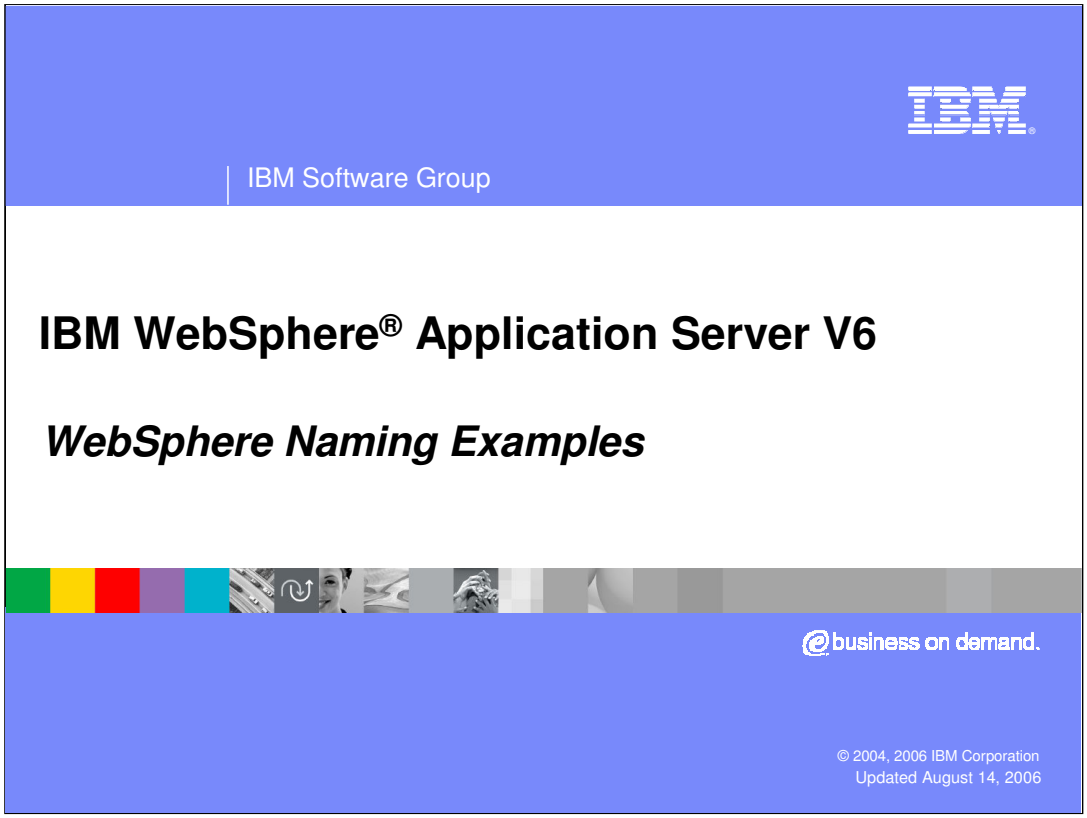
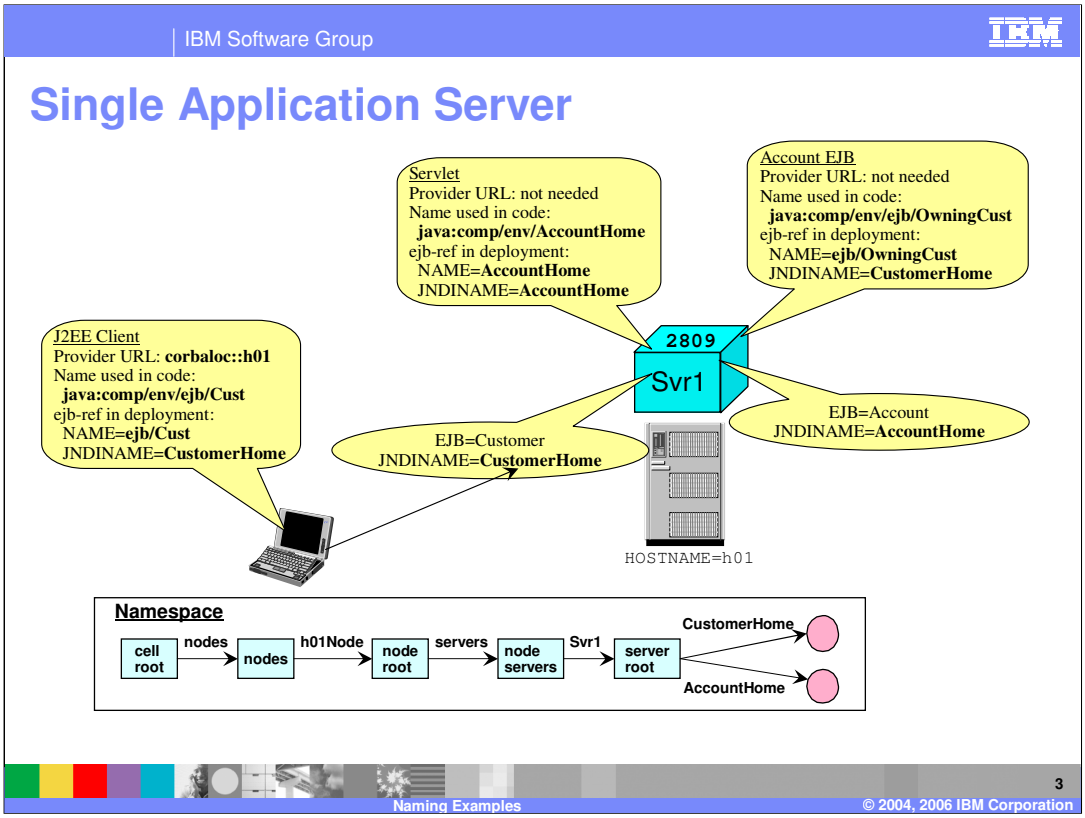


