



IBM Software Group

# IBM® WebSphere® Application Server V6

## *High Availability Details*



@business on demand.

© 2005 IBM Corporation  
Updated April 29, 2005

This presentation will focus on the details of High Availability services, including Core Groups and High Availability Policies.

## Goals

- Details of high availability services
  - ▶ Core Groups
  - ▶ High Availability (HA) policy options
  - ▶ Failure detection
  
- What is not covered
  - ▶ Custom HA policies, multiple Core Groups and Core Group bridges




The goals of this presentation are to introduce you to the administrative concepts related to high availability, including Core Groups and High Availability Policies. The presentation will also teach you how to configure WebSphere Application Server to meet your high availability needs. This presentation does not cover advanced topics like custom High Availability Policies, configuring multiple Core Groups, or Core Group bridges.

## Section

# ***Core Groups***

The next section will cover Core Groups.

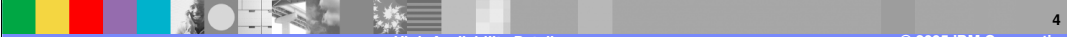
IBM Software Group 

## Core Groups

- A Core Group defines the set of WebSphere Application Server processes that participate in providing High Availability function to each other
- Processes can be Deployment Managers, Node Agents, Application Servers, Cluster Members
  - ▶ A process is a member of exactly one Core Group
  - ▶ All members of a cluster must be within the same Core Group
- Workload Management (WLM) information is shared automatically between Core Group members
- Singleton services running in a Core Group can failover only to another member of the same Core Group that is capable of running the service

Servers

- Application servers
- Generic servers
- Core groups
- JMS Servers
- Web Servers
- Clusters
- Core group bridge settings
- Cluster topology



High Availability Details 4  
© 2005 IBM Corporation

A Core Group is a logical boundary that defines the set of processes that can provide each other with High Availability functionality. Node Agents, Cluster Members (or stand-alone Application Servers), and the Deployment Manager can be members of a Core Group. A process can be a member of one and only one Core Group. Additionally, all members of a Cluster must be members of the same Core Group.

Members of a Core Group share information among one another, so that they are aware of the status of other Core Group members. Singleton services, such as WLM routing, running on one member of a Core Group can be failed over to any other member of the Core Group.

## Default Core Group

- Deployment Manager installation creates a default Core Group
  - ▶ Called “DefaultCoreGroup”
  - ▶ Has default HA policies for Transaction Manager and messaging
- As managed processes are added to the cell, they are automatically added to “DefaultCoreGroup”
- In most cases, the default setting will suffice



When the Deployment Manager is installed, a default Core Group, named “DefaultCoreGroup”, is created. It has High Availability policies for the Transaction Manager and Messaging Engines predefined. Whenever a managed process is added to the cell, it is automatically added to DefaultCoreGroup.

The majority of users will not need to worry about changing Core Group settings. This means that enabling high availability for singleton services is incredibly simple—you just have to install the product. Core Group settings can be modified, however, to create more advanced configurations. Creating multiple Core Groups is only useful in specific situations, such as a geographically distributed cell, and is not covered in this presentation.

## HA Coordinators

- Each Core Group has an HA Coordinator that coordinates all HA activities among the Core Group processes
  - ▶ Collects all the HA information – what services are running in which processes
  - ▶ If a service in one process fails, then the services are restarted in another process (based on HA policies)
- Any process in the Core Group can be the HA Coordinator
  - ▶ Selected using an internal algorithm
  - ▶ Election is held again if the original Coordinator fails
- HA Coordinator work can be partitioned across multiple processes
  - ▶ The coordination work is shared so one process does not get overloaded
  - ▶ Useful when you have many clusters within the cell
- Can assign preferred servers to be the HA Coordinators
  - ▶ WebSphere Application Server will try to use the preferred servers
  - ▶ Will use other servers only if the preferred servers are down



Each Core Group has a service called the High Availability Coordinator that manages the High Availability related work for the Core Group. It keeps track of what services are running in which processes and, in the case of a failure, decides which process should restart that service. The processes hold an election among themselves to determine which process will be the HA Coordinator, and if that process fails, the election will be held again to decide which of the remaining processes will become the Coordinator. In cases where you have several Clusters within your Cell, it can be useful to configure more than one HA Coordinator in your cell, so that a single process does not get overloaded. Core Group settings also allow you to define servers where you prefer that the HA Coordinator be run. If possible, the HA Coordinator will be run on a preferred server, but if no preferred servers are available, one of the remaining processes will be chosen as the coordinator.

