

---

# White Paper: Using WebSphere Adapter in WebSphere High Availability Environment

By

Ni Yong  
Li Dong  
Song Han  
Zhang Kai

IBM Software Group

|  |           |
|--|-----------|
| <b>1. OVERVIEW.....</b>  | <b>3</b>  |
| 1.1 WEBSHERE HIGH AVAILABILITY ENVIRONMENT OVERVIEW .....                                    | 3         |
| 1.2 WEBSHERE ADAPTER IN HA ENVIRONMENT .....   | 5         |
| <b>2. ADAPTER DEPLOYMENT AND CONFIGURATION IN HA ENVIRONMENT .....</b>                       | <b>7</b>  |
| 2.1 WEBSHERE HA ENVIRONMENT .....  | 7         |
| 2.2 STANDALONE DEPLOYMENT .....  | 7         |
| 2.2.1 <i>Standalone Deployment in WebSphere Application Server 7.0 HA Environment.....</i>   | <i>8</i>  |
| 2.2.1.1 Scenario 1 - Deploy Adapter in Cluster Level.....                                    | 8         |
| 2.2.1.2 Scenario 2 - HA configuration in Cluster-level different with it in node-level ..... | 16        |
| 2.2.1.3 Scenario 3 - Deploy Adapter only in Node Level.....                                  | 17        |
| 2.2.2 <i>Standalone Deployment in WebSphere Process Server 6.2 cluster environment.....</i>  | <i>19</i> |
| 2.2.2.1 Scenario 4 - Deploy Adapter in Cluster Level.....                                    | 19        |
| 2.2.2.2 Scenario 5 - Deploy Adapter in Node Level and generate AS or MCF manually.....       | 21        |
| 2.2.2.3 Scenario 6 - Deploy Adapter in Node Level and generate AS or MCF automatically.....  | 22        |
| 2.2.2.4 Scenario 7 - Both Node-level JNDI and Cluster-level JNDI Exists.....                 | 23        |
| 2.2.3 <i>Standalone Deployment in WebSphere Process Server 7.0 cluster environment.....</i>  | <i>24</i> |
| 2.2.3.1 Scenario 8 - Deploy Adapter in Cluster Level.....                                    | 24        |
| 2.3 EMBEDDED DEPLOYMENT .....  | 25        |
| 2.3.1 <i>Embedded Deployment in WebSphere Process Server 6.2 HA environment.....</i>         | <i>25</i> |
| 2.3.1.1 Scenario 9 - Embedded deployment in WebSphere Process Server 6.2 HA environment ..   | 25        |
| 2.3.2 <i>Embedded Deployment in WebSphere Process Server 7.0 cluster environment.....</i>    | <i>26</i> |
| 2.3.2.1 Scenario 10 - Embedded deployment in WebSphere Process Server 7.0 HA environment     | 26        |
| <b>3. ACHIEVE HIGH AVAILABILITY OF ADAPTER TRANSACTION IN HA ENVIRONMENT</b>                 | <b>27</b> |
| 3.1 HA REQUIREMENT FOR GLOBAL TRANSACTION .....  | 27        |
| 3.2 TRANSACTION SCENARIO IN HA ENVIRONMENT .....   | 28        |
| <b>4. SUMMARY .....</b>  | <b>31</b> |
| <b>5. GLOSSARY .....</b>   | <b>31</b> |
| <b>6. RESOURCES .....</b>  | <b>32</b> |

# 1. Overview

WebSphere Adapter is a connectivity-oriented middleware and is often used with runtime server such as WebSphere Process Server or WebSphere Application Server and so on. There are multiple deployment options and approaches for using an adapter in HA environments for these servers. However, so far, there is no document that consolidates these usages, suggests an approach, or provides the best practices to use adapter in these HA environments. This paper is intended for this purpose and aim to help an enterprise application developer to understand the configuration and deployment through various actual usage scenarios.

## 1.1 WebSphere High Availability Environment Overview

This paper mainly covers two kinds of WebSphere High Availability Environment, WebSphere Application Server HA environment and WebSphere Process Server HA environment. The two kinds of HA environment are based on the network deployment architecture. For example, in the WebSphere Application Server HA environment, it uses several logical components that slot together to form high availability application server cluster environments. See Figure 1.

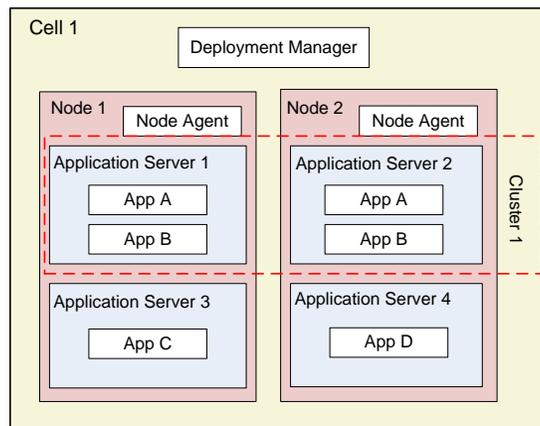


Figure 1.

The most top level unit is a cell, which is a logical grouping of nodes that are centrally managed and have access to shared resources. A node is a managed container for one or more application servers. Typically, a single node corresponds to a single machine. A node comprises a node agent and application servers. The node agent is an architectural component that enables the deployment manager of the cell to remotely manage the node. The deployment manager is an application server whose task is to manage and configure the cell.

Multiple application servers are grouped together to form a WebSphere cluster to perform the same task as a team. The member application servers can be distributed across one or more nodes in any configuration.

The cluster can provide load balancing by way of distributing the requests across the application servers in response to the individual load and availability of each server in order to prevent an individual server from being overloaded. See Figure 2.

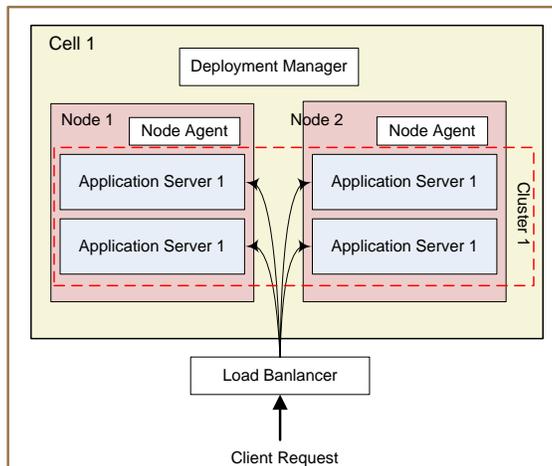


Figure 2.

Additionally, the cluster can provide failover capability to enable the high availability and business continuity. If one node fails the other node in the same cluster will take over and complete the unfinished business job and process the other business requests. This ensures that the cluster enhances the business high availability. See Figure 3.

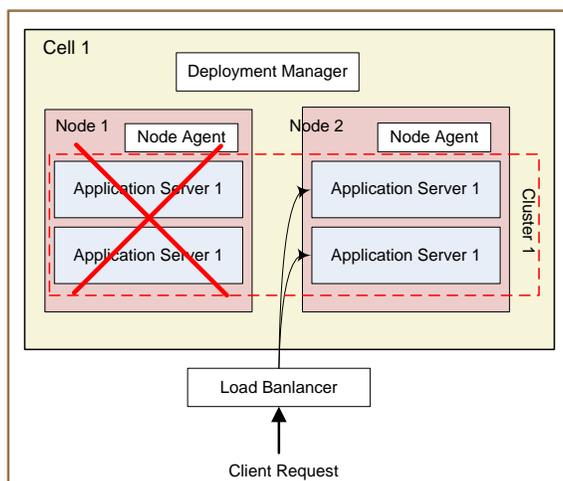


Figure 3.

Based on the cluster environment, the WebSphere Application Server ND v6 introduces a new feature called High Availability Manager (commonly called HAManager) that enhances the availability of WebSphere singleton services such as transaction or messaging services. It provides a peer recovery mechanism for in-flight transactions or messages among clustered WebSphere application servers. The HAManager runs as a service within each WebSphere process (Deployment Manager, Node Agent, or application server) that monitors the health of WebSphere singleton services. In the event of a server failure, the HAManager will failover any singleton service that was running on the failed server to a peer server. To ensure the high availability in the scope of a cell, a core group is used for grouping all of the application servers as a high availability domain. See Figure 4.

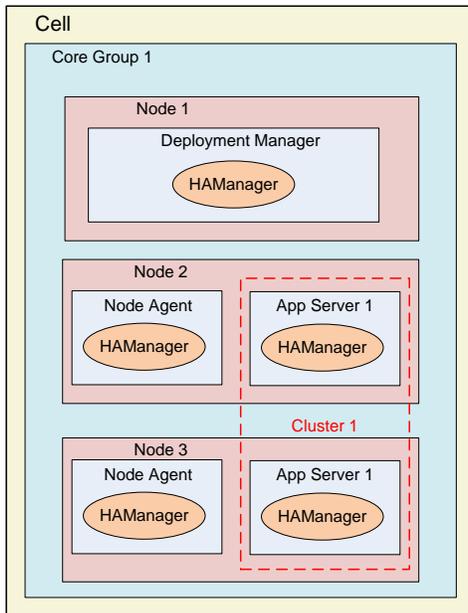


Figure 4.

A cell must have at least one core group. The WebSphere Application Server creates a default core group called DefaultCoreGroup for each cell. The multiple application servers can join the core group to host high availability service. A core group uses the policy to determine how many and which members of a high availability group are activated to accept work at any point of time. A policy manages a set of high availability groups (HA groups). HA group is a set of dynamic components created in a core group at run time. Each group represents a highly available singleton service, such as the messaging service or adapter inbound service, and so on. The available members in a group are ready to host the service at any time. There are several types of core group policies that can be used. This paper only covers the One of N policy, which determines that only one server activates the singleton service at a time and the HAManager starts the service on another server if a failure occurs.

## 1.2 WebSphere Adapter in HA environment

The IBM WebSphere Adapter portfolio provides a series of adapters based on the Java™ Connector Architecture (JCA) specification. WebSphere Adapters enable managed, bidirectional connectivity and data exchange between Enterprise Information System application and J2EE components supported by WebSphere platform. It can support multiple WebSphere platforms to provide the end-to-end integration bridge between two EIS.

The typical end-to-end integration scenario is to use two adapters to connect the two individual EIS, EIS A and EIS B. See Figure 5. The two adapters enable the integration between the generic data format used in the server business component and the application specific data format in the EIS. The data in one EIS flows into another through the adapter and the process server in bidirectional way. With the connectivity and transformation provided by adapter, the business process can focus on the business logic since the interaction and communication logic with EIS is completed by adapters and is transparent to it.

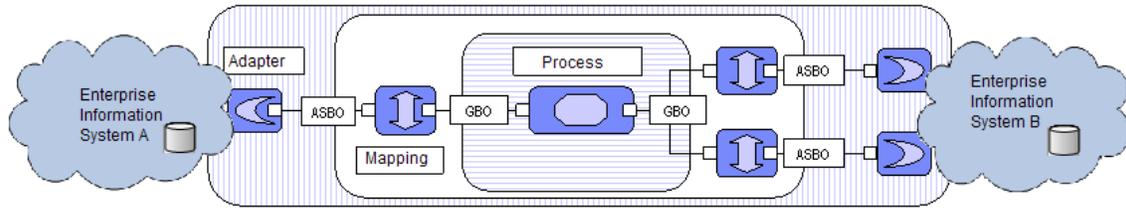


Figure 5.

In the real customer's production environment, the high availability of the enterprise solution is more and more important. In the above end to end scenario, the availability of the adapter application is critical to the whole integration solution. Whenever the adapter application is unavailable, the interaction and communication with EIS will break. This impacts the normal running of the integration solution and lead to the business break and uncountable loss for customers. So the adapter application is usually deployed to the HA environment to ensure high availability of the whole integration solution.