This presentation will focus on Naming examples and scenarios.

Goals

- Illustrate aspects of naming for various topologies
  - Provider URL used to connect to name space (host/port)
  - “java:” name used by the code
  - Name of the Enterprise Java™ Bean (EJB) in the global name space
  - Contents of EJB Reference in Deployment Descriptor
- Topologies:
  - stand-alone application server
  - stand-alone application server on non-default port
  - Two stand-alone application servers on the same box
  - Two network deployment servers on the same box
  - Two cluster members, node agent bootstrapping
  - Two cluster members, cluster member bootstrapping

The following slides present examples of the use of naming in increasingly complex scenarios. Each scenario has a defined topology and uses sample clients to illustrate the use of the provider URL (that is, the host and port for bootstrapping), the use of the “java:comp/env” name specified in the code, and the EJB Reference in the deployment descriptor which defines the name of the target EJB in the global name space. The scenarios cover stand-alone servers through network deployment configurations including strategies for high availability of the name space.



This is the first topology, a stand-alone Application Server. This will be looked at in detail so as to highlight everything that is being shown in this slide.

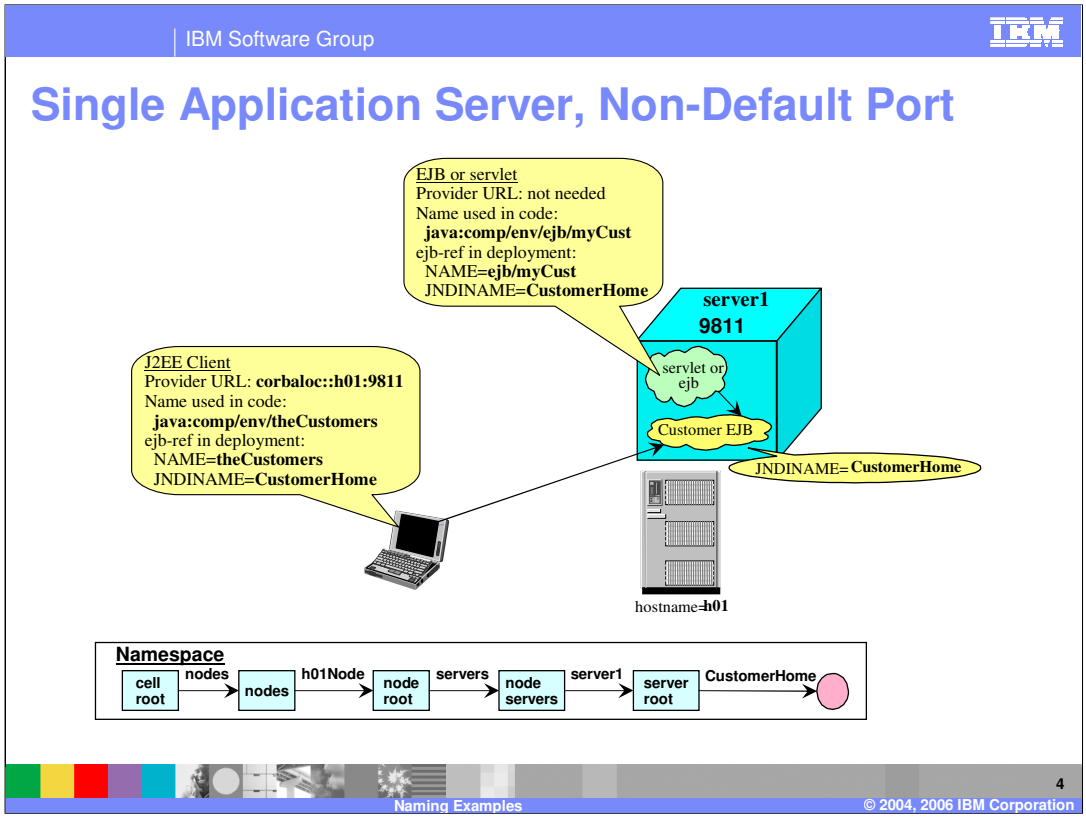
Starting with the blue box there is a stand-alone WebSphere Application Server which is named “server1” and is using port 2809 for bootstrapping. This is running on a machine with the hostname of “h01”. This server has two EJBs installed, one being the Customer EJB with a JNDI name of CustomerHome, and the other being the Account EJB with a JNDI name of AccountHome.

