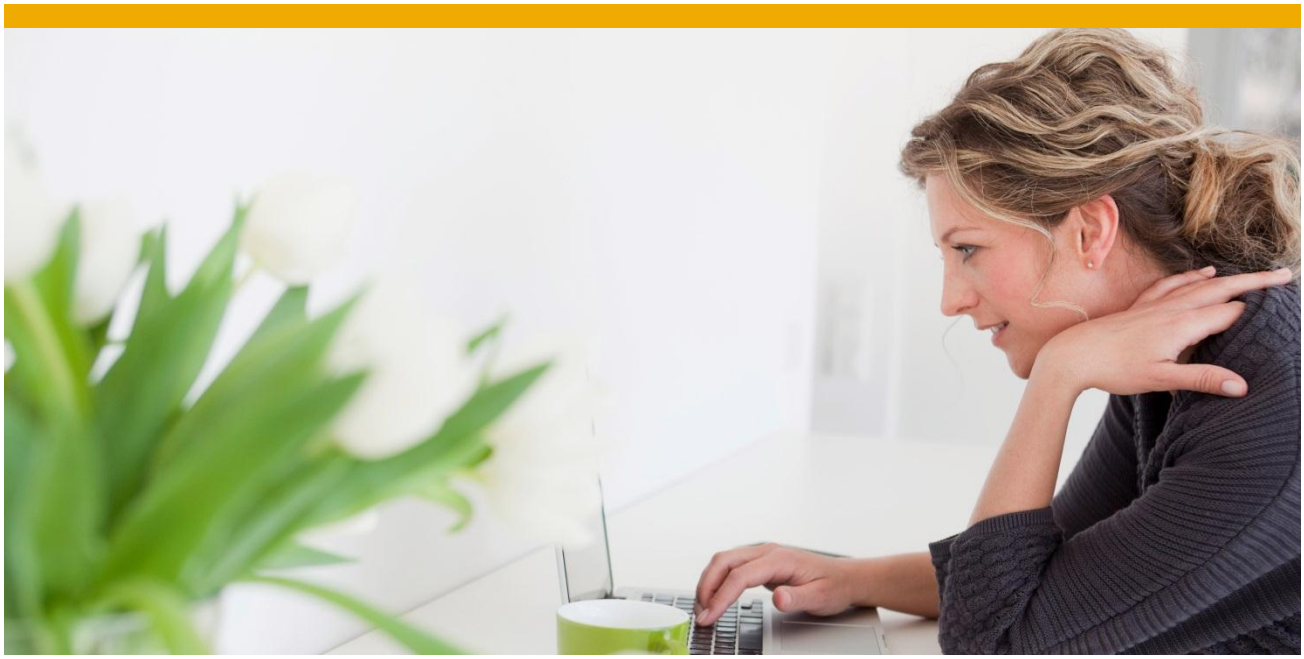

Considerations for High Availability with ESP

TABLE OF CONTENTS

This document describes considerations for high availability for an ESP system. A typical system may consist of multiple ESP servers and projects, data feeds written as external input adapters, bindings between projects, connections to databases, external output adapters, and connections to other external systems.

This document covers the considerations of high availability when one of the following components fails:

- A database node
- An external ESP input adapter
- An ESP server/project
- An external output adapter
- An external system, such as a queuing system

The information in this document is based on the behaviour of ESP 5.1 SP03.

WHAT IF A DATABASE NODE CRASHES?

If a database node crashes or becomes unreachable, any ESP projects that had a connection to the node will lose the connection. Whether a connection gets re-established depends on which adapter is being used.

HANA Output Adapter

The ODBC data source used by the HANA Output Adapter specifies the host and port of the HANA master node. In the event that a connection to the master node goes down, ESP will attempt to re-establish a connection and resend the current operation.

Note: Connection re-establishment requires ESP 5.1 SP02.

If the current operation was an array insert, the adapter will resend the array insert after reconnecting. Since the commit for insert operations occurs every “bulkBatchSize” records (see the parameters for the HANA Output Adapter) and not for each insert array, the insert arrays sent prior to the database connection going down are lost when the original transaction rolls back. The adapter resends the current insert array, not the entire batch. To avoid losing any records in the event of database failure, the “bulkBatchSize” and “bulkInsertArraySize” parameters on the adapter should be set to the same value. This will cause a commit to be issued for each insert array.

Similar action is taken in the event that the adapter was processing an array of delete or update statements at the time the database connection was lost. The transaction that was in progress rolls back. ESP will re-establish the connection and resend the current array.

The adapter will try several times to re-establish the connection. The number and frequency of the reconnect attempts can be controlled using the “reconnectAttemptDelayMSec” and “maxReconnectAttempts” properties on the adapter.