To make sure the WebSphere Adapter application can work with the high availability provided by the infrastructure of WebSphere HA environment, a good practice is to deploy it in cluster level and in this way it can be managed by the HA manager to provide the HA capability. What needs to pay attention together is, there're some difference between the HA behavior of adapter outbound application and that of adapter inbound application in HA environment.

When the adapter inbound application is deployed to WebSphere HA environment with HA capability enabled, there is a HA group created for the adapter inbound service. See Figure 6. One of N HA policy is used in this HA group. In this way, only one adapter inbound instance is activated and others are all in standby state. This is known as the singleton pattern for HA behavior of adapter inbound service. Applying this pattern for adapter inbound is based on the fact that it's invalid usage to enable more than one inbound instance to process the same event, which will lead to the event conflict and unpredictable errors. When the node with the active inbound instance fails, the failover occurs and the adapter inbound instance in the other node will be activated and go on processing events so as to ensure the high availability for the business. To enable the singleton mode of adapter inbound in various WebSphere HA environments, some deployment and configuration steps will be necessary, which are described in the following sections through specific scenarios.

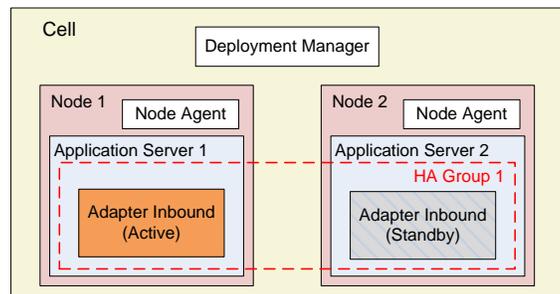


Figure 6.

When the adapter outbound application is deployed into WebSphere HA environment, it will be active in every node of the cluster. This is different with the inbound service, since there is no singleton requirement for it and thus the One of N HA policy of HA group has no impact for adapter outbound service. Every adapter outbound instance in this cluster is ready to process an event at any time when the event is delivered to it. The load balance will be enabled for the multiple adapter outbound to serve the large amounts of events and prevent the single node overload. Once some node fails then the request originally delivered to it will be automatically forwarded to other nodes to ensure the service availability.

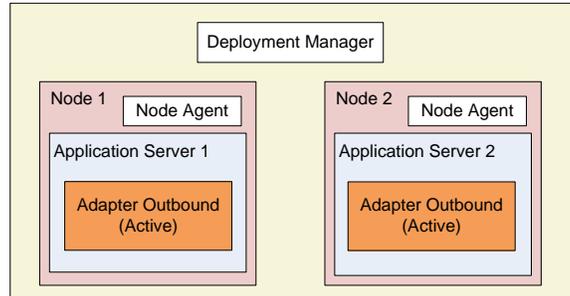


Figure 7.

## 2. Adapter Deployment and Configuration in HA Environment

This section describes how to use the standalone deployment and the embedded deployment of an adapter to achieve the correct runtime behavior as described in Section 1.2. Also, the various configuration options will be explored to clarify the runtime behavior through various specific scenarios and finally outline the best practices for depolying and configuring an adapter in Websphere HA environment.

### 2.1 WebSphere HA Environment

Among all of the scenarios, the following three kinds of Websphere HA environment are discussed here:

- WebSphere Application Server 7.0 HA environment
- WebSphere Process Server 6.2 HA environment
- WebSphere Process Server 7.0 HA environment

This section will illustrate the features and compare the differences for adapter application deployment in these HA environment. Table 1 lists the runtime and tooling version for every environment.

| <b>WebSphere HA Platform</b>                    | <b>Runtime</b>                                      | <b>Tooling</b>                      |
|---|---|-------------------------------------|
| WebSphere Application Server 7.0 HA Environment | WebSphere Application Server 7.0 Network Deployment | Rational Application Developer 7.5  |
| WebSphere Process Server 6.2 HA Environment     | WebSphere Process Server 6.2 Network Deployment     | WebSphere Integration Developer 6.2 |
| WebSphere Process Server 7.0 HA Environment     | WebSphere Process Server 7.0 Network Deployment     | WebSphere Integration Developer 7.0 |

Table 1.

Specifically, below HA runtime and tooling environment are used for scenarios in this paper. For WebSphere Application Server 7.0 HA Environment, there is one cluster with two nodes, each node includes an application server. In the following scenario description, the nodes will be named as Node A and Node B. To simplify the description, the adapter application running in the application server in Node A will simplified as “the adapter application running in Node A”. For WebSphere Process Server 6.2 HA Environment, there are three clusters, AppTarget cluster, Messaging cluster and Support cluster. Each cluster has two application servers distributed in two different nodes.

For WebSphere Process Server 7.0 HA Environment, it has the same network deployment topology with the WebSphere Process Server 6.2 HA Environment.

### 2.2 Standalone Deployment

In Standalone Deployment, the adapter RAR is installed independently before the adapter application EAR is deployed. The adapter application EAR contains no RAR in it, yet associates and uses the installed adapter RAR in runtime.

## 2.2.1 Standalone Deployment in WebSphere Application Server 7.0 HA Environment

### 2.2.1.1 Scenario 1 - Deploy Adapter in Cluster Level

JCA Adapter RAR is managed as resource at the node-level in WebSphere Application Server and WebSphere Process Server HA environment, which means the adapter RAR's life cycle and runtime behavior is managed by individual node independently in its node scope. However, the adapter inbound or outbound application runs at cluster level. Many customers choose to install adapter RAR with node level in all nodes in the cluster, and create the AS or MCF with same JNDI name and the same properties individually in every adapter RAR. Then, deploy the adapter application to refer to the JNDI name. This is the typical usage that is described in the Scenario 2. However, this way brings the uncertainty about which adapter RAR will be actually invoked by the application in runtime, and also, this way cannot make use of adapter's HA feature to ensure the high availability of customer's business. So the most suggested and the reliable way is to generate a cluster-level adapter based on the node-level adapter RAR in all nodes within the cluster, then configure AS/MCF in the cluster-level adapter and deploy adapter application referring to this AS/MCF JNDI.

#### Configuration Steps

1. Install Adapter RAR in Node A and Node B.

Open the Admin Console of deployment manager, navigate to the **Resources->Resource Adapters->Resource adapters**. Click the **Install RAR** button in the right side pane, the Install RAR File page is opened. See Figure 8.

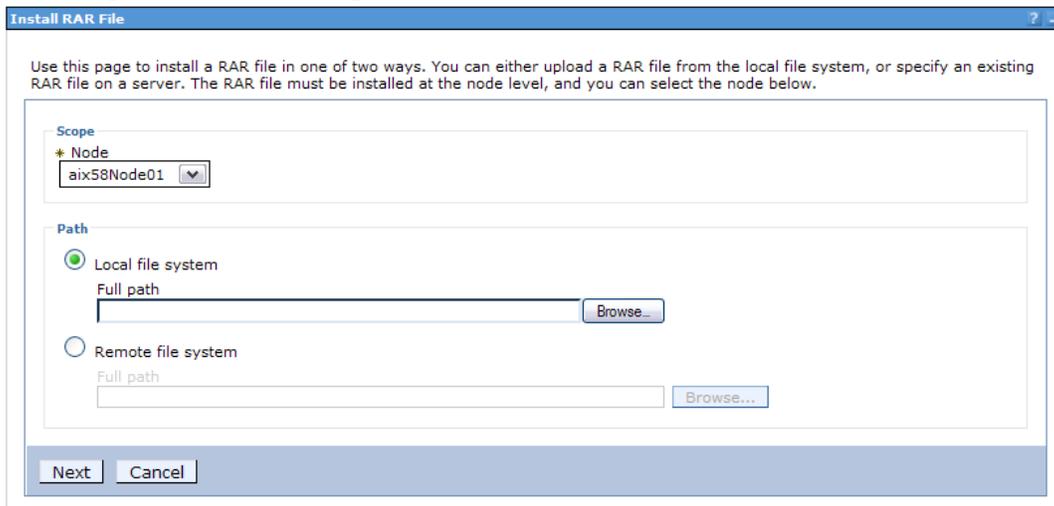


Figure 8.

Specify the target Node A in the **Node** combo box. Click **Browse** and navigate to the target RAR file. Then click **Next**. Input the name of the adapter RAR in the **Name** edit box or keep the default value for it in the configuration page. Click **OK** and save the configuration to complete the RAR installation. Then install the RAR on Node B following the same steps.

2. Generate the cluster-level adapter based on the two node-level adapters.

Navigate to the **Resources->Resource Adapters->Resource adapters**, specify the scope to the target cluster and click **New** to open the configuration page. See Figure 9.

## Resource adapters > New

Use this page to manage resource adapters, which provide the fundamental interface for connecting applications to an Enterprise Information System (EIS). The WebSphere(R) Relational Resource Adapter is embedded within the product to provide access to relational databases. To access another type of EIS, use this page to install a standalone resource adapter archive (RAR) file. You can configure multiple resource adapters for each installed RAR file.

Configuration

**General Properties**

\* Scope  
cells:was7c2Cell01:clusters:HenryCluster

\* Name  
SAPRAND

Description

\* Archive path

Choose an archive path from the list of installed RARs (recommended)

List of installed RAR files  
\${CONNECTOR\_INSTALL\_ROOT}/CWYAP\_SAPAdapter\_Tx.rar

Specify the archive path of an installed RAR

Class path

**Additional Properties**

- J2C connection factories
- J2C activation specifications
- J2C administered objects
- Advanced resource adapter properties
- Custom properties
- View Deployment Descriptor

Figure 9.

Specify the adapter name in the **Name** edit box, for example - SAPRAND. In the Archive path area, the “**Choose an archive from the list of installed RARs**” radio is selected by default. Here specify “**`\${CONNECTOR\_INSTALL\_ROOT}/CWYAP\_SAPAdapter\_Tx.rar`**”, which is the installed adapter RAR in Step 1. Notes that the RAR appears in the choice list only if the RAR is installed on all nodes in the same cluster. This ensures that the cluster level adapter has real adapter RAR in every node in the cluster. Click **OK** and save the configuration to complete the generation for the cluster level adapter. See Figure 10.

**Resource adapters**

Use this page to manage resource adapters, which provide the fundamental interface for connecting applications to an Enterprise Information System (EIS). The WebSphere(R) Relational Resource Adapter is embedded within the product to provide access to relational databases. To access another type of EIS, use this page to install a standalone resource adapter archive (RAR) file. You can configure multiple resource adapters for each installed RAR file.

Scope: =All scopes

Show scope selection drop-down list with the all scopes option

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, [see the scope settings help](#).

All scopes

Preferences

Install RAR New Delete Update RAR

| Select                   | Name  | Scope                |
|--------------------------|---|----------------------|
| <input type="checkbox"/> | <a href="#">IBM WebSphere Adapter for SAP Software with transaction support</a> | Node=was7c2Node01    |
| <input type="checkbox"/> | <a href="#">IBM WebSphere Adapter for SAP Software with transaction support</a> | Node=was7c2Node02    |
| <input type="checkbox"/> | <a href="#">SAPRAND</a>   | Cluster=HenryCluster |

Total 3

Figure 10.

3. Generate AS or MCF for cluster-level adapter.

Click the cluster level adapter in the adapter RAR table to enter the configuration page. Click the J2C connection factory to create MCF for outbound application or click J2C activation specification to create AS for inbound application. For the specific configuration steps you can refer to [1].

4. Configure the HA property for cluster-level adapter.

After the creation of MCF or AS, return to the main configuration page. Click **Custom properties**, specify the **enableHASupport** as true.

Return to the main configuration. Click **Advanced resource adapter properties**, and select **Register this resource adapter with the high availability manager**. Then specify the **Endpoint failover** option.