IBM Software Group

# Coordinator Configuration

**General Properties**

\* Name  
DefaultCoreGroup

Description  
Default Core Group. The default core group cannot be deleted.

\* Number of coordinators  
1

**Additional Properties**

- Core group servers
- Custom properties
- Policies
- Preferred coordinator servers

**Related Items**

- Core group bridge

**Preferred coordinator servers**

**Core group servers**

RHG1/server1  
RHG1/ClusterMember 1  
RHG1/ClusterMember 2  
RHG1/server 2

**Preferred coordinator servers**

RHG1/nodeagent

Preferred coordinator servers order

Move up ^  
Move down v

Add >>  
Remove <<

Specify number of HA Coordinators - If more than 1 specified, then the coordinators share the load

List of Core Group servers that can be made preferred

Select preferred coordinator servers

High Availability Details

© 2005 IBM Corporation

7

Configuring High Availability coordinators is straightforward. There is a field that allows you to specify the number of coordinators that run in your Core Group. You can also choose your preferred coordinator servers from a list of the servers in your Core Group, and even specify the order in which they should be preferred.

## Section

# ***High Availability of Transaction Logs and Messaging Engines***



The next section will cover high availability of transaction logs and messaging engines.



## Transaction Log Hot Standby

- Allows failover of in-transit two-phase commit (2PC) transactions
- WebSphere Application Server V6 should be configured to store transaction logs for each server on a shared file system (such as a Network-Attached Storage device)
- When a V6 cluster member fails, then a peer is elected and directed to recover the transaction log from the failed server
  - ▶ All peers can see all other peers' transaction logs
  - ▶ On z/OS™, RRS is used.
- This allows the in doubt transactions from a failed server to be recovered very quickly
  - ▶ Huge improvement over V5, where recovery was in minutes and required OS clustering and shared disks
- This option must be enabled explicitly

The screenshot shows a configuration panel with the following fields and options:

- \* Cluster name: Cluster2
- Short name: [ ]
- Unique ID: [ ]
- \* Bounding node group name: DefaultNodeGroup
- Prefer local
- Enable high availability for persistent services

The 'Enable high availability for persistent services' checkbox is highlighted with a red border.

WebSphere Application Server V6 enables hot failover of in-transit two-phase commit transactions. To enable transaction log failover, you will need to store your transaction logs on a shared file system that is accessible from any of your servers, such as a Network-Attached Storage device. When a cluster member fails, another server will be elected to replace it, and the in-doubt transactions will be recovered from the transaction logs. Since every server can read every other server's transaction logs on the shared file system, transaction recovery is quick and easy. This is a huge improvement over V5, which required external clustering software to enable transaction log recovery. To enable failover of your transactions, you must check the box marked "Enable high availability for persistent services" when configuring your cluster. This box turns XA transaction log hot stand by, on or off. Without HA, the cells use PRR for failover. Remember both have pros and cons, but HA is the strategic direction. Non-XA data is stored in RRS. Where XA transaction data is saved, is determined in the configuration panels, and checking or not checking this box has no effect on that. However if HFS is selected and HA is checked then the HFS must be shared, i.e. available to the recovering host. Note that WebSphere data (including XA) is always stored in RRS and the XA logs only supplement this and only XA transactions do this.

## Comparison of V5 and V6

- Failure Scenario:
  - ▶ Client calls Enterprise Java™ Bean (EJB) that updates database using 2PC transactions
  - ▶ Failure during in-process transaction (after prepared statement) – database record is locked until the transaction is recovered (committed or rolled back)
- v5
  - ▶ Server needs to be restarted and recover the transaction log (PRR)
  - ▶ Can be several minutes
    - Other clients are locked out if they need the same record
- v6
  - ▶ HA Manager detects the failure
  - ▶ Failover to a peer server that recovers the transaction log, shared on a Network Attached Storage (NAS) device, from the failed server
  - ▶ Recovery time limited by file system lock release time (~15 seconds)



To highlight the improvement between V5 and V6, picture a situation where a client call results in an EJB making a database update using a two-phase commit transaction. If that transaction fails in-flight, the database record will be locked until the transaction is either committed or rolled back. With version 5, the server will need to be restarted to recover the transaction log. Depending on your configuration this can take several minutes, and other clients needing to access the same record will be locked out. In V6, the High Availability Manager will detect the process failure and fail over the service to a peer server that will recover the transaction log from the shared file system. This recovery time is measured in seconds, a clear improvement over V5.

