |
| Database Input and Output Adapters | ODBC | • SAP ASE  
• SAP® IQ  
• SQL Anywhere®  
• IBM DB2  
• Oracle  
• Microsoft SQL Server  
• TimesTen  
• MySQL 5.x  
• PostgreSQL  
• SAP HANA |
<p>| SAP HANA Adapter | ODBC | SAP HANA |</p>
<table>
<thead>
<tr>
<th>Adapter</th>
<th>Supported Drivers</th>
<th>Supported Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP IQ</td>
<td>ODBC</td>
<td>SAP IQ</td>
</tr>
</tbody>
</table>

On UNIX systems, SAP recommends upgrading to version 2.3.1 or later of unixODBC. If you are using a version lower than 2.3.1, 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.

- For 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`.

### 3.1 Discovering a Service

Use the Data Services View to discover databases or tables from a data service.

**Prerequisites**

Connect to any clusters that you are going to be working with in the Data Services View.

**Context**

In order to use discovery, start a local cluster in the background during ESP perspective activation. When you start the local cluster you will see projects that may have been run before in other Studio sessions, because of the persistence feature.

If you have database definitions in a service.xml file from previous versions of ESP, they will need to be migrated to the data services format using the `service.xml` migration command of the `streamingclusterutil` command line utility.

If you want to activate remote discovery on a separate server you will need to modify the cluster configuration on the remote server.

**Procedure**

1. Right-click the database.
2. Select Discover to activate schema and key discovery.

Note
Dependent on a combination of database, connectivity and adapter details, some tables do not support schema discovery. In this case, key discovery results only will be displayed.

Note
Activation of cluster discovery services may be in progress when you try to perform a discovery operation. If this is the case, check the SAP ESP Run-Test perspective and make sure that the local host is connected.

Note
You may be prompted to set an SAP HANA service entry if it has not already been set.

3. Double-click, or right-click and select Discover, to drill down.

Note
Make sure that the Data Service has the ODBC Driver Library set correctly. Always use streamingdbodbc_lib.dll for Windows (regardless of version). On other platforms, use libstreamingdbodbc64_lib.so if you are working with the SAP HANA adapter. If not, the choice of driver library depends on the size of SQLLEN in the driver manager. If SQLLEN is 4 bytes, use libstreamingdbodbc_lib.so. If SQLLEN is 8 bytes use libstreamingdbodbc64_lib.so.

3.2 Loading a Data Service

Load the data services which already exist for a particular server.

Prerequisites

Connect to any clusters that you are going to be working with in the Data Services View.

Context

The Data Services View is automatically populated with the connections you create to remote servers in the Server View, which is visible in the SAP ESP Run-Test perspective.
Procedure

1. Right-click the server.
2. Select Load Services.
   All existing data services are listed.

3.3 Importing a Data Service

You can migrate database services defined in the service.xml file of previous releases of ESP, using the service.xml migration command of the streamingclusterutil command line utility.

Context

Use the -s command of the streamingclusterutil command line utility to convert database service definitions from service.xml files to data services.

Procedure

1. Specify the path to the legacy file and to the new output file.
2. Specify the path, password and alias (if applicable) of the keystore file for decrypting parameters in the legacy service file.
3. Specify the path to the cipher key file for encrypting parameters in the new service file.

3.4 Adding a JDBC Connection to an External Database

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

Prerequisites

- Obtain the third-party JDBC .jar files from a given vendor and copy them to your $STREAMING_HOME/libj directory. Ensure that the driver is saved to $STREAMING_HOME/libj before attempting to use it.
- Create the connection to the remote server in the Server View in the SAP ESP Run-Test perspective ofStudio.
Context

To set up a JDBC connection from within Event Stream Processor, use the Data Service View in the SAP ESP Authoring perspective of Studio.

Procedure

1. In the Studio Data Services View, highlight the server on which you want to create a new database service definition.
2. Right click the server.
3. From the drop down menu, select Add Service.
   The Property sheet displays.
4. Set the Name parameter to a unique service name for the database service. This name is:
   - unique
   - 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.
5. From the drop down list, select the Driver.
   

   **Note**
   If you select Other JDBC Database Connection you are not required to provide the Host and Port, or Database/Instance, nor will you be able to Enable as HANA Reference Service.

6. Set the User parameter to the user name that you want to use when communicating with the external database.
7. Set the Password parameter to the password for your user name.
8. (Optional) Add a description of the database service in the Description parameter.
9. Define your connection using one of two methods:
   a. Enter values for the Host and Port parameters.
   b. Enter values for the Connection String parameter.
   