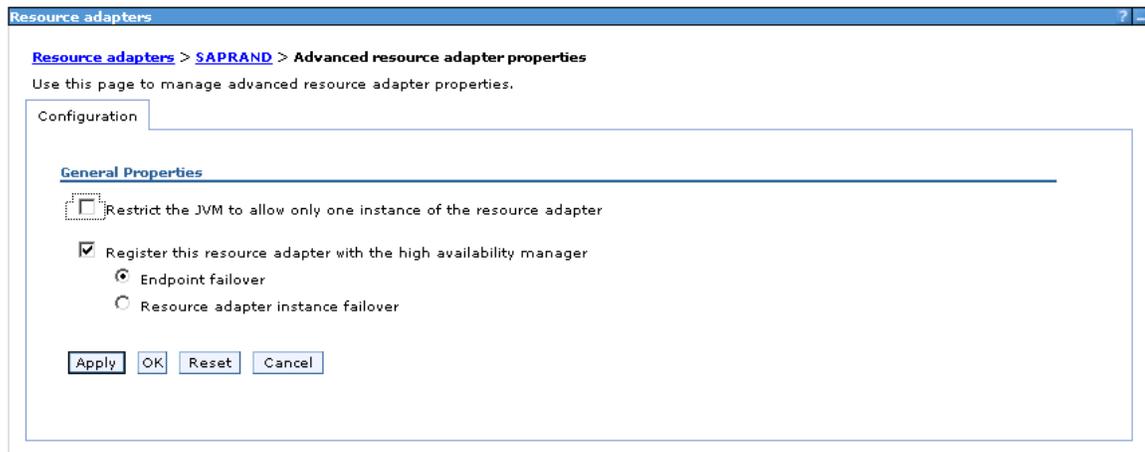


Figure 11.

Registering the resource adapter with the high availability manager specifies that the high availability (HA) manager will manage the lifecycle of a JCA resource adapter in the cluster, ensuring that applications using resource adapters remain highly available. Endpoint failover option allows only one resource adapter in an HA group to receive messages across multiple servers. The result is that only one resource adapter can have endpoints active at one time. Resource adapter instance failover option allows only one resource adapter in an HA group to be started across multiple servers.

This HA related configuration is recorded into the configuration file in the WebSphere Application Server 7.0 ND installation directory, specifically,

WAS\_Root\profiles\xxNode\config\cells\xxcell\clusters\xxcluster\resources.xml

Some of the content in this file corresponds to the above configuration.

```
<resources.j2c:J2CResourceAdapter
xmi:id="J2CResourceAdapter_1295432333031" name="SAPRAND"
isolatedClassLoader="false"
archivePath="{CONNECTOR_INSTALL_ROOT}/CWYAP_SAPAdapter_Tx.rar"
singleton="false" hACapability="RA_ENDPOINT_HA"
isEnabledHASupport="true">
```

The EnableHASupport corresponds to Register this resource adapter with the high availability manager checkbox choice. The hACapability attribute corresponds to the Endpoint failover or Resource adapter instance failover radio box choice.

#### 5. Develop Adapter Application

Develop the Adapter application with the Rational Application Developer 7.5. Specify the JNDI name to the preconfigured cluster level AS or MCF. The detailed development steps refer to [1]. Export the application as EAR file in Rational Application Developer 7.5.

#### 6. Deploy and start the application

In the Admin Console navigate to the **Applications > Application Types > WebSphere enterprise** applications. Click **Install** in the right pane, browse the exported EAR file. Complete the installation wizard to install EAR. Start the application.

#### Observations and Runtime Behavior

If the application is for inbound in this scenario, it is activated for polling events on only one node and is standby on another node. When the active node fails, the failover happens and the adapter inbound is activated on another node and go on processing events. From the trace in two

application server in the two nodes, there is some trace records reflecting the different behavior. Following is an example of the trace of server1 on Node A:

```
[1/20/11 0:00:40:000 GMT+08:00] 00000022 EJBContainerI I WSVR0037I: Starting
EJB jar: SAPI inboundEJB.jar
[1/20/11 0:00:41:062 GMT+08:00] 00000022 EJBContainerI I CNTR0167I: The
server is binding the sapinboundpkg.SapInboundInt interface of the
SapInboundIntSB enterprise bean in the SAPI inboundEJB.jar module of the
SAPI inboundEJB application. The binding location is:
ejblocal:SAPI inboundEJB/SAPI inboundEJB.jar/SapInboundIntSB#sapinboundpkg.SapIn
boundInt
[1/20/11 0:00:41:093 GMT+08:00] 00000022 EJBContainerI I CNTR0167I: The
server is binding the sapinboundpkg.SapInboundInt interface of the
SapInboundIntSB enterprise bean in the SAPI inboundEJB.jar module of the
SAPI inboundEJB application. The binding location is:
ejblocal:sapinboundpkg.SapInboundInt
[1/20/11 0:00:41:703 GMT+08:00] 00000022 ResourceAdapt 2
com.ibm.j2ca.sap.SAPResourceAdapter
endpointActivation(MessageEndpointFactory,ActivationSpec) Entering method.
[1/20/11 0:00:41:718 GMT+08:00] 00000022 ResourceAdapt 1
com.ibm.j2ca.sap.SAPResourceAdapter
endpointActivation(MessageEndpointFactory,ActivationSpec) Adding endpoint.
Factory = com.ibm.ejs.container.MessageEndpointFactoryImpl@fd8599b6,
ActivationSpec = com.ibm.j2ca.sap.SAPActivationSpecWithXid@0
[1/20/11 0:00:41:718 GMT+08:00] 00000022 ResourceAdapt 3
com.ibm.j2ca.sap.SAPResourceAdapter
endpointActivation(MessageEndpointFactory,ActivationSpec) ->>>>>>>> adding
endpoints()
[1/20/11 0:00:41:718 GMT+08:00] 00000022 ResourceAdapt 2
com.ibm.j2ca.sap.inbound.EndpointManager addEndpoint Entering method.
[1/20/11 0:00:42:390 GMT+08:00] 00000022 ResourceAdapt 1
com.ibm.j2ca.sap.inbound.SAPEventListenerManager installAleFunctions --
>aSpec.getNumberofListeners()=0
[1/20/11 0:00:42:390 GMT+08:00] 00000022 ResourceAdapt 1
com.ibm.j2ca.sap.inbound.SAPEventListenerManager installAleFunctions Warning
NumberOfListeners=0 , at least one listner is required for inbound , adding one
listner. -->NumberOfListeners=1
....
[1/20/11 0:00:46:687 GMT+08:00] 00000022 ResourceAdapt 2
com.ibm.j2ca.sap.inbound.SAPEventListenerManager startAleEventListeners
Entering method.
[1/20/11 0:00:46:859 GMT+08:00] 00000022 ResourceAdapt 2
com.ibm.j2ca.sap.inbound.SAPEventListenerManager installALEFunctions Entering
method.
[1/20/11 0:00:46:859 GMT+08:00] 00000022 ResourceAdapt 1
com.ibm.j2ca.sap.inbound.SAPEventListenerManager installAleFunctions Attempting
to install ALE functions
[1/20/11 0:00:48:062 GMT+08:00] 00000022 ResourceAdapt 1
com.ibm.j2ca.sap.inbound.SAPEventListenerManager installAleFunctions Function
module IDOC_INBOUND_ASYNCHRONOUS has been installed successfully.
[1/20/11 0:00:48:078 GMT+08:00] 00000022 ResourceAdapt 2
com.ibm.j2ca.sap.inbound.SAPEventListener SAPEventListener Entering method.
[1/20/11 0:00:48:093 GMT+08:00] 00000022 ResourceAdapt 3
com.ibm.j2ca.sap.ale.inbound.SAPAleEventListener SAPAleEventListener
*****onNotification*****
[1/20/11 0:00:48:109 GMT+08:00] 00000022 ResourceAdapt 2
com.ibm.j2ca.extension.utils.persistencystore.EventPersistence
EventPersistence:constructor Entering method.
[1/20/11 0:00:48:109 GMT+08:00] 00000022 ResourceAdapt 2
com.ibm.j2ca.extension.utils.persistencystore.EventPersistence
EventPersistence:constructor DataSourceJNDIName is null.
[1/20/11 0:00:48:125 GMT+08:00] 00000022 ResourceAdapt 2
com.ibm.j2ca.extension.utils.persistencystore.EventPersistenceMemoryImpl
EventPersistenceMemoryImpl:constructor Entering method.
```

```

[1/20/11 0:00:48:125 GMT+08:00] 00000022 ResourceAdapt 3
com.ibm.j2ca.extension.utils.persistencestore.EventPersistenceMemoryImpl
EventPersistenceMemoryImpl:constructor setLogUtils Successful
[1/20/11 0:00:48:140 GMT+08:00] 00000022 ResourceAdapt 2
com.ibm.j2ca.extension.utils.persistencestore.EventPersistenceMemoryImpl
createEventTableInMemory() Entering method.
[1/20/11 0:00:48:140 GMT+08:00] 00000022 ResourceAdapt <
com.ibm.j2ca.extension.utils.persistencestore.EventPersistenceMemoryImpl
createEventTableInMemory() Exiting method.
[1/20/11 0:00:48:140 GMT+08:00] 00000022 ResourceAdapt 3
com.ibm.j2ca.extension.utils.persistencestore.EventPersistenceMemoryImpl
EventPersistenceMemoryImpl:constructor EventMemoryTable: NULL
[1/20/11 0:00:48:140 GMT+08:00] 00000022 ResourceAdapt <
com.ibm.j2ca.extension.utils.persistencestore.EventPersistenceMemoryImpl
EventPersistenceMemoryImpl:constructor Exiting method.
[1/20/11 0:00:48:140 GMT+08:00] 00000022 ResourceAdapt <
com.ibm.j2ca.extension.utils.persistencestore.EventPersistence
EventPersistence:constructor Exiting method.
[1/20/11 0:00:48:140 GMT+08:00] 00000022 ResourceAdapt <
com.ibm.j2ca.sap.ale.inbound.SAPAleEventListener SAPAleEventListener Exiting
method.
[1/20/11 0:00:48:156 GMT+08:00] 00000022 ResourceAdapt <
com.ibm.j2ca.sap.inbound.SAPEventListenerManager startAleEventListeners Exiting
method.
[1/20/11 0:00:48:156 GMT+08:00] 00000022 ResourceAdapt <
com.ibm.j2ca.sap.inbound.EndpointManager addEndpoint Exiting method.
[1/20/11 0:00:48:156 GMT+08:00] 00000022 ResourceAdapt <
com.ibm.j2ca.sap.SAPResourceAdapter
endpointActivation(MessageEndpointFactory,ActivationSpec) Exiting method.
[1/20/11 0:00:48:156 GMT+08:00] 00000022 ActivationSpe I   J2CA0523I: The
Message Endpoint for ActivationSpec sapinbound
(com.ibm.j2ca.sap.SAPActivationSpecWithXid) and MDB Application
SAPInboundEJB#SAPInboundEJB.jar#SapInboundIntMDB is activated.
[1/20/11 0:00:48:281 GMT+08:00] 00000022 EJBContainerI I   WSVR0057I: EJB jar
started: SAPInboundEJB.jar
[1/20/11 0:00:48:312 GMT+08:00] 00000022 ApplicationMg A   WSVR0221I:
Application started: SAPInboundEJB#SAPInboundEJB.jar
The Trace in server2 on Node B,
[1/20/11 0:00:52:937 GMT+08:00] 00000033 ApplicationMg A   WSVR0200I: Starting
application: SAPInboundEJB#SAPInboundEJB.jar
[1/20/11 0:00:52:953 GMT+08:00] 00000033 ApplicationMg A   WSVR0204I:
Application: SAPInboundEJB#SAPInboundEJB.jar Application build level: Unknown
[1/20/11 0:00:54:609 GMT+08:00] 00000033 EJBContainerI I   WSVR0037I: Starting
EJB jar: SAPInboundEJB#SAPInboundEJB.jar
[1/20/11 0:00:55:765 GMT+08:00] 00000033 EJBContainerI I   CNTR0167I: The
server is binding the sapinboundpkg.SapInboundInt interface of the
SapInboundIntSB enterprise bean in the SAPInboundEJB.jar module of the
SAPInboundEJB#SAPInboundEJB.jar application. The binding location is:
ejblocal:SAPInboundEJB#SAPInboundEJB.jar/SapInboundIntSB#sapinboundpkg.SapIn
boundInt
[1/20/11 0:00:55:812 GMT+08:00] 00000033 EJBContainerI I   CNTR0167I: The
server is binding the sapinboundpkg.SapInboundInt interface of the
SapInboundIntSB enterprise bean in the SAPInboundEJB.jar module of the
SAPInboundEJB#SAPInboundEJB.jar application. The binding location is:
ejblocal:sapinboundpkg.SapInboundInt
[1/20/11 0:00:56:718 GMT+08:00] 00000033 EJBContainerI I   WSVR0057I: EJB jar
started: SAPInboundEJB#SAPInboundEJB.jar
[1/20/11 0:00:56:765 GMT+08:00] 00000033 ApplicationMg A   WSVR0221I:
Application started: SAPInboundEJB#SAPInboundEJB.jar
[1/20/11 0:00:56:796 GMT+08:00] 00000033 CompositionUn A   WSVR0191I:
Composition unit WebSphere:cuname=SAPInboundEJB#SAPInboundEJB in BLA
WebSphere:blaname=SAPInboundEJB#SAPInboundEJB started.