## Section

# *High Availability Policies*

The next section will cover High Availability Policies.

## HA Policies

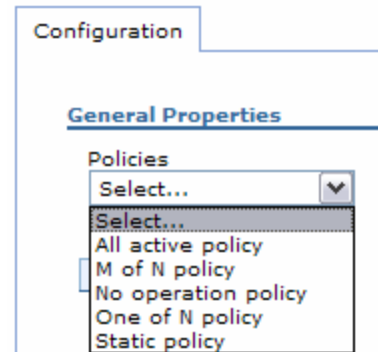
- An HA policy defines how failover occurs and which servers to use for failover
- A Core Group can have different HA policies for different services
  - ▶ For example, WLM clustering for transaction can use one HA policy, while Messaging Engines can use another HA policy
- By default, the following HA policies are defined for the DefaultCoreGroup:
  - ▶ Clustered Transaction Manager (TM) policy for clustered applications
  - ▶ Messaging policy for Service Integration Bus resources



An HA policy defines how failover occurs and which servers to use for failover. You can define different policies for different services within the same Core Group. For example the transaction manager can use one policy and your messaging engines can use a different policy. When the default Core Group is created, a clustered transaction manager policy and a messaging policy for Service Integration Buses are automatically created.

## HA Policies

- Singleton services can be configured to run in 3 basic modes
  - ▶ Static
  - ▶ 1 of N
  - ▶ No Operation
- Other modes are available but as useful for TM or messaging failover



You can also create your own custom policies to define how singleton services should fail over. The three basic modes that can be configured are Static, 1 of N, and No Operation. These three policies will be explained in the following slides. There are several other policies, but those are not as useful for transaction manager or messaging failover, and they will not be covered in this presentation.

## HA Policy: Static

- Makes WebSphere Application Server V6 act like V5
- The singleton is only placed by WebSphere Application Server on a single fixed server
- This means that if that server is not running, then the singleton service is not running
  - ▶ However, the fixed server can be changed at any time without restarting, which is different than V5
- If failover is required then the whole node needs to be failed over using PRR.



If you choose the “Static” policy, your V6 environment will act as if it were a V5 environment. This means that singleton services run on a single server, and the failure of that server will not result in another server recovering that service. The service will only reactivate when the failed server is restarted. One difference from V5 is that you can explicitly change the fixed single server at any time, without restarting any servers. If you are using the “Static” policy, but require failover of singleton services, you will need to fail over the entire node to a backup node using PRR.

## HA Policy: 1 of N

- HA Coordinator decides on which candidate server the singleton services can be started
- All external resources must be available to all possible candidates
  - ▶ For messaging this implies a remote database or a cloudscape database on a NAS mounted on all candidate machines
  - ▶ For transactions this implies the transaction logs are on a NAS mounted on all candidate machines
- Coordinator will keep the singleton service running on exactly one of the candidate servers using this mode



The “1 of N” policy enables the HA Coordinator to manage failover. The coordinator determines which processes should run which services and assigns them accordingly. When failure occurs, the Coordinator can fail over your services to any other server, provided that any external resources such as databases or transaction logs are stored in a way that they are accessible to all servers. The coordinator will ensure that the singleton service is running on exactly one server at any given time. This is the suggested HA policy for most users.