   **Note**
   If a connection string is provided, Host and Port values will be ignored.

10. (Optional) Enter a value in the Connection Pool Size.
11. (Optional) Provide the name of the Database or Instance.
12. Select Enable as HANA Reference Service to set this database service as a HANA service entry.
   By doing so, if you later wish to establish a reference to a table in an external HANA database, using the CREATE REFERENCE statement, then this database service will be available for that purpose.
13. Click anywhere outside the Property sheet to save your work.
3.5 Adding an ODBC Connection to an External Database

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

Prerequisites

To create a database service definition on an external server:

- Install an ODBC Driver and ensure that you have linked to your ODBC manager library.
- Create the connection to the remote server in the Server View in the SAP ESP Run-Test perspective of Studio.

Context

To set up an ODBC connection from within Event Stream Processor, use the Data Service View in the SAP ESP Authoring perspective of Studio.

Procedure

1. In the Studio Data Services View, highlight the server on which you want to create a new database service definition.
2. Right click the server.
3. From the drop down menu, select Add Service.
   The Property sheet displays.
4. Set the Name parameter to a unique service name for the database service. This name is:
   - unique
   - 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.
5. From the Driver drop down list, select streamingdbodbc_lib or streamingdbodbc64_lib.
6. Set the **User** parameter to the user name that you want to use when communicating with the external database.

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

8. (Optional) Add a description of the database service in the **Description** parameter.

9. Set the **ODBC 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.

10. (Optional) Select **Enable as HANA Reference Service** to set this database service as a HANA service entry. By doing so, if you later wish to establish a reference to a table in an external HANA database, using the **CREATE REFERENCE** statement, then this database service will be available for that purpose.

11. Click anywhere outside the Property sheet to save your work.

---

If the data service cannot be saved (because, for example, fields contain missing or incomplete information), you will receive notification through the Authoring Console.

### 3.6 Adding an OCS Connection to an External Database

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

**Prerequisites**

To create a database service definition on an external server, first create the connection to the remote server in the Server View in the SAP ESP Run-Test perspective of Studio.

**Context**

To set up an OCS connection from within Event Stream Processor, use the Data Service View in the SAP ESP Authoring perspective of Studio.
Procedure

1. In the Studio Data Services View, highlight the server on which you want to create a new database service definition.
2. Right click the server.
3. From the drop down menu, select Add Service.
   The Property sheet displays.
4. Set the Name parameter to a unique service name for the database service. This name is:
   ○ unique
   ○ 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.
5. Set the User parameter to the user name that you want to use when communicating with the external database.
6. Set the Password parameter to the password for your user name.
7. Enter details about the Host and Port.
8. (Optional) Add a description of the database service in the Description parameter.
9. (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.
10. (Optional) Set the Application Name parameter to help identify Open Client database connections used by the ASE Output adapter.
    See the CS_APPNAME connection property in the Open Client documentation for more information.
11. Click anywhere outside the Property sheet to save your work.

Note
If the data service cannot be saved (because, for example, fields contain missing or incomplete information), you will receive notification through the Authoring Console.

3.7 Updating a Data Service

Modify an existing database service definition.

Prerequisites

Connect to any clusters that you are going to be working with in the Data Services View.
Procedure

1. In the Studio Data Services View, select the data service to be modified.
   The Property sheet displays.
2. Revise the fields, as appropriate.
3. Click anywhere outside the Property sheet to save your work.

   **Note**
   If the data service cannot be saved (because, for example, fields contain missing or incomplete information), you will receive notification through the Authoring Console.

### 3.8 Copying a Data Service

Copy a database service definition from one server to another server.

**Procedure**

1. In the Studio Data Services View, right-click the data service.
2. Select Copy.
3. Select a Target Server from the list of all available hosts.
4. Click OK.

   The system checks the name of all existing services on the target host to ensure that the name of the copied service will be unique to the target host. If prompted, provide a new name for the service.

   Protected values stored on the data service are not copied. If prompted, enter new protected values in the list of properties.

### 3.9 Deleting a Data Service

Remove an existing database service definition.

**Context**

**Note**
If you are currently working with a table or schema which you have discovered using this data service, once you delete the data service the discovered element will no longer be available.
Procedure

1. In the Studio Data Services View, right-click the database service definition to be deleted.
2. From the drop-down menu, select Delete.

3.10 Configuring to Support HANA Failover

Add connection information for all the additional nodes in the SAP HANA cluster.