```

In the JCA adapter application starting process, only if `ResourceAdapter.endpointActivation()` is invoked, can the inbound functionality be enabled and the inbound polling activated. In the trace of Node A, `ResourceAdapter.endpointActivation()` is invoked, accordingly, the polling is started. In the trace of Node B, `ResourceAdapter.endpointActivation()` is not invoked, indicating that inbound functionality is not enabled and the inbound instance in Node B is standby.

From WebSphere Application Server HA mechanism, when the configuration in this scenario is set, the resource adapter is registered to HA manager and one HA group is generated to ensure the inbound singleton service. Navigate to **Servers->Core Groups->Core group** settings. A list of the core group in this cell is displayed. Then, click **DefaultCoreGroup**. The configuration page for DefaultCoreGroup is displayed. See Figure 12.

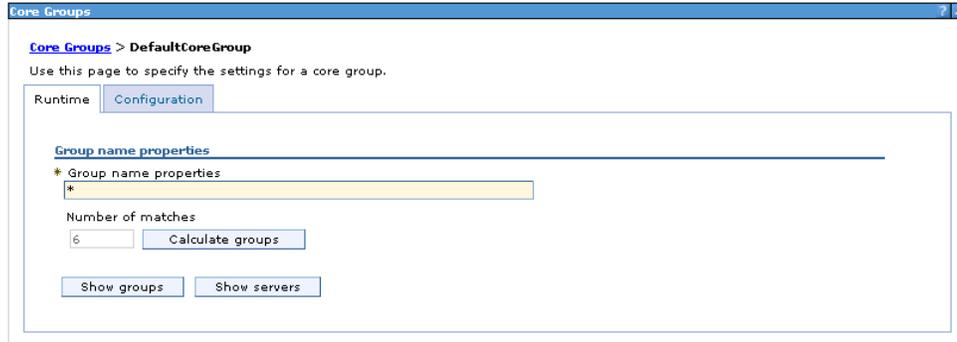


Figure 12.

Then switch to Runtime page, keep all of the values in this page as default and click **Show groups**. All of the HA groups in DefaultCoreGroup are listed in the table, one of which is the HA group generated for the cluster level adapter and the corresponding policy is One of N policy. See Figure 13.

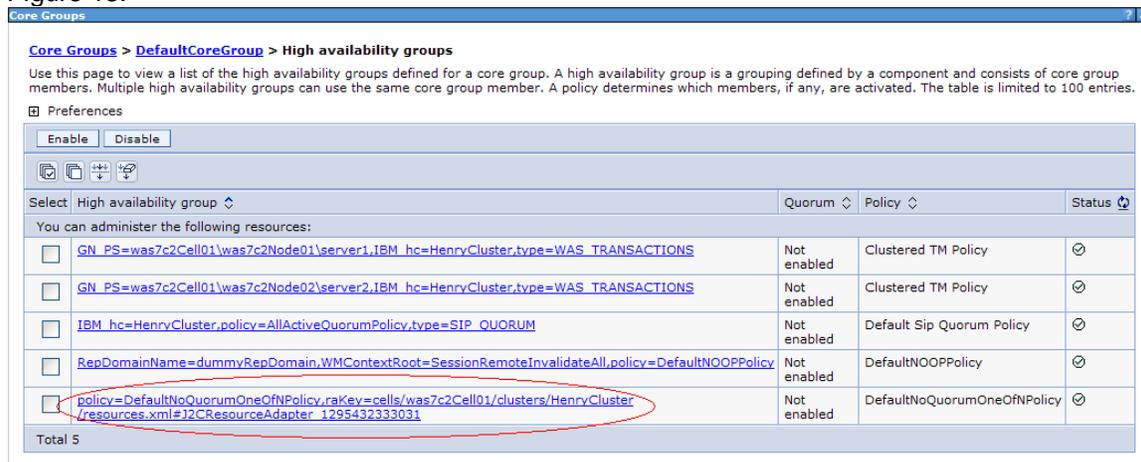


Figure 13.

Click this HA group list item to show the status. It indicates that the inbound service is active in the server1 and standby in server2. This is consistent with the actual runtime behavior and the node traces analysis.

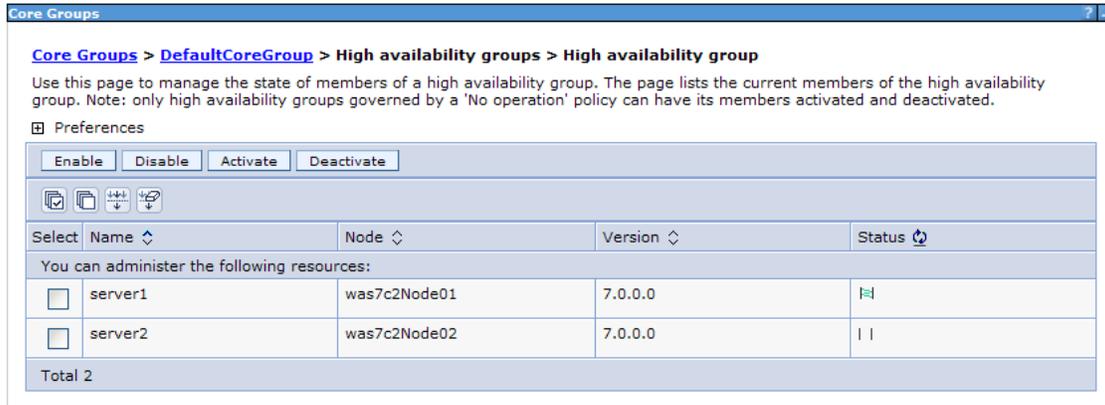


Figure 14.

On the other hand, if you change the configuration for the HA related properties, it will display the following runtime behavior.

You can uncheck the `isEnableHASupport` at the advanced property window for resource adapter as false and not change the other properties. When you redeploy and start the application, the adapter inbound instance on both nodes are active. If you check the `DefaultCoreGroup` status, you will find that the HA group mentioned above has been eliminated. Hence, the inbound singleton behavior has been disabled.

In another case, if you configure the `enableHASupport` in the custom properties window of the Resource Adapter as false and not change the other properties, the result is just the same as above case.

The different configuration and the corresponding runtime behavior for inbound adapter application are summarized in Table 2.

| Items                            | isEnabledHASupport at advanced property for cluster-level resource adapter | HACapability at advanced property for cluster-level resource adapter | EnableHASupport at customer property for cluster-level resource adapter | Runtime Behavior   |
|----------------------------------|--|--|---|--|
| Adapter Inbound at cluster level | true   | RA_INSTANCE_HA or RA_ENDPOINT_HA                                     | true  | <ol style="list-style-type: none"> <li>only one adapter inbound instance active in only one node.</li> <li>If active node fails, the inbound instance activate in the standby node.</li> </ol> |
| Adapter Inbound at cluster level | true   | RA_INSTANCE_HA or RA_ENDPOINT_HA                                     | false   | Adapter inbound instances are active on both nodes.  |
| Adapter Inbound at cluster level | false  | RA_NO_HA   | true  | Adapter inbound instances are active on both nodes.  |
| Adapter Inbound at cluster level | false  | RA_NO_HA   | false   | Adapter inbound instances are active on both nodes.  |

Table 2.

From the runtime behavior summary, the common rule is that the isEnabledHASupport at advanced properties and the enableHASupport at custom properties are like two switches for the adapter inbound HA behavior. If either of them are turned off it will lead to the HA singleton behavior being disabled.

When this scenario is for outbound adapter application, the outbound instance is activated on both nodes and starts working concurrently. When sending a large number of requests to the outbound adapter application, the outbound on two nodes will each take a half of the work load under the load balance. If one node fails, all the requests sent to that node are forwarded to the other node thereby processing all requests successfully. When the failed node is active again, the load balance is restored and the two outbound instance work concurrently again.

Regardless of the value that is set in the HA related properties, the runtime behavior for outbound application on HA environment are the same. The specific configuration and the according runtime behavior are summarized in Table 3.

| Items                             | isEnableHASupport at advanced property for cluster-level resource adapter | HACapability at advanced property for cluster-level resource adapter | EnableHASupport at customer property for cluster-level resource adapter | Runtime Behavior  |
|-----------------------------------|---|--|---|---|
| Adapter Outbound at cluster level | true  | RA_INSTANCE_HA or RA_ENDPOINT_HA                                     | true  | 1. Outbound is active on both nodes.<br>2. When one node fails, the requests to it are all forward to another node. |
| Adapter Outbound at cluster level | true  | RA_INSTANCE_HA or RA_ENDPOINT_HA                                     | false   | Same with above.  |
| Adapter Outbound at cluster level | false   | RA_NO_HA   | true  | Same with above.  |
| Adapter Outbound at cluster level | false   | RA_NO_HA   | false   | Same with above.  |

Table 3.

### Scenario Summary

In cluster-level deployment mode, only if the isEnableHASupport at advanced properties and the enableHASupport at custom properties are both true, the singleton HA behavior for inbound is enabled. For outbound, no matter how the cluster-level HA properties are configured, the feature of all active and load balance are always enabled. The cluster-level deployment is the recommended approach for adapter in WebSphere Application Server 7.0 HA environment.

### 2.2.1.2 Scenario 2 - HA configuration in Cluster-level different with it in node-level

In the configuration steps in Scenario 1, the node-level adapter's HA configuration keeps as default value. However, if the node-level HA configuration changes to other values, what is the runtime behavior of the inbound application and the outbound application? Does the HA configuration in node-level affect the runtime behavior remarkably? The answers are discussed in the following scenario.

#### Configuration Steps

1. Follow the [Step 1 to Step 3](#) in Scenario 1 to setup cluster-level adapter. Keep the cluster-level HA properties as default value.
2. Configure the HA property for node-level adapter in Node A to enable HA behavior. Navigate to the **Resource adapters > IBM WebSphere Adapter for SAP > Advanced resource adapter properties**, select the **Register this resource adapter with the high availability manager**, and keep the **Endpoint failover** radio as default. Configure the node-level adapter in Node B with the same step.
3. Follow the [Step 5 and Step 6](#) in Scenario 1 to deploy and start the adapter application.