To ensure that the adapter can always re-establish a connection, multiple HANA nodes should be run. The ODBC data source for HANA allows specification of secondary nodes to use in the event that a connection cannot be made to the primary node. It is recommended that all master and standby nodes be listed in the ODBC data source. To list multiple nodes, specify a semi-colon separated list of host and port values.

```
Servernode=hanahost1:30342;hanastandbyhost:30315
```

The ODBC driver itself may hold connections to multiple HANA nodes for parallel processing within the HANA cluster. If any of these connections fail, the ODBC driver will return a connection error to ESP. The HANA standby node will take over for the failed node, and ESP will try to re-establish its ODBC connection.

Any applications using data in the HANA database should also specify master and standby nodes for any client connections (ODBC, JDBC, etc.).

IQ Output Adapter

The IQ Output Adapter uses ODBC to load files into IQ. The ODBC data source used by the IQ Output Adapter specifies the host and port of a writer node. In the event that a connection to the node goes down, ESP will attempt to re-establish the connection and resend the current operation.

Note: Connection re-establishment requires ESP 5.1 SP02.

The adapter will try several times to re-establish the connection. The number and frequency of the reconnect attempts can be controlled using the “reconnectAttemptDelayMSec” and “maxReconnectAttempts” properties on the adapter. If the adapter fails to reconnect after the configured number of attempts, the file being processed is left in the file system and the adapter goes into a “dead” state. The file can be manually loaded when the database is back up, or can be loaded by running the IQ Output Adapter in “recoverOnly” mode. See the IQ Output Adapter parameters for more information about “recoverOnly” mode.

To ensure that the adapter can always re-establish a connection, multiple IQ writer nodes should be run. The ODBC data source for IQ allows specification of secondary writer nodes to use in the event that a connection cannot be made to the primary node. To list multiple nodes, specify a comma separated list of host and port values.

```
LINKS=TCPIP(HOST=Server1:1234,Server2:5678);
```

Database Input Adapter (DB-In)

The DB-In adapter executes a query on a table, iterates through the result set, and sends rows as records into a stream or window. If the database connection is down when the adapter executes the query, the adapter will make one attempt to re-establish a connection and execute the query. If that attempt fails, the adapter moves to the “dead” status. If the database connection fails after the query has been executed and while the adapter is iterating through the result set, the adapter moves to the “dead” status. There is no attempt to re-establish the connection in this case and the adapter should be restarted.

Depending on the project, restarting the adapter may cause duplicate rows to be sent through the project. If the DB-In adapter is being used to do a one-time load of reference data into a window for data enrichment or validation (i.e. a join or look-up) and not as a source of incoming events, then there are no undesirable side effects when the adapter is restarted. However, if the database table is being used as a source of events, restarting the adapter may cause the same records in the table to be sent a second time when the adapter restarts.

To ensure that the database reconnection attempt is successful, multiple database nodes should be run and the ODBC or JDBC configuration should specify multiple server host/port values.

Database Output Adapter (DB-Out)

The DB-Out adapter does not contain logic to recover database connections. If the database connection goes down, the transaction for the record being processed rolls back and the adapter moves into a “dead” state. If batch processing is enabled and the connection is down when it is time to process the batch, the entire batch is discarded when the transaction rolls back and the adapter goes into a “dead” state. In order to re-establish a database connection, the adapter must be restarted.

If the adapter is configured to truncate the output table at start-up, the table will be truncated when the adapter is restarted. Any records stored in the table prior to the database failure will be deleted by the truncation operation.

ASE Output Adapter

The ASE Output Adapter does not try to re-establish a connection upon failure. The adapter must be restarted in order for the adapter to re-open a connection to ASE. Any records that arrive after the database connection goes down will be considered bad and are discarded by the adapter.

To define an Open Client connection to a highly available multi-server ASE system, specify a “hafailover” line in the interfaces file. For more information, see the Open Client Configuration Guide.

WHAT IF A DATA FEED CRASHES?

The first question to ask regarding the data feeds is, “How highly available is highly available?” Is it acceptable for one of the data feed input adapters to crash and take 60 seconds to restart or does each data feed adapter need to be truly highly available?

If it is acceptable for a data feed to be down for the amount of time it would take to restart it, then the data feed just needs to be designed such that, upon restart, it knows how to pick up where it left off. It needs to re-establish its connection to the source of data, it needs to re-establish its publishing connection to the ESP cluster, and it needs to know where it left off.

If the data feed needs to be truly highly available, then the data feed should run as an active-passive pair. The passive data feed would need a means of knowing when the active data feed has failed. There would need to be some mechanism for the pair to know when one has gone down (heartbeat over a socket, shared file or database table to which heartbeats are written) and some mechanism for the passive data feed to know how to pick up where the active data feed left off.

WHAT IF AN ESP SERVER/PROJECT CRASHES?