Context

If the SAP HANA database you are connecting to is running on a SAP HANA cluster, there is the possibility of a failover event. To configure ESP to support SAP HANA failover, you must add connection information for all the other nodes in the cluster to the existing data service configuration.

Procedure

1. In Studio, click Window ➤ Show View ➤ Other. The Show View window is displayed.
2. In the Show View window, expand the SAP Event Stream Processor entry, select Data Services, and click OK. The Data Services view is displayed.
3. In the Data Services view, right click on the name of the SAP HANA cluster.
4. Select the data service that connects to the SAP HANA database.
5. If the Properties window specifies that your existing data service uses JDBC, go to the Connection String field. For each additional node in the cluster, enter a semicolon, followed by the fully qualified domain name of the host, a colon, and the port number to use when connecting to the SAP HANA database.
6. If the Properties window specifies that your existing data service uses ODBC, go to the ODBC DSN field and obtain the location of the ODBC DSN.
   a. Open the ODBC DSN.
   b. Locate the connection for the SAP HANA database.
   c. For each additional node in the cluster, enter a semicolon, followed by the fully qualified domain name of the host, a colon, and the port number to use when connecting to the SAP HANA database.
3.11 Set up an ODBC Driver Manager

An ODBC driver manager is required for using adapters and other connections that rely on ODBC.

Context

The ODBC driver manager unixODBC 2.3.1 is required for use with ESP, but is not included with the ESP installation package. If you do not have this ODBC driver already installed, obtain it from your Linux provider, or from http://www.unixodbc.org/download.html.

Procedure

1. If you have not already, install the unixODBC driver manager. By default this is installed to the /usr/local/lib folder.
2. If libodbc.so.1 is not available in that folder, create a symbolic link from libodbc.so.1 to libodbc.so.2 (or to libodbc.so.3, depending on which version of UnixODBC you installed).

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

   This link is required because ESP connects to libodbc.so.1, which unixODBC 2.3.1 has renamed libodbc.so.2. With the symbolic link, ESP will now use libodbc.so.2.
3. Create the .odbc.ini file.
4. Update LD_LIBRARY_PATH by adding:

   ```
   LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
   export LD_LIBRARY_PATH
   ```

   LD_LIBRARY_PATH is set where libodbc.so.1 is available. In this example it is located at /usr/sap/home/unixodbc2.3.1/lib. If you have libodbc.so.2 or libodbc.so.3 instead of libodbc.so.1, create a file called libodbc.so.1, link it to libodbc.so.2 (or libodbc.so.3), and set LD_LIBRARY_PATH as follows:

   ```
   LD_LIBRARY_PATH=/usr/sap/home/unixodbc2.3.1/lib:$LD_LIBRARY_PATH export LD_LIBRARY_PATH
   ```
4 Configuring ESP Server Processes to Run as Windows Services

If you are running SAP Event Stream Processor in Windows, you can configure server processes to run as Windows services to eliminate the need to restart the processes each time you log into Windows. This also facilitates automatic restart in situations where the Windows machine is rebooted.

Prerequisites

- To ensure you start the executable or batch file from the correct location, edit the corresponding file by adding `cd %STREAMING_HOME%\<file name>` to the beginning of the file. For example, for the Web services provider: `cd %STREAMING_HOME%\wsp.bat`.
- Open a command prompt window using the Run as administrator option.

Context

The following instructions use the Web services provider (WSP) as an example, but you can apply the same steps to other executables such as `streamingclusternode.exe` and `dbsrv16.exe` (which runs the cluster database), or any other server process you want to run as a service.

Only perform this procedure if you are an administrator and have experience with advanced Windows functionality such as editing the registry.

Procedure

1. In the command prompt window, enter the following to create the Windows service:
   ```bash
   instsrv.exe <service_name> srvany.exe
   ```
   where `<service_name>` is the name you assign to the Windows service you are adding. These instructions use the service name `Streaming_WSP` as an example.

2. Open the registry editor to add the service.
   a. Within the registry editor, navigate to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Streaming_WSP`.
   b. Right-click `Streaming_WSP` and select New > Key.
   c. Name the new key `Parameters`.
   d. Right-click the new `Parameters` key and select New > String Value.
   e. Enter `Application` as the new string value.
f. Double-click the new Application entry, and for the Value Data, add the full path to the batch file or executable you want to run as a service. For example, c:\ESP51-SP09\ESP-5_1\wsp\wsp.bat.

3. From the command prompt window, enter the following to start the service:

   sc start <service name>, for example, sc start Streaming_WSP.

   In the console, the status will initially appear as "start pending."