#### Observations and Runtime Behavior

If the application is for inbound, the inbound instance will be active on both nodes, which indicates the inbound singleton HA behavior is disabled in this scenario. The reason is that the cluster-level HA property maintains the default value, so the Register this resource adapter with the high availability manager property is unchecked by default. In this scenario, the HA behavior is decided by cluster-level adapter's HA configuration.

If you specify any kind of configuration combination for the HA configuration in cluster-level and node-level, you will obtain the following summary as shown in the table below:

| Cluster-level HA support | Node-level HA Support | Runtime Behavior  |
|--------------------------|-----------------------|---|
| True                     | True                  | <ol style="list-style-type: none"> <li>only one adapter inbound instance active in one node.</li> <li>If active node fails, the inbound instance activate in the standby node.</li> </ol> |
| True                     | False                 | <ol style="list-style-type: none"> <li>only one adapter inbound instance active in one node.</li> <li>If active node fails, the inbound instance activate in the standby node.</li> </ol> |
| False                    | True                  | Adapter inbound instances are active on both nodes.   |
| False                    | False                 | Adapter inbound instances are active on both nodes.   |

Table 4.

In this table, the Cluster-level HA support is true means the mentioned two HA switch (isEnabledHASupport at advanced properties and the enableHASupport at custom properties) in cluster-level adapter are both true. If the Cluster-level HA support is false it implies that either of these two switches is false. The similar is for Node-level HA support, the only difference is that it corresponds to the two node-level HA switches. From this table, notice that the inbound HA behavior is consistent with the cluster-level HA configuration. Conversely, the node-level HA configuration does not affect the inbound HA behavior.

If the application is for outbound, no matter any kind of configuration combination for the HA configuration in cluster-level and node-level, the runtime behavior is just the same with it in Scenario 1.

### Scenario Summary

For adapter inbound, the cluster-level HA configuration overrides the Node-level HA configuration. This means that the inbound HA behavior is decided by the HA configuration for the cluster-level adapter, if it exists. For adapter outbound, both the cluster-level HA configuration and node-level HA configuration does not affect the HA behavior.

### 2.2.1.3 Scenario 3 - Deploy Adapter only in Node Level

In this scenario, adapter is deployed only in the node level and no cluster-level adapter is configured. Although this is not a recommended approach for the usage of adapter in HA environment, the different runtime behavior with the cluster-level deployment is illustrated here to understand why it's not recommended.

#### Configuration Steps

1. Install Adapter RAR in Node A and Node B following the [Step 1](#) in Scenario 1.
2. Configure the AS or MCF for Adapters in both nodes and set the same property values for the both.
3. Enable the HA properties for both nodes following the [Step 4](#) in Scenario 1.
4. Follow the [Step 5 and Step 6](#) in Scenario 1 to deploy and start the adapter application.

#### Observations and Runtime Behavior

If the application is for inbound, the inbound instance is active on both nodes. Although the HA related properties are all set to true in the node-level adapter, the inbound singleton HA runtime behavior is disabled. In fact, irrespective of the option specified for the HA configuration at the node-level, the runtime behavior is just the same.

Also by comparing the inbound application starting trace in Node A and Node B, the invocation on ResourceAdapter.endpointActivation() happened on both nodes, which is another explanation for this runtime behavior.

To understand the runtime behavior, we need to observe the HA group related settings. Navigate to **Servers->Core Groups->Core group**, click **DefaultCoreGroup**. Switch to **Runtime** page, keep all of the values in this page as default and click **Show groups**. Here there are two HA groups are generated for the individual node-level adapter in two nodes.

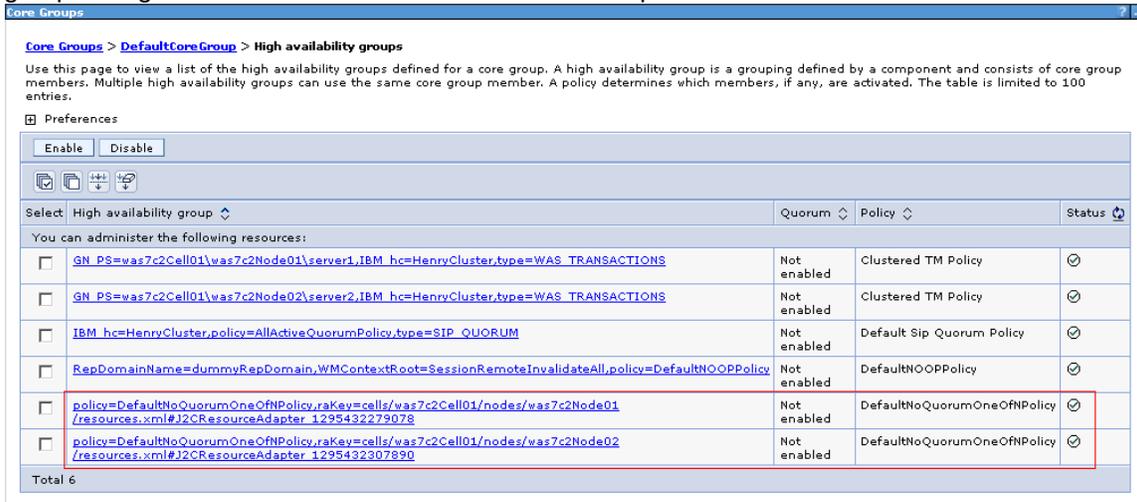


Figure 15.

Click either of the two HA group list item to enter the HA group page, it indicates that there is only one node in this HA group and the adapter inbound is active in it. By means of two HA group, the HA manager ensures each adapter inbound instance is always active in its node.

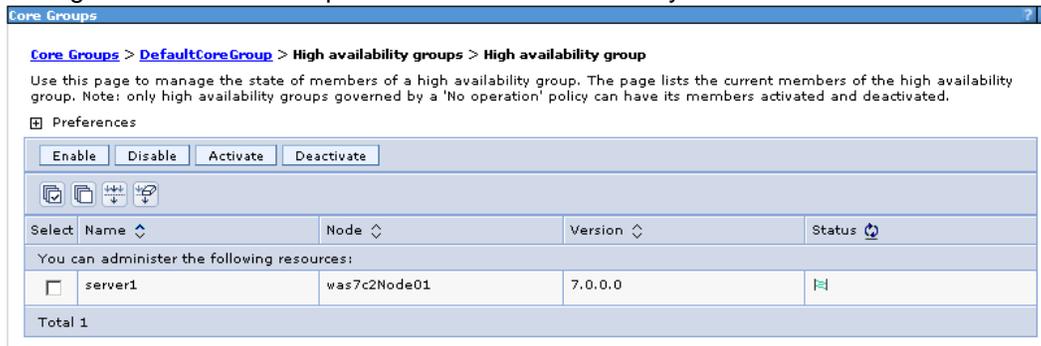


Figure 16.

If the application is for outbound, no matter any kind of configuration combination for the HA configuration in both node-level adapter, the runtime behavior is just the same with it in Scenario 1.

### Scenario Summary

Deploying adapter only in node level will disable HA singleton behavior for inbound. Hence, it is not recommended approach in HA environment. For outbound, the all active and load balance feature are the same with the cluster-level deployment. However, you need to maintain a consistency of MCF properties for both nodes in this case. And if there are more nodes in the cluster, the maintenance will become sick and tired job. The approach in this scenario is not recommended to customers and the cluster-level deployment described in Scenario 1 is the best practice and recommended way for adapter usage in WebSphere Application Server 7.0 HA environment.

## 2.2.2 Standalone Deployment in WebSphere Process Server 6.2 cluster environment

This section will describe the adapter HA configuration and the runtime behavior on WebSphere Process Server 6.2 HA environment through several scenarios. Also the limitations of WebSphere Process Server 6.2 HA environment for adapter deployment are described here. This will help clarify the incorrect approach and highlights these limitations in a real solution development through WebSphere Integration Developer 6.2 and targeted to WebSphere Process Server 6.2 HA environment.

### 2.2.2.1 Scenario 4 - Deploy Adapter in Cluster Level

In this scenario, the adapter will be deployed to the cluster level. The approach is similar with Scenario 1, however there are some differences in the HA related configuration.

#### Configuration Steps

1. Install adapter RAR on the two custom nodes in the svttop.AppTarget cluster. For each node, make the configuration following the [Step 1](#) in Scenario 1.
2. Generate a cluster-level adapter based on the node-level adapter, following the [Step 2](#) in Scenario 1. Specify the svttop.AppTarget cluster in the combo box for scope. Then, click **New** to generate the cluster-level adapter.

##### Resource adapters

Use this page to manage resource adapters, which provide the fundamental interface for connecting applications to an Enterprise Information System (EIS). The WebSphere(R) Relational Resource Adapter is embedded within the product to provide access to relational databases. To access another type of EIS, use this page to install a standalone resource adapter archive (RAR) file. You can configure multiple resource adapters for each installed RAR file.

☒ Scope: Cell=**aix115Cell01**, Cluster=**svttop.AppTarget**

Show scope selection drop-down list with the all scopes option

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, [see the scope settings help](#).

Cluster=svttop.AppTarget

##### Preferences

|   |   |                          |
|---|---|--------------------------|
| <input type="button" value="Install RAR"/> <input type="button" value="New"/> <input type="button" value="Delete"/> <input type="button" value="Update RAR"/> |   |                          |
|   |   |                          |
| Select  | Name  | Scope                    |
| You can administer the following resources:   |   |                          |
| <input type="checkbox"/>  | <a href="#">Platform Messaging Component SPI Resource Adapter</a> | Cluster=svttop.AppTarget |
| Total 1   |   |                          |

Figure 17.

After the generation of the cluster-level adapter complete, go into the cluster-level adapter configuration page. There is no Advanced properties link in this page as present in WebSphere Application Server 7.0. This difference is based on the fact that the WebSphere Process Server 6.2 is built on the WebSphere Application Server 6.1.0.23. Hence, in the WebSphere Application Server version not higher than 6.1.0.23 or on the WebSphere Process Server with version not higher than 6.2, there is no Advanced properties link in this page. The Advanced properties configuration is an enhancement from the higher version of WebSphere Application Server, specifically, WebSphere Application Server 7.0 release.

Here you cannot configure the isEnabledHASupport in the advanced properties of RAR as done in WebSphere Application Server 7.0 scenario. However, the enableHASupport in custom properties of RAR need to be configured to true.

3. Create cluster-level AS or MCF for the cluster-level adapter, following the configuration in [Step 3](#) in Scenario 1.

4. Develop adapter application with WebSphere Integration Developer 6.2.

In EMD, specify to use the standalone deployment by setting the **Deploy connector project** option as the “**On server for use by multiple applications**”. Specify **Connection properties** as the “**Use the predefined connection properties**”. Specify the **JNDI Lookup Name** to the AS or

MCF created in Step 3. After the application is developed and built, export it as EAR file. For the complete EMD process in WebSphere Integration Developer 6.2, please refer to [2].