When a project fails, any connections in to or out of the project will be lost. These connections may include external adapters that are publishing data using the ESP Publisher SDK, project-to-project bindings, connections to destination systems (for example, databases or queuing systems) and applications subscribing to stream data using the ESP Subscriber SDK.

Project Failover

If the project is deployed with failover enabled (<Failover enable="true"/> in the project’s CCR file), ESP will restart the project on another ESP node. The “Failures per interval” and “Failure Interval” settings in the project’s CCR file control how often a project can failover.

“Failures per interval” specifies the number of restart attempts within a given interval. “Failure interval” specifies the time, in seconds, of the length of the interval. If left blank, the interval is infinite. For example, to allow the project 5 restart attempts within 2 minutes, specify an interval of 300 seconds and failures per interval of 5.

If a project needs to run on particular nodes, the project’s CCR file can specify Controller Affinities. To support failover, yet still specify affinities, specify multiple weak controller affinities. For example, if there are four controller nodes in the cluster, but the project must run on either node 3 or node 4, and must support failover, define two weak affinities for nodes 3 and 4.

It is important that projects be designed such that they can withstand a restart without any ill effects. Depending on the project, restarting the project may cause duplicate rows to be sent through the project. For example, suppose the project reads a CSV file. If the file is being used to do a one-time load of reference data into a window for data enrichment or validation (i.e. a join or look-up) and not as a source of incoming events, then there are no undesirable side effects when the project is restarted. However, if the file is being used as a source of events, restarting the project may cause the same records in the file to be sent a second time when the project restarts.

Cluster managers use heartbeats to know if a project is alive. If a cluster manager does not receive periodic heartbeats from a project, the cluster manager will conclude that the project has crashed and will start the

project on another node. If a server is too busy to respond, the cluster manager will erroneously conclude that the project has crashed and restart the project. It is important to set the <ApplicationHeartbeatTimeout> value in the node configuration file (for example, node0.xml) to be sufficiently long so as to avoid this and to ensure that the capacity of the hardware on which the project is running is sufficient to allow some capacity for the server to send heartbeats.

Publishers and Subscribers

For any publisher or subscriber connection to ESP, the publisher code should specify multiple ESP cluster manager nodes in the connection options. Specifying multiple cluster manager nodes will allow the ESP SDK to contact a secondary node in the event that the primary node fails. The SDK will only return an error if a connection cannot be established to any of the managers specified in the connection options. Multiple managers can be specified using a semi-colon separated list in the URI.

```
esp[s]://host1:port1;host2:port2[/workspace/project/stream]
```

If working in callback or select access modes, the SDK can be configured with additional levels of tolerance for loss of connectivity. In these modes, if no manager can be contacted, the SDK does not disconnect an ESP server instance. Instead, it generates an ESP_SERVER_EVENT_STALE event. If the SDK manages to reconnect after a (configurable) number of attempts, it generates an ESP_SERVER_EVENT_UPTODATE. Otherwise, it disconnects and generates an ESP_SERVER_EVENT_DISCONNECTED event. In addition, if a project goes down, an ESP_PROJECT_EVENT_STALE event is generated. If the SDK is able to reconnect to the project (likely after it restarts), it generates an ESP_PROJECT_EVENT_UPTODATE event. Otherwise, it generates an ESP_PROJECT_EVENT_DISCONNECTED event.

Project-to-Project Bindings

When a project is restarted, bindings from the restarted project to other projects are established. Also, any bindings from other projects to the restarted project are re-established.

Suppose there are two projects called Producer and Consumer. Consumer has a binding to a stream (not a window) in Producer. Producer and Consumer have failover enabled.

Suppose Consumer crashes. A cluster manager restarts Consumer. Consumer re-establishes its binding to Producer's stream. Any records that arrived at Producer’s stream while Consumer was down and being restarted will not be delivered to Consumer. If Consumer had a binding to a window in Producer instead of a stream, Consumer would receive the contents of the window when the binding is re-established and would then receive any new records arriving at the window.

Suppose Producer crashes. Consumer loses its binding to Producer and will periodically try to re-establish the binding. A cluster manager restarts Producer. Consumer re-establishes its binding to Producer's stream. Any records arriving at the Producer stream prior to the binding being re-established (records arriving between the time the Producer starts and the time the Consumer issues its next retry) will not be delivered to Consumer. If Consumer has a window-to-window binding, then the contents of the Consumer window are removed when the binding connection drops, and the contents of the Producer window are resent to Consumer when the binding is re-established.

If the Consumer had a binding to a window attached to a stream binding seem to get wiped out and the all the data is sent again, to me that resend of the data is a more difficult situation. You already have the data in Consumer (it stayed up), and the fact you can't setup the binding to only send deltas...