## 1 of N: Additional Options

- Preferred Server list
  - ▶ Specify a list of servers that the HA Manager will prefer when choosing where to run a singleton
- Preferred servers only
  - ▶ Defines whether the list is exclusive (only run on those in the list), or WebSphere Application Server will use the list, but if all preferred servers are down it can choose a server from the remaining servers
- Fail back
  - ▶ If a more preferable server becomes available, move the service back to that server
- Quorum
  - ▶ Need a quorum (half the servers) to start the singleton services
- All the above configuration can be changed without restarting the Application Server
  - ▶ *cron* can be used to change configuration using a schedule

Additional Properties	
<input type="checkbox"/>	<a href="#">Custom properties</a>
<input type="checkbox"/>	<a href="#">Match criteria</a>
<input type="checkbox"/>	<a href="#">Preferred servers</a>

General Properties	
* Name	Sample 1 of N Policy
* Policy type	One of N policy
Description	
* Is alive timer	5 seconds
<input type="checkbox"/>	Quorum
<input type="checkbox"/>	Fail back
<input type="checkbox"/>	Preferred servers only

When you are creating a “1 of N” policy, there are several additional options that allow you to customize how failover occurs. You can enter a list of “preferred servers”, telling the HA Manager to run singleton services on the listed servers if possible, but if no preferred servers are available, the services will run on another server. If the “preferred servers only” box is checked, singleton services will run only on servers in the preferred list, and if none of them are available, the singleton will fail. The “Fail back” checkbox tells the HA Manager that after a process has failed over, the process should be moved back to a preferred server when that server becomes available. If unchecked, a failed-over service will remain running in the server to which it failed over. The “Quorum” checkbox specifies that singleton services should only be started if at least half of the servers in the Core Group are running. This means that if more than half of your servers fail, your singleton services will be automatically stopped. Use this feature with caution.

Any of these options can be changed dynamically using *wsadmin*. This allows you to use a service like *cron* to change the Core Group options on a schedule, if you require different policy options at different times of the day or week.



## Example: 1 of N Policy Options

- Servers elect one server to run each service
  - ▶ “Special election” held when that server is no longer available
- Configurable options:
  - ▶ Preferred server list – {A,C}
  - ▶ Preferred servers only
    - True - Try A, try C, then fail
    - False - Try A, try C, try anything else visible
  - ▶ Fail back
    - True - A dies, C takes over, when A recovers, A takes over
    - False - A dies, C takes over, when A recovers, C stays



As an example of how this policy works, imagine you have three cluster members named A, B, and C. When the cluster starts up, the servers will decide what services will run on each server. Assume that a singleton service has been elected to run on server A, and that you have defined a preferred server list of A then C. If you check “preferred servers only” the HA Manager will try to run your singleton services on A, and if A is not available, C. B is not allowed to run that singleton service. If “preferred servers only” is not checked, then it will also try to run the service on server B. Enabling the “Fail back” option means that after a service from A has failed over to B or C, the service will be moved back to server A as soon as it becomes available again.