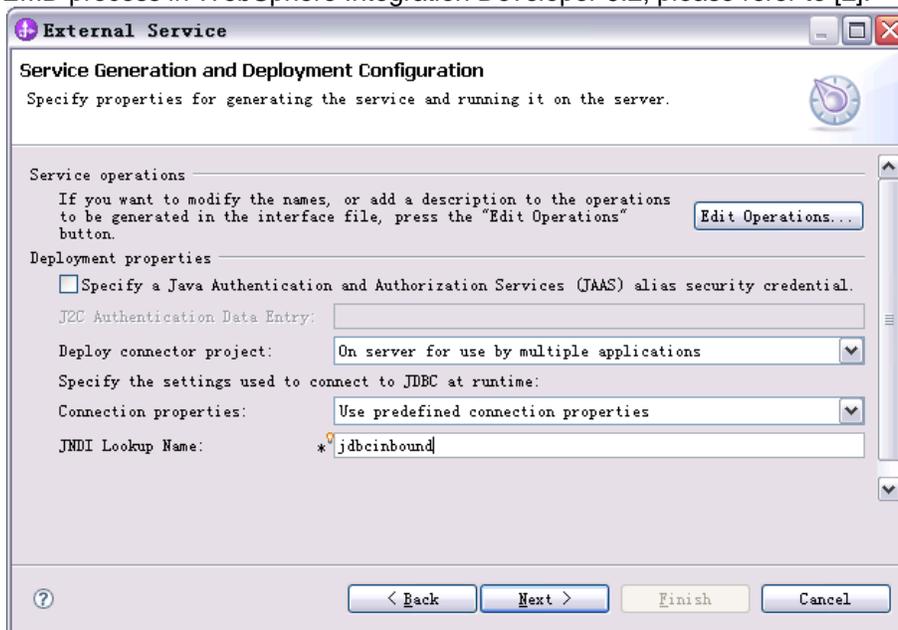


Figure 18.

5. Deploy the application and start, following the [Step 6](#) in Scenario 1. The only difference is to specify the correct target cluster for the application, as there are more than one cluster in this topology.

In the Step 2 of the deployment wizard, select the item “WebSphere:cell=aix116Cell01,cluster=svttop.AppTarget” in the **Clusters and servers** list box. And select all of the modules in the module list, click **Apply**. As a result all the modules are mapped to the svttop.AppTarget cluster. Then go on and finish this wizard to complete the deployment. Start the application.

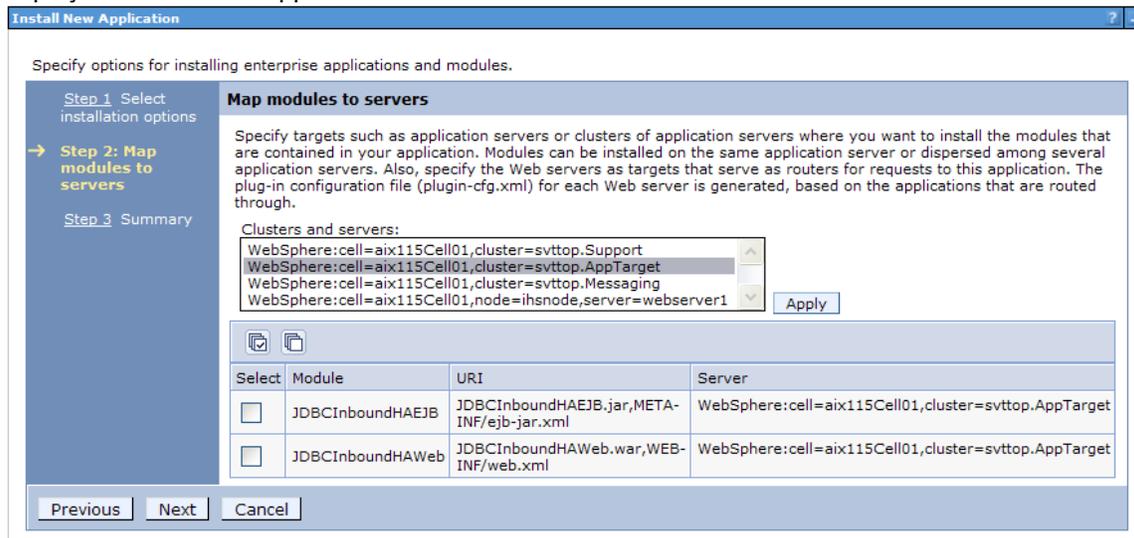


Figure 19.

### Observations and Runtime Behavior

If the application is for inbound in this scenario, the adapter inbound is active on only one node and the inbound instance on another node is standby. Comparing the trace of two nodes, the ResourceAdapter.endpointActivation() is invoked by only one node, where the inbound instance is activated. And for the node where the method is not invoked, the inbound instance is standby.

And also there is one HA group generated for the cluster-level adapter, one of N policy is specified for this HA group.

If the application is for outbound, it starts working on both nodes in the svctop.TargetApp cluster, and the feature of all active and load balance take effective.

There is the similar question as in the Scenario 2 for WebSphere Application Server 7.0 HA environment, what impact will be brought by specifying various configuration combination in the cluster-level adapter HA configuration and the node-level adapter HA configuration? All the cases and the corresponding runtime behavior are summarized in the table below.

| Cluster-level HA support | Node-level HA Support | Runtime Behavior                              |
|--------------------------|-----------------------|---|
| True                     | True                  | Singleton HA behavior for inbound is enabled. |
| True                     | False                 | Singleton HA behavior for inbound is enabled. |
| False                    | True                  | Inbound is active on both nodes.              |
| False                    | False                 | Inbound is active on both nodes.              |

Table 5.

In this table, the Cluster-level HA support being True stands for the enableHASupport property in cluster-level adapter is set to true and vice versa. The Node-level HA support being True stands for the enableHASupport property in node-level adapter is set to true and vice versa.

From this table, the inbound HA behavior is decided by the cluster-level HA support. And the node-level HA support has no impact on the inbound HA behavior. The enableHASupport property in cluster-level is the only HA switch comparing with the two switches in WebSphere Application Server 7.0 HA environment.

If the application is for outbound, no matter what kind of HA configuration make for cluster-level and node-level, the feature of all active and load balance are always enabled, just as the description in the Table 3 in Scenario 1.

### Scenario Summary

In WebSphere Process Server 6.2 HA environment, the cluster-level deployment mode enables the singleton HA behavior for adapter inbound application. The enableHASupport at custom properties in cluster-level adapter is the switch for enabling the singleton HA behavior for inbound. For outbound, irrespective of the kind of the cluster-level HA configuration, the feature of all active and load balance is always enabled. The cluster-level deployment is the recommended approach for adapter in WebSphere Process Server 6.2 HA environment.

### 2.2.2.2 Scenario 5 - Deploy Adapter in Node Level and generate AS or MCF manually

Similar with Scenario 3 for WebSphere Application Server 7.0, the adapter can also be deployed on only node-level in WebSphere Process Server 6.2 HA environment. Accordingly, this is also not a recommended approach for adapter usage in WebSphere Process Server 6.2 HA environment. However, the following discussion and the illustration of the actual configuration and runtime behavior states why this scenario is not recommended.

#### Configuration Steps

1. Install JDBC RAR in Node A, then configure the AS or MCF for the RAR.
2. Install JDBC RAR in Node B, then configure the AS or MCF with the same name and the property value of it in Node A.
3. In adapter EMD wizard, specify to use the standalone deployment through setting the **Deploy connector project** option as the “**On server for use by multiple applications**”. Specify **Connection properties** as the “**Use predefined connection properties**”. Specify the **JNDI Look Name** as the MCF or AS configured in Step 1. Complete the EMD and generate the application.
4. Deploy the application and start it.

## Observations and Runtime Behavior

If the scenario is for inbound adapter application, the adapter inbound start polling on both nodes and the singleton HA behavior is disabled.

If the scenario is outbound, it has the same behavior as described in the Scenario 4, the feature of all active and load balance is enabled.

## Scenario Summary

This approach will not achieve the correct runtime behavior for adapter inbound, and hence is not recommended in the actual usage. Also, for maintenance, it is not a recommended approach to maintain consistency of the multiple AS or MCF that are distributed on multiple nodes.

### 2.2.2.3 Scenario 6 - Deploy Adapter in Node Level and generate AS or MCF automatically

In this scenario, most of the configuration steps are similar with Scenario 5, except that the AS or MCF generation can be automated in deployment process, which is achieved by making some settings in EMD wizard in the WebSphere Integration Developer 6.2. As mentioned above, this scenario is not recommended because of the some known limitations in WebSphere Process Server 6.2 HA environment.

## Configuration Steps

1. Install JDBC RAR in Node A, then configure the AS or MCF for the RAR.
2. Install JDBC RAR in Node B, then configure the AS or MCF with the same name and the property value of it in Node A.
3. In EMD, specify to use the standalone deployment through setting the Deploy connector project option as the “**On server for use by multiple applications**”. Specify **Connection properties** as the “**Specify connection properties**”. Complete the connection information in this page. Complete the EMD and generate the application.

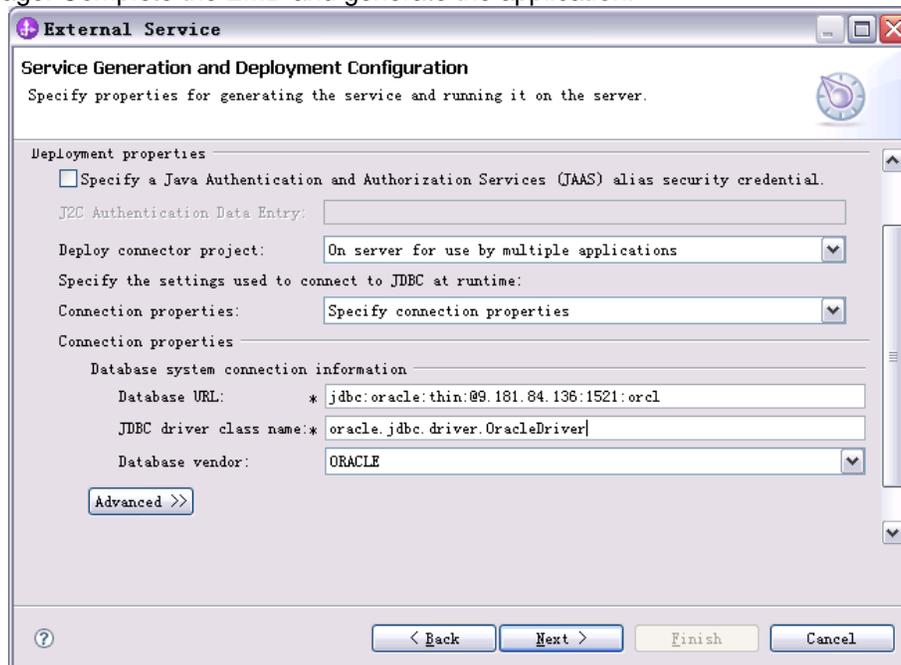


Figure 20.

4. Deploy the application and start it.

## Observations and Runtime Behavior

After Step 4, the AS or MCF is automatically generated in only one node with the property values specified in EMD. There is no AS or MCF is generated in another node. This is a known

functionality limitation in WebSphere Process Server 6.2 HA environment, although this approach can work well in standalone WebSphere Process Server 6.2 environment. As a result, it will lead to some abnormal runtime behavior accordingly.

For inbound, after Step 4, adapter inbound can only start in the node with the AS generated. And it fails starting in the node with no AS generated since the AS JNDI cannot be found. The below error happens,

```
J2CA0052E: The lookup of the ActivationSpec with JNDI Name
JDBCInboundHAAuto/JDBCInboundInterface1_AS failed due to exception
javax.naming.NameNotFoundException
```

For outbound, after Step 4, adapter outbound started on both nodes successfully. However, the “JNDI Not Found exception” happens if invoke requests are distributed to the adapter outbound instance on the node where the MCF is not generated.

Additionally, if another step is added to configure a cluster-level adapter in this scenario before deployment and when you complete the operations in this scenario, the result is the same. The AS or MCF is generated on only one node-level adapter, and no AS or MCF is generated in the cluster level adapter, although, generating AS or MCF in the cluster-level adapter automatically is the expected behavior in this scenario.

Based on the above scenario, when you uninstall the application, sometimes the automatically generated MCF or AS cannot be removed automatically, which is another known limitation in WebSphere Process Server 6.2 HA environment. In this case, you need to remove the AS or MCF manually to ensure that everything generated in the deployment process has been clean up.