To establish a binding, ESP needs to contact a cluster manager. For high availability, there should be at least two cluster managers in any cluster, running on separate hardware. While establishing a binding, if ESP cannot contact the first cluster manager, ESP will try each of the cluster managers in turn. For remote clusters, the list of cluster managers needs to be specified in the <Cluster> definition in the <Clusters> area of the project configuration file (CCR).

```
<Clusters>
```

```
    <Cluster name="cluster1" type="remote">
        <Username>xxx</Username>
        <Password>xxx</Password>
        <Auth>user</Auth>
        <Managers>
            <Manager>http://localhost:19011</Manager>
            <Manager>http://localhost:19012</Manager>
            <Manager>http://localhost:19013</Manager>
        </Managers>
    </Cluster>
<Clusters>
```

If the cluster is of type “local”, the list of cluster managers can be determined from the local node and so they do not need to be listed explicitly.

```
<Clusters>
    <Cluster name="cluster1" type="local">
        <Username>xxx</Username>
        <Password>xxx</Password>
        <Auth>user</Auth>
    </Cluster>
<Clusters>
```

Database Connections

When a project is restarted, any database connections will be established as part of the adapter start-up.

Active-Active Deployments

A project deployed as active-active (<Project ha="true"> in the project’s CCR file) is run as two instances of the project that run simultaneously. One instance is elected as the primary instance. If one of the instances is already active, it is the primary instance. If a failed instance restarts, it assumes the secondary position and maintains this position unless the current instance fails or is stopped.

If the secondary instance starts and does not find the primary instance, it reattempts a connection to the primary server for 30 seconds. If it fails to successfully connect to the primary server, it becomes the primary server.

Any data published to the primary instance is automatically mirrored to the secondary instance. If the currently connected instance goes down, the SDK tries to reconnect to the alternate instance. This happens transparently. If the reconnection is successful, there is no indication generated to the user.

If subscribed to an active-active project, the SDK does not disconnect the subscription if the instance goes down. Instead, it generates an ESP_SUBSCRIBER_EVENT_DATA_LOST event. It then tries to reconnect to the peer instance. If it is able to reconnect, the SDK subscribes to the same streams. Subscription clients then receive an ESP_SUBSCRIBER_EVENT_SYNC_START event, followed by the data events, and finally an ESP_SUBSCRIBER_EVENT_SYNC_END event. Clients can use this sequence to maintain consistency with their view of the data if needed. Reconnection during publishing is also supported but only if publishing in synchronous mode. It is not possible for the SDK to guarantee data consistency otherwise.

Active-active projects are typically configured so that the two instances of the project run on different nodes. Whether the instances can run on the same node is controlled using Instance Affinities in the project’s CCR file (not to be confused with Controller Affinities in the previous section). If the instances should never run on the same node, set strong negative Instance Affinities. If the instances should not run on the same node unless they have to due to node failure, set weak negative Instance Affinities.

Because active-active projects consist of two instances of the project running simultaneously, active-active deployments are typically not suitable for systems that feed data from ESP into an external system (for

example, a database or a queuing system) unless there are also two instances of the database (or database table) or queue. Active-active projects are also not suitable for high volume systems since the load on the system doubles and the cost of hardware to support the double load is prohibitive.

WHAT IF AN EXTERNAL OUTPUT ADAPTER CRASHES?

If an external output adapter crashes, the connections used by the adapter will be re-established when the external adapter is restarted.

If the external output adapter is attached to a stream, then any records arriving at the stream while the adapter is down will be lost. If the external output adapter is attached to a window, the adapter may receive the contents of the window when the adapter is restarted depending. Some adapters allow configuration of whether to receive the contents of the window when the adapter starts. Note that receiving the contents of a window when an adapter restarts can cause duplicate records to be sent into the adapter.

For example, in the case of the JMS Output Adapter, the connection from the adapter to the ESP project is lost, as is the connection to the JMS queue. Both connections will be re-established when the JMS Output Adapter is restarted. If the JMS Output Adapter is attached to a stream, then any records arriving at the stream while the adapter is down will be lost. If the JMS Output Adapter is attached to a window, the adapter will receive the contents of the window when the adapter is restarted.

WHAT IF AN EXTERNAL SYSTEM CRASHES?

If the ESP project has a connection to an external system (for example, a queuing system) and the external system goes down, whether or not the connection will be re-established is dependent on the particular adapter being used and the behaviour of the external system.

For example, in the case of the JMS Output Adapter, suppose that the JMS queue goes down. The connection from the adapter to the queue is lost. If the adapter tries to commit a JMS transaction while the connection is down, and if the adapter is configured for guaranteed delivery, the adapter goes into the “Done” state, indicating to the project that it cannot currently receive records. If the adapter is not configured for guaranteed delivery, then any records arriving while the queuing system is down will be discarded. Note that most queuing systems provide high availability and guaranteed delivery. If these are being used, then the likelihood of the JMS Output Adapter receiving a connection error from the JMS client is unlikely.

www.sap.com