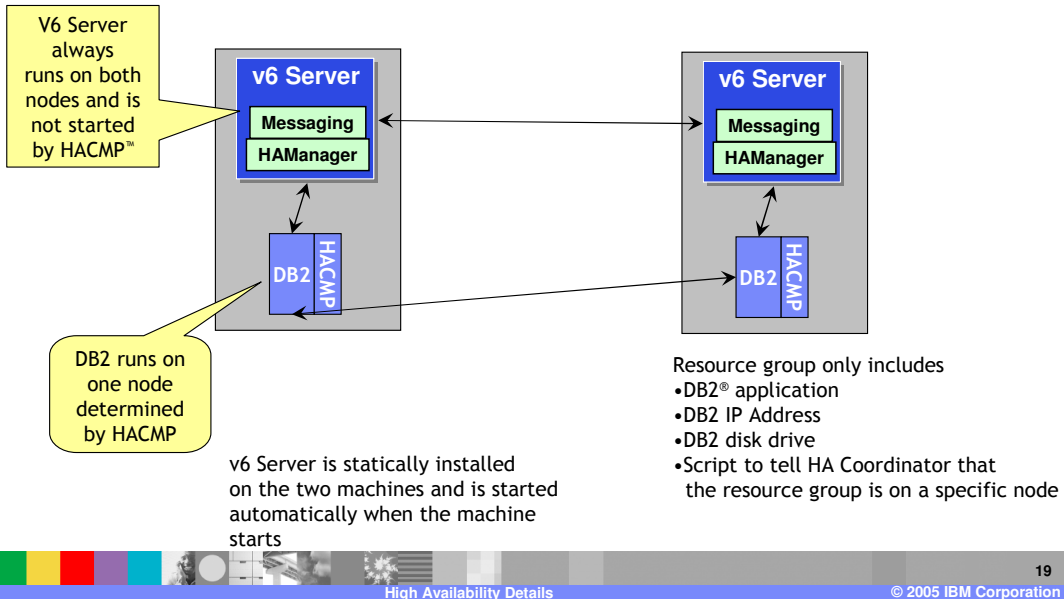
## HA Policy: No Operation

- WebSphere Application Server never activates the singleton on its own
- For failover, a third party must use JMX to tell the HA Coordinator where to place a group of singletons, which are grouped together using a keyword
- Typically, this mode is used when the singleton has dependencies on resources managed by external clustering software
  - ▶ Transaction logs may be on a shared disk that is only mounted on a single server at a time
  - ▶ Messaging may need to use a co-located database which is managed by external clustering software
- This allows external clustering software to leverage the hot standby capabilities of WebSphere Application Server V6, reducing recovery times from minutes to seconds



The “No Operation” policy specifies that WebSphere Application Server will never activate singletons on its own. Instead, it will wait to be told what to do using JMX commands. This mode is needed when an application depends on external resources that are externally managed by clustering software. For example, the Application Server might be co-located with a database, as shown in the example on the next slide. The value of this policy is that it allows you to make the HA Coordinator aware of the actions taken by external clustering software, so that applications will always have access to the necessary resources.

## Example: No-Operation Policy



This example shows WebSphere Application Server collocated with a DB2 database and configured on two servers in a SYSPLEX. JMX calls to tell the Application Server when the resource group fails over to the other machine. When this JMX command is executed, the Application Server on the second machine will take over the singleton services, recover the transaction logs from logstreams or shared DASD, and use the collocated DB2 instance that is installed on the local machine. The DB2 data is still available by DB2 data sharing.

## Section

# ***Failure Detection***

The next section will discuss automatic failure detection.

## Failure Detection: TCP Keep-Alive

- Open a keep-alive socket between peers
  - ▶ If the socket to another peer is closed then that peer is suspected
- This works well on machines that may swap or become unresponsive
- Tune the operating system's `KEEP_ALIVE` setting to detect failure in an acceptable time



The HA Manager uses two parallel methods to monitor processes: TCP Keep-Alive sockets, and an active heart beat. Both of these methods are used across the core group in a peer-to-peer fashion. If all of the sockets from a server to its peers are closed, then it is presumed to have failed. This method is the most reliable method for detecting failures on machines that may swap processes out of active memory or servers that may respond slowly, because it does not require the process itself to issue a response. Since WebSphere Application Server processes on z/OS will tend not to swap out of active memory or become slow unless the LPAR becomes slow, the best practice would be not to change the system Keep-Alive timeout value to accommodate HA. In addition, such a change would affect system wide performance. The HA heart beat mechanism is the preferred monitoring method.

## Failure Detection: Active Heart Beat

- Active heart beat
  - ▶ All processes send a heart beat to each other every  $N$  seconds
  - ▶ If a process does not receive a heart beat from a peer for  $M$  of these intervals then it tells the others to suspect that peer
  - ▶ These values are fixed



The second method used to determine server status is an active heart beat. Each process sends a heart beat message to every other process once per  $N$  seconds. If a peer fails to respond to  $M$  consecutive heart beats, then services from that process will be failed over to other processes. The closure of the Keep-Alive socket will usually identify a failure before the heart beat will, but the heart beat method is necessary in cases where a TCP socket cannot be kept open.

## Summary

- New V6 High Availability Services provide impressive levels of availability quicker than PRR
- HA is currently the strategic direction of WebSphere
- A Core Group defines the set of processes that can provide each other with fail over capability
- The HA functions can be customized by creating HA policies

In summary, this presentation has focused on the new High Availability Services of WebSphere Application Server V6. These services provide levels of uptime that could previously only be achieved using PRR, ARM and user created and maintained scripts. V6 also provides this availability at significantly lower levels of cost and complexity.

HA is the strategic direction of WebSphere.

The key configuration object in configuring high availability is the Core Group – it defines the set of processes that can provide each other with fail over capability.

The high availability functions can be customized to meet your requirements by creating HA policies for your Core Groups. HA policies decide how and where services will be failed over.

## Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM (logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)/business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2004. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.