### **Scenario Summary**

The AS or MCF automatic generation function is not as the expected to generate AS or MCF to the cluster-level adapter. And even for the node-level adapter for which this function is originally designed, it has some limitations when it is used in HA environment. This approach is not encouraged to be used in the WebSphere Process Server 6.2 HA environment and it is more applicable to be used in WebSphere Process Server 6.2 standalone environment.

### **2.2.2.4 Scenario 7 - Both Node-level JNDI and Cluster-level JNDI Exists**

The AS and MCF can be generated on either node-level adapter or cluster level adapter, naturally coming the question is that if both the node level AS or MCF and the cluster level AS or MCF is coexisted and some properties are different, for instance, the polling properties are different, which level of AS or MCF will take effect? This question is to be answered in this scenario.

### **Configuration Steps**

1. Install Adapter RAR in Node A, then create and configure the AS named “InboundAS”.
2. Install Adapter RAR in Node B, then create and configure the AS named “InboundAS” which has the same properties with the one on Node A.
3. Specify the enableHASupport in the custom properties of both the two RAR to true.
4. Develop Inbound Adapter application to use Stand-alone deployment and set JNDI name as “InboundAS”.
5. Deploy the application and start it.
6. Configure the Cluster-level adapter for this Adapter RAR in svttop.AppTarget cluster then configure the different property value with the node-level AS properties, specify the enableHASupport in the custom properties of RAR to true.
7. Restart the application.
8. Remove the Cluster-level AS and restart application.
9. Restart application server on Node A.

### **Observations and Runtime Behavior**

The following is the observations in the above scenarios,  
After Step 5, the inbound instance starts polling on both nodes.

After Step 7, the inbound instance restarts with the cluster-level AS properties on Node A, and the inbound instance in another node stops.

After Step 8, the inbound instance on Node B also starts polling. At this time, the inbound instance on Node A is polling with the Cluster-level AS properties, since the deletion of cluster level AS has not take effect. The inbound instance on Node B is polling with the local node level AS properties since it's started with the node-level AS.

After Step 9, the inbound adapter application restarts polling on Node A with the local node level AS properties.

### **Scenario Summary**

The cluster-level AS or MCF has the higher priority to take effect than the node level AS or MCF with the same JNDI name. The property value in cluster level AS or MCF will override the property value in node level AS or MCF. The node level AS or MCF can be effective only when there is no cluster level counterpart exists. Another important fact is that the deletion of AS or MCF will not take effect at once. You need to restart the server for this change to take effect. Also in this scenario it proves again that only when there is a cluster-level adapter can it achieve the inbound singleton HA behavior.

Generally speaking, configuring AS or MCF on both cluster level and node level is not recommended to customer. The best practice is to create AS or MCF on the cluster level only, just as described in [Scenario 4](#).

## **2.2.3 Standalone Deployment in WebSphere Process Server 7.0 cluster environment**

WebSphere Process Server 7.0 is built on the WebSphere Application Server 7.0.0.7, which is the derived version from WebSphere Application Server 7.0. So the adapter's HA configuration is similar with the WebSphere Application Server 7.0 HA environment. Also, from the perspective of WebSphere Integration Developer tooling and the server runtime behavior, the WebSphere Process Server 7.0 HA environment inherits many features from the WebSphere Process Server 6.2 HA environment. Scenario 5, Scenario 6 and Scenario 7 are also applicable for WebSphere Process Server 7.0 HA environment. The configuration approach, the runtime behavior and the mentioned limitations in those scenarios are also real for WebSphere Process Server 7.0 HA environment. Here the difference for adapter HA configuration and runtime behavior in WebSphere Process Server 7.0 HA environment in comparison with it in the other two environments is highlighted through the following scenario.

### **2.2.3.1 Scenario 8 - Deploy Adapter in Cluster Level**

Similar with it in WebSphere Application Server 7.0 environment, the most suggested way is to generate a cluster-level adapter RAR based on the node-level adapter RAR, then configure AS/MCF in the cluster-level adapter and deploy adapter application referring to the AS/MCF JNDI.

#### **Configuration Steps**

1. Follow the [Step 1 to Step 3 in Scenario 4](#) to install Adapter RAR in Node A and Node B, generate the cluster-level adapter and create the cluster-level AS or MCF.
2. Configure the HA property for cluster-level adapter. Keep the enableHASupport property in Custom properties as true. Enter the Advanced resource adapter properties, the Register this resource adapter with the high availability manager is by default unselected. Keep it no change.
3. Follow the [Step 4 and Step 5 in Scenario 4](#) to complete the adapter EMD, generate the adapter application and start it. The only difference for EMD is that here use the WebSphere Integration Developer 7.0. The specific steps refer to [3].

#### **Observations and Runtime Behavior**

If the scenario is for inbound, it is active in only one node and is standby in another node. So although the Register this resource adapter with the high availability manager checkbox is unselected, the singleton HA behavior for inbound is enabled. This is a prominent difference with Scenario 1. The Register this resource adapter with the high availability manager checkbox function was revoked in latest WebSphere Application Server version beginning from WebSphere Application Server 7.0.0.7. The “Endpoint failover” and “Resource adapter instance failover” radio buttons are still visible in the Admin Console. Regardless of the configuration settings for these options, the implementation in the server runtime assigns the fixed value for these options to enable the singleton HA behavior by default.

Accordingly, the enableHASupport property at custom properties in cluster-level adapter becomes the only HA switch to enable or disable HA behavior in WebSphere Process Server 7.0 HA environment. Note that from the WebSphere Application Server 7.0.0.7 and WebSphere Process Server 7.0 version, the resource adapter’s HA configuration mechanism restores to the status in WebSphere Application Server 6.1.0.23 and WebSphere Process Server 6.2.

For outbound, the runtime behavior is the same with it in Scenario 4, the feature of all active and load balance is enabled in this scenario.

### Scenario Summary

The cluster-level adapter deployment is the most recommended approach for the usage in the WebSphere Process Server 7.0 HA environment. The advanced properties for resource adapter HA behavior are revoked in WebSphere Process Server 7.0 and the enableHASupport property at custom properties in cluster level adapter decides the final HA behavior.

## 2.3 Embedded Deployment

For Embedded Deployment, the adapter RAR is embedded in the adapter application EAR and is installed with the EAR. In comparison with the Standalone Deployment, this approach brings the convenience to deploy application easily without the need to deploy RAR separately on the node and cluster. However, in this mode, the adapter RAR is bound to the specific application and cannot be shared by multiple applications, thus brings more resource consumption in runtime. Since WebSphere Application Server 7.0 does not support the embedded deployment, this section mainly focuses on the embedded deployment in WebSphere Process Server 6.2 HA environment and WebSphere Process Server 7.0 HA environments.

### 2.3.1 Embedded Deployment in WebSphere Process Server 6.2 HA environment

#### 2.3.1.1 Scenario 9 - Embedded deployment in WebSphere Process Server 6.2 HA environment

To deploy adapter with embedded deployment, specify the option in WebSphere Integration Developer 6.2 EMD wizard and generate the EAR with the adapter embedded in it. And then deploy it in the admin console.

#### Configuration Steps

1. Start the adapter EMD wizard in WebSphere Integration Developer 6.2. Specify to use the embedded deployment by setting the **Deploy connector project option** as the **“With module for use by single application”** in the page of Service Generation and Deployment Configuration. Complete the EMD wizard to generate the application artifacts.
2. Export the application artifacts as EAR file and follow the [Step 5 in Scenario 4](#) to deploy the application to the svttop.AppTarget cluster.
3. Open the Admin Console, navigate to **Enterprise Applications > JDBCInboundHAEmbeddedApp > Manage Modules > CWYBC\_JDBC.rar > JDBCInboundHAEmbeddedApp.IBM WebSphere Adapter for JDBC > Custom properties**, here the enableHASupport is true by default. Keep it no change.



### Configuration Steps

1. Start the EMD in WebSphere Integration Developer 7.0, follow the [Step 1 to Step 2 in Scenario 9](#) to deploy the adapter application with embedded deployment.
2. In Admin Console, navigate to Enterprise **Applications > JDBCInboundHAEmbeddedApp > Manage Modules > CWYBC\_JDBC.rar > JDBCInboundHAEmbeddedApp.IBM WebSphere Adapter for JDBC**. Enter the **Custom properties**, here the enableHASupport is set to true by default. Then enter the **Advanced properties**, the “Register the resource adapter with the high availability manager” is by default is unselected. Keep it no change.
3. Start the application.

### Observations and Runtime Behavior

If the scenario is for inbound, it is active in only one node and is standby in another node. Just as mentioned in Scenario 8, the Register this resource adapter with the high availability manager checkbox function is revoked in WebSphere Process Server 7.0. It enables the singleton HA behavior for adapter inbound service in runtime by default. Irrespective of what values set to these advanced properties, the runtime behavior is the same.

In this scenario, there is also a HA group generated automatically for the embedded adapter to ensure the singleton HA behavior.

If set the enableHASupport at custom properties to false, then restart the application, the inbound instance becomes active in both nodes. The singleton HA behavior is disabled. This behavior is consistent with what was discussed in Scenario 8. The enableHASupport property for adapter is the only one HA switch for enable or disable the singleton HA behavior.

If the scenario is for outbound, the feature of all active and load balance take effect on both nodes, no matter what value set to the HA properties of the embedded RAR.

### Scenario Summary

In WebSphere Process Server 7.0 HA environment , embedded deployment by default enables the singleton HA behavior for inbound. The enableHASupport property value of adapter RAR is the only switch to enable or disable the singleton HA behavior for adapter inbound. This approach is also a typical usage recommended in the WebSphere Process Server 7.0 HA environment.

## 3. Achieve High Availability of Adapter Transaction in HA Environment

Standalone WebSphere Process Server or WebSphere Application Server environment provides support for the adapter transaction configuration and management. Transaction manager in WebSphere Application Server runtime provides the coordination for global transactions on multiple distributed XA resource in a business process. Transaction logs provide the transaction recovery in the case of server crash in global transaction. However, in comparison with the standalone environment, there's some difference in leveraging the transaction service provided by WebSphere Process Server or WebSphere Application Server container in HA environment to achieve the high availability for adapter application's transaction. This section will mainly focus on the typical HA configuration for transactions and illustrate the transaction failover and recovery runtime behavior.

### 3.1 HA requirement for Global Transaction

In the business scenarios, adapter with global transaction support needs to join the global transaction to operate multiple distributed resource to ensure that the update of data in a global view with atomicity. The typical scenario is displayed in Figure 22. In the WebSphere HA environment, Adapter outbound A and outbound B are involved in a business process. Additionally, they both join a global transaction. Adapter Outbound A updates the data of EIS A, Adapter outbound B update the data of EIS B. Both EIS A and EIS B implement the XA resource and provide the support for the two phase transaction commit protocol. So when Adapter

outbound A and Adapter outbound B both succeed, the whole global transaction commits, else if either of the two outbound operations fails, the global transaction will rollback as a whole. Also if the application crash or server crash happens in the process of global transaction, it can go on commit or rollback after the application or server environment restored, this is due to the usage of transaction logs, which records the transaction context information for commit or rollback. The transaction logs are by default configured in the specific directory in server installation path, app\_server\_root/ tranlog/cell\_ name/node\_ name/server\_ name.

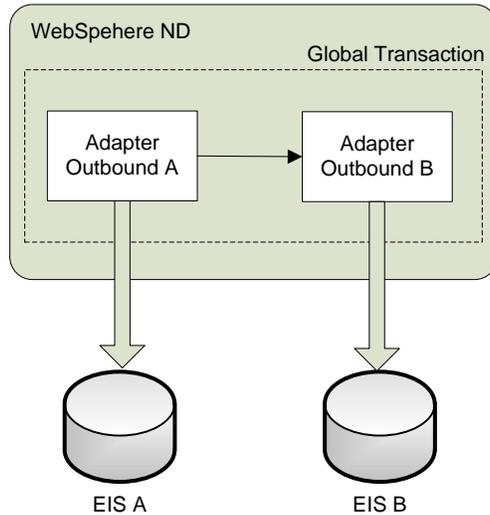


Figure 22.