4. To verify that the service is running, open the Services view from your Windows control panel and view the service’s status.
5 Troubleshooting

Resolve problems with SAP 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 [page 7].

5.1 Login and Connection Problems

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

5.1.1 Error: Invalid Login Credentials

Problem: You cannot log in; when you try, SAP 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? SAP Event Stream Processor supports a variety of authentication providers, all of which require configuration. To identify the configuration method or methods enabled on this node, open STREAMING_HOME/cluster/config/<cluster-name>/cluster.xml and look in the <Security> element. Compare the <Authenticator> elements here with the examples located at STREAMING_HOME/cluster/config/<cluster-name>/auth_*.xml.
- 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...
5.1.2 Prompted for Local Cluster Password

Problem: While connecting to the Studio local cluster from a component outside of Studio (such as command line tools, external adapters, or custom applications), you are prompted for a password.

Context

Solution: The Studio local cluster password is system-generated by default. In order to enter this password, you will first have to change it.

Procedure

1. In Studio, open the SAP ESP Run-Test perspective.
2. In the Server View, stop any projects running in the local cluster.
3. Right-click the local cluster, and select Disconnect Server.
4. Right-click the local cluster, and select Change Username and Password.
5. Enter the new values for the local cluster user name and password. Ensure that you make note of the new values as you will be prompted to enter them every time you connect to the local cluster with a component from outside of SAP ESP Run-Test perspective.
6. Right-click the local cluster, and select Connect Server.

5.1.3 Cannot Connect to the Cluster

When running a project, you cannot connect to the SAP Event Stream Processor cluster.

Context

Solution: If there is a single database server running the cluster database, and the server goes offline, you will not be able to connect to the cluster, particularly if authorization is enabled.

Procedure

1. Check the status of the SAP Event Stream Processor server, and restart if necessary.
2. If necessary, restart the cluster database using the dbstart command.
Note

If authorization is enabled on the cluster database, in the future, try running SQL Anywhere HA option for the cluster database.

3. If you cannot restart the cluster database,
   a. Manually terminate the projects and the cluster manager node using the Linux “kill” command or the Windows Task Manager.
   b. Restart the cluster database.
   c. Restart the cluster manager node.
   d. Restart all projects.

5.1.4 A Studio Project Does Not Run, Reports Login Failure

Problem: Attempts to run a project in a remote cluster fail with these errors:

Failed to connect to server "esp[s]://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 the Studio 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, see Configuring a Remote Cluster Connection in the SAP Event Stream Processor: Studio Users Guide.

5.1.5 A Utility Fails to Connect to the Server

Problem: Using a command-line utility such as streamingprojectclient or streamingsubscribe, 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 streamingprojectclient, streamingcnc, streamingconvert, streamingkdbin, streamingkdbout, streamingquery, streamingsubscribe, and streamingupload.

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:

```
> streamingprojectclient -c your_user_name:your_password -p localhost:51011/your_workspace/your_project -e
```

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 SAP Event Stream Processor: Utilities Guide.

5.1.6 A Utility Fails to Connect to a Project

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

For example:

```bash
% streamingprojectclient -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 `streamingclusteradmin` to find out and to start the project if necessary.

5.1.7 Cannot Start ESP Cockpit Due to Configuration Errors

Problem: ESP Cockpit cannot start due to errors in configuration, or you do not have ESP Cockpit installed and need to edit cluster configuration.

Context

Solution: Use the `streamingclusternode` utility to manually edit cluster configuration.

Procedure

1. Shut down the affected cluster.
2. From a command line, navigate to `STREAMING_HOME/bin` and launch the `streamingclusternode` utility using the `--show` option and the cluster bootstrap file.

   ```bash
   streamingclusternode --config <cluster-example>.cfg --show
   ```

   The default cluster bootstrap file is `cluster.cfg`.

3. Copy the cluster configuration to the display, in XML format.
4. Make any required changes to the cluster configuration, then save it as an `xml` file.
5. Navigate to `STREAMING_HOME\cluster\config\<cluster_name>` and replace `cluster.xml` with the new cluster configuration file.

6. Relaunch the `streamingclusternode` utility using the `--deploy` option.

   ```
   streamingclusternode --config <cluster-example>.cfg --deploy --config-type file --file <cluster_example>.xml
   ```

   The command deploys the file to the database, replacing any previous cluster configuration.

7. Restart ESP Cockpit.

5.1.8 An External Adapter Fails to Start

Problem: Attempts to start an external adapter fail.

Context

You encounter an error message like this when attempting to run an external adapter:

```
Failed call to:https://<ESP hostname>:61308/RPC2 (Failed to read server's response: 
<ESP hostname>) java.io.IOException: Failed call to:https://<ESP hostname>:61308/
RPC2 (Failed to read server's response: <ESP hostname>)
```

This error is an example of the Event Stream Processor server not being resolved.

Solution: Use the `ping` command to verify that the hostname of the server to which you are trying to connect can be resolved. If the hostname cannot be resolved:

Procedure

1. Determine the IP address of the host on which the server is running. For example, if you want to determine the IP address of the Event Stream Processor server host, run this command from that machine:

   ```
   nslookup <ESP hostname>
   ```

2. Add the following line to `C:\Windows\System32\drivers\etc\hosts (Windows)` or `/etc/hosts (UNIX):`

   ```
   <ipaddress of server hostname>        <Server hostname>
   ```
5.1.9 An Adapter Fails to Connect to a Project

Problem: When attempting to start an adapter without editing its sample XML configuration file, the adapter fails to start.

Context

The adapter may be unable to connect to the example workspace specified in the sample XML adapter configuration file if you have SSL enabled on the cluster and the URI specified in the file uses esp instead of esps. The mismatch between the cluster and adapter configuration would cause the adapter to fail to connect.

Solution: If SSL is enabled on the cluster on which you are attempting to run this adapter, follow the steps below.

Procedure

1. Ensure the URI in the adapter XML configuration file uses esps instead of esp.
2. If attempting to run one of the adapter examples provided with your installation, edit the `set_example_env.bat` or `set_example_env.sh` script file to specify:
   ```bash
   set ADAPTER EXAMPLE CLUSTER NODE_PROTOCOL=esps
   ```

5.2 Project Problems

Solve problems with projects.

5.2.1 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.
5.2.2  **A Studio Project Does Not Run, Reports Login Failure**

Problem: Attempts to run a project in a remote cluster fail with these errors:

Failed to connect to server "esp[s]://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 the Studio 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, see *Configuring a Remote Cluster Connection* in the *SAP Event Stream Processor: Studio Users Guide*.

5.2.3  **A Utility Fails to Connect to a Project**

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

For example:

```bash
% streamingprojectclient -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 `streamingclusteradmin` to find out and to start the project if necessary.

5.2.4  **A Project Runs in the Wrong Cluster**

Problem: One or more projects are running on nodes that you thought belonged to another SAP 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:

```xml
<Cache>
  <Host>dino</Host>
  <Port>19001</Port>
  <Name>test-name-1</Name>
  <Password>test-password-1</Password>
  <Managers enabled="true">
    <Manager>dino:19001</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.)

5.2.5 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.

5.2.6 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 SAP Event Stream Processor: Studio Users Guide for information on editing the project configuration. The default location of the CCR file on the Studio machine is the same as the default location for the CCR file.

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.
5.2.7 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 SAP Event Stream Processor: Developer Guide or the SAP Event Stream Processor: 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 stream or window receives all the data processed by that stream or window even if the client is not connected when the data is produced. GD is supported on streams and windows (not on delta streams) and each GD stream or 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 SAP Event Stream Processor: Developer Guide or the SAP Event Stream Processor: Studio Users Guide for information on guaranteed delivery. See the SAP Event Stream Processor: Studio Users Guide for information on PSP.
5.2.8 Cannot Publish or Subscribe to a Project that Seems to be Running

Your project appears to be running, but you cannot publish or subscribe to it.

Context

Solution: If there is a single database server running the cluster database, and the server goes offline, you will not be able to connect to the cluster, particularly if authorization is enabled.

Procedure

1. Check the status of the database server, and restart if necessary.
2. If necessary, restart the cluster database using the dbstart command.

   **Note**
   
   If authorization is enabled on the cluster database, in the future, try running SQL Anywhere HA option for the cluster database.

3. If you cannot restart the cluster database,
   a. Manually terminate the projects and the cluster manager node using the Linux “kill” command or the Windows Task Manager.
   b. Restart the cluster database.
   c. Restart the cluster manager node.
   d. Restart all projects.

5.2.9 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 in \STREAMING_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, STREAMING_HOME/cluster/nodes/<nodename>/node-name>.xml.

3. Replace the Manager element with this code:

```xml
<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.
Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP’s gross negligence.

Accessibility

The information contained in the SAP documentation represents SAP’s current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of wilful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP’s gross negligence or wilful misconduct. All links are categorized for transparency (see: http://help.sap.com/disclaimer).