In WebSphere HA environment, the servers in different nodes have the independent transaction log directory located in their own installation path. When one node fails, the node failover happens and all the work will switch to the peer node, accordingly, the in-fly transaction in the process of commit or rollback will resume in the peer node. To achieve this, the peer node need the access to the transaction logs persisted by the predecessor node. This is implemented by transaction log sharing in a shared disk.

### 3.2 Transaction Scenario in HA environment

To satisfy the HA requirement for the global transaction, setup the scenario based on the WebSphere Process Server 6.2 HA environment. It is displayed in Figure 23.

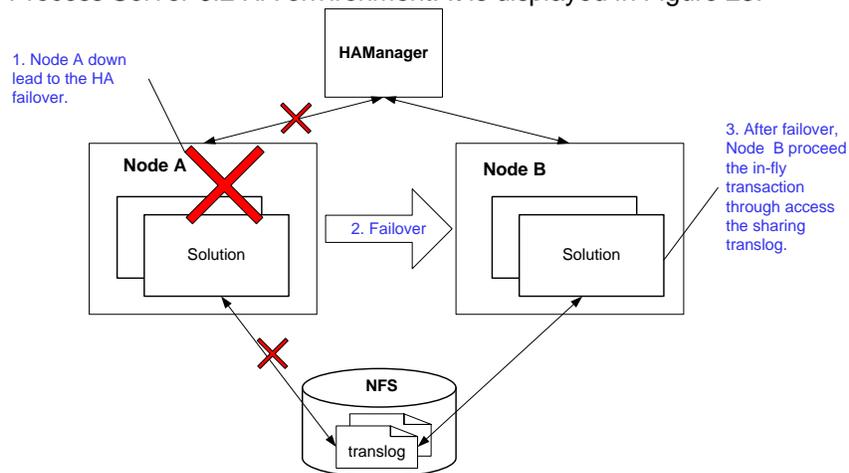


Figure 23.

In this scenario, the business application is deployed to the WebSphere HA environment. The Node A (preferred node) and Node B (backup node) work together with the HAManager. A disk

sharing is configured for the disk access for both two nodes. Configure the transaction log directory of both two nodes to the same location in the shared disk. The application is started in the preferred Node A at the beginning. And when it went down in the process of XA transaction, the node failover happens to switch the running business application and its transaction process to Node B. Accordingly, Node B will take over the in-fly transaction and go on processing it, through the way of accessing the transaction context and status information in the shared transaction logs. The node failover may happens in any point of the global transaction process, that is, it may happens in the 1<sup>st</sup> phase commit operation, may happens in the second phase commit, may happens in the rollback operation and so on. No matter which point the node failover happens, the context information of the whole transaction are recorded in the transaction logs and shared between the two nodes, so both nodes have the enough information to go on processing the in-fly transactions in any time when the application is activated in it. The peer recovering ability in this scenario has been provided by WebSphere HA platform, using the network-attached storage device (NAS) as the medium for placing the transaction logs. All the nodes have shared access to the transaction logs, and also the NAS ensures only a single client process can access the log at a time to maintain the integrity of a recovery log file.

### Configuration Steps

1. Configure the NFS v4 as the medium for the shared transaction logs. And mount local directory of every node in the svctop.AppTarget cluster to specific directory in the NFS v4 server. For example, /home/SVT/TransactionLog in Node A mount to the /home/Share112/tranlog directory in the NFS disk. Configure to ensure that all nodes have read and write access for the mounted folder.
2. Enable the One of N HA policy for transaction service in ND cluster. In the Admin Console, navigate to the **Core Groups > DefaultCoreGroups > Policies**, ensure the Clustered TM Policy is associated with the transaction service by the type = WAS\_TRANSACTIONS match criteria. This policy is a One of N type policy which provides automated peer recovery processing support such that only one server owns the transaction log at any time.

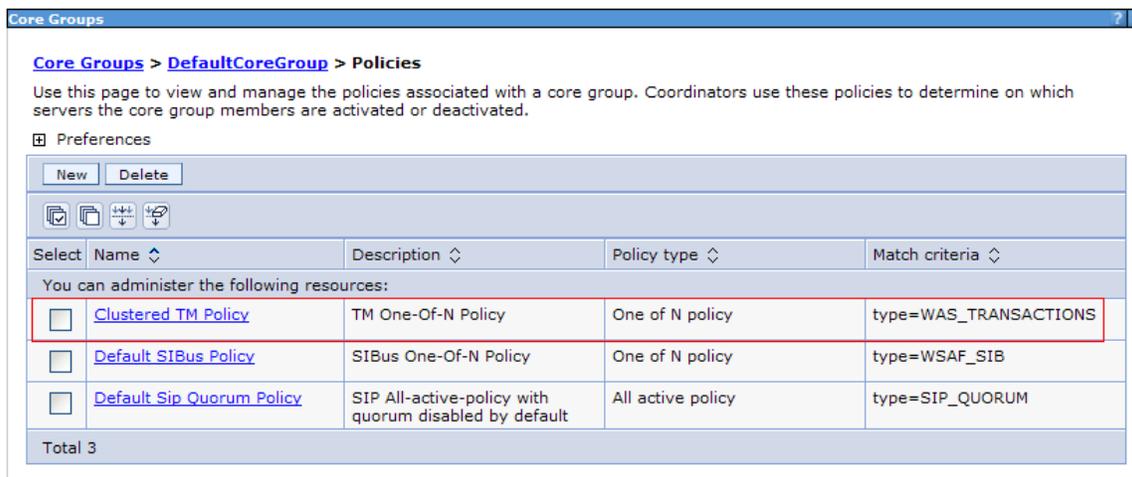


Figure 24.

3. Enable high availability for persistent services, namely the transaction service. Navigate to the **WebSphere application server clusters > XXXServer**, switch to the Configuration tab page, check the **Enable failover of transaction log recovery** checkbox and restart the servers for the configuration changes to take effect.

[WebSphere application server clusters](#) > [svttop.AppTarget](#)

Use this page to change the configuration settings for a cluster. A server cluster consists of a group of application servers. If one of the application servers that is a member of the cluster fails, requests are routed to other members of the cluster.

Runtime Configuration Local Topology

**General Properties**

\* Cluster name  
svttop.AppTarget

Bounding node group name  
DefaultNodeGroup

Prefer local

Enable failover of transaction log recovery

Apply OK Reset Cancel

**Cluster messaging**

- Messaging engines

**Business Integration**

- Business Integration Configuration
- Business Space Configuration
- REST services
- Service Component Architecture
- Common Event Infrastructure
- Business Process Choreographer
- Business Rules

Figure 25.

4. Provide the transaction log location for all the nodes in this cluster. For every server in the svttop.AppTarget cluster, navigate to the **Application Servers>XXX Server**, choose the **Container Settings>Container Services>Transaction Service**. Specify the local mount directory path in the **Transaction log directory** edit box. Check the **Enable file locking** checkbox.

[Application servers](#) > [svttop.AppTarget.aix115Node01.0](#) > [Transaction service](#)

Use this page to specify settings for the transaction service. The transaction service is a server updates to multiple resource managers to ensure atomic updates of data. Transactions are : container in which the applications are deployed.

Runtime Configuration

**General Properties**

Transaction log directory  
/home/SVT/TransactionLog

\* Total transaction lifetime timeout  
120 seconds

\* Async response timeout  
30 seconds

\* Client inactivity timeout  
60 seconds

\* Maximum transaction timeout  
300 seconds

Heuristic retry limit  
0 retries

Heuristic retry wait  
0 seconds

Enable logging for heuristic reporting

Heuristic completion direction  
ROLLBACK

Accept heuristic hazard

Enable file locking

Enable transaction coordination authorization

Figure 26.

5. Develop and deploy adapter application with cluster-level deployment as described in [Scenario 4](#). Configure to join the JDBC adapter outbound applications to the global transaction.
6. Start the application and send a large number of events for processing.

7. Shut down the Node A while processing and shut down Node B after a while.
8. Continue sending events until all of the events have been processed.

### **Observations and Runtime Behavior**

The business process can process all of the events with high availability and maintain every event process as an atomic transaction. The transaction recovery logs in both nodes are all generated in the shared transaction log directory. Whenever and whichever node shutdown, the in-fly transaction resume processing in the peer node until it completes.

### **Scenario Summary**

To achieve the high availability for global transaction of adapter application in the HA environment, it is necessary that the infrastructure for transaction recovery log sharing is setup and make it available for all nodes to access.

## **4. Summary**

This white paper describes two deployment approaches and the runtime behavior for WebSphere Adapter in WebSphere Process Server or WebSphere Application Server HA environment through several scenarios. Also, it highlights the best practices in these deployment approaches and clarifies the known limitations in WebSphere Process Server HA environment. At the end of this paper the Section 3 presents a description of how to configure the transaction recovery log sharing in WebSphere Process Server or WebSphere Application Server HA environment to enable the high availability of global transaction for adapter applications.

## **5. Glossary**

**Failover.** A standby or backup system is used to take over for the primary system if the primary system fails. In principle, any kind of service can become highly available by employing system failover techniques.

**load balance.** Load balance is a technique to distribute workload evenly across two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload.

**RAR.** A Resource Adapter Archive (RAR) file is the valid format for deployment of resource adapters on application servers.

**EAR.** An Enterprise Archive, or EAR, is a file format used by Java EE for packaging one or more modules into a single archive so that the deployment of the various modules onto an application server happens simultaneously and coherently.

**AS.** An Activation Specification (AS) is a JavaBean used during endpoint activation. Endpoint activation is the process of notifying resource adapters of eligible message listeners. The activation specification is a class implementing `javax.resource.spi.endpoint.ActivationSpec`.

**MCF.** A Managed Connection Factory (MCF) is a JavaBean which represents adapter outbound connectivity information to an EIS instance from an application via a specific resource adapter instance. This contains the configuration information pertaining to outbound connectivity to an EIS instance.

**EMD.** Enterprise Metadata Discovery (EMD) is a standard which defines a common API that adapters can use to expose their services and business objects to tools for the generation of JCA based applications. It is used by Enterprise Service Discovery wizard in WebSphere Integration Developer or J2C tooling in Rational Application Developer to create EIS metadata related artifacts.

**global transaction.** Opposite to the local transaction, refer to the transaction involves accessing multiple resource managers and preserve ACID properties.

**XA.** The XA standard is a specification by The Open Group for distributed transaction processing (DTP). It describes the interface between the global transaction manager and the local resource manager. The goal of XA is to allow multiple resources (such as databases, application servers,

message queues, etc.) to be accessed within the same transaction, thereby preserving the ACID properties across applications. XA uses a two-phase commit to ensure that all resources either commit, or rollback, any particular transaction simultaneously.

**network-attached storage.** Network-attached storage (NAS) is file-level computer data storage connected to a computer network providing data access to heterogeneous network clients. NAS systems contain one or more hard disks, often arranged into logical, redundant storage containers to provide both storage and a file system.

## 6. Resources

- [1] [WebSphere Adapter in Rational Application Developer Information Center](#)
- [2] [WebSphere Adapter 6.2 Information Center](#)
- [3] [WebSphere Adapter 7.0 Information Center](#)
- [4] [WebSphere Application Server Network Deployment V6:High Availability Solutions](#)
- [5] [IBM WebSphere Developer Technical Journal: Transactional high availability and deployment considerations in WebSphere Application Server V6](#)
- [6] [WebSphere Application Server 7.0 - Network Deployment](#)