On the bottom of the slide, is a picture of the global name space for this server. You will notice that both the CustomerHome and the AccountHome are bound into the server root context. Also notice, that even though this is a stand-alone server, the fully architected global name space starting from the cell root is present within the server.

Each of the yellow callouts represent a scenario of an EJB client accessing an EJB home.

Starting with the J2EE client in the lower left, you will see that the client used a provider URL of “corbaloc::h01” to bootstrap into the server’s name space. The “h01” identifies the host, and there is no port specification because the server is listening on the default port of 2809. Within the J2EE client code, there is a lookup of the name “java:comp/env/ejb/Cust”. In the deployment descriptor, there is an EJB Reference with the name of “ejb/Cust” which associates it with the “java:comp/env” name used in the lookup. The EJB Reference then identifies a JNDI name of “CustomerHome”. Since the bootstrapping places the client in the server root context, the name “CustomerHome” is sufficient to access the Customer EJB home.

Moving to the scenario in the center, notice that there is a servlet running in server1 that needs to access the home for Account EJBs. Similar to the J2EE client, there is a “java:comp/env” name in the code that is associated with an EJB Reference in the deployment descriptor that then identifies the JNDI name “AccountHome” for performing

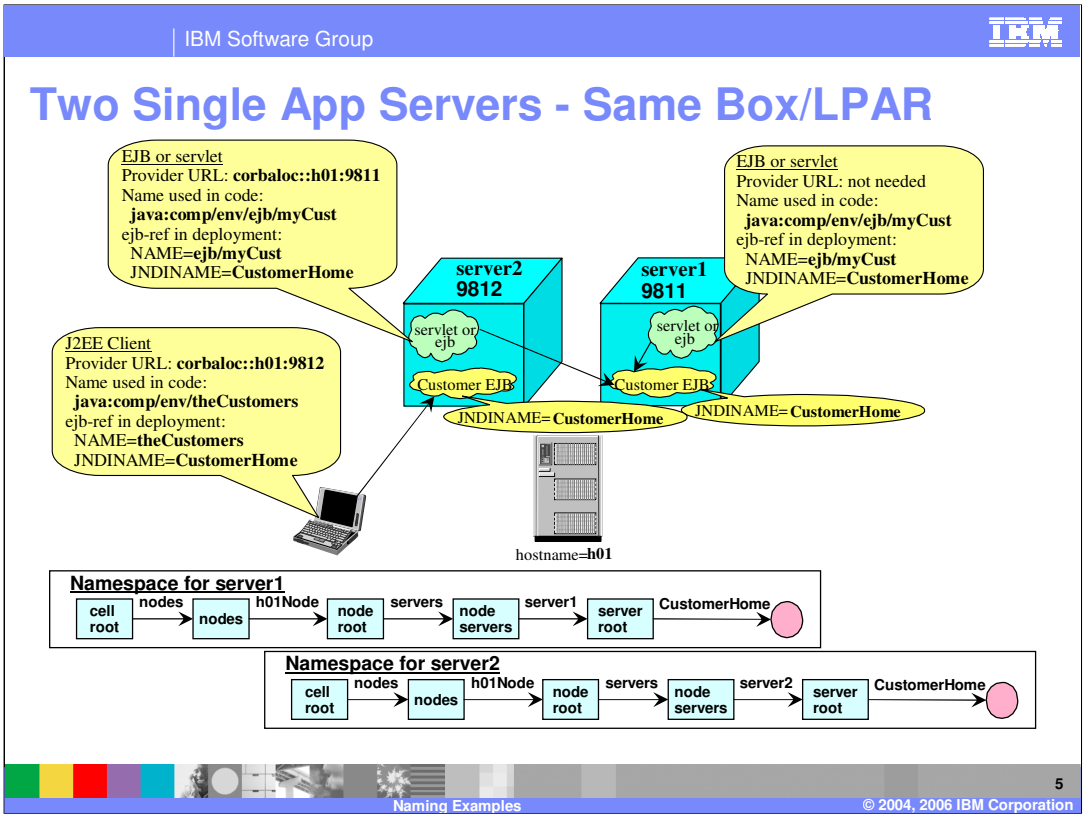


The second topology is also a stand-alone application server and differs from the previous slide only in that server1 is now listening on a non-default port, in this case port 9811.

To simplify the slide and description, the scenario shows only the Customer EJB. Also, the servlet client and EJB client were combined as there is essentially no difference in these two scenarios.

The J2EE client is identical to the previous slide with the exception of the provider URL. Because the server is not listening on the default port, the port must be specified, as is seen in the URL “corbaloc::h01:9811”.

As for the servlet and EJB clients, their scenarios are identical to the previous slide. Since the provider URL is not needed with a server, the fact that the server is not listening on the default port does not need to be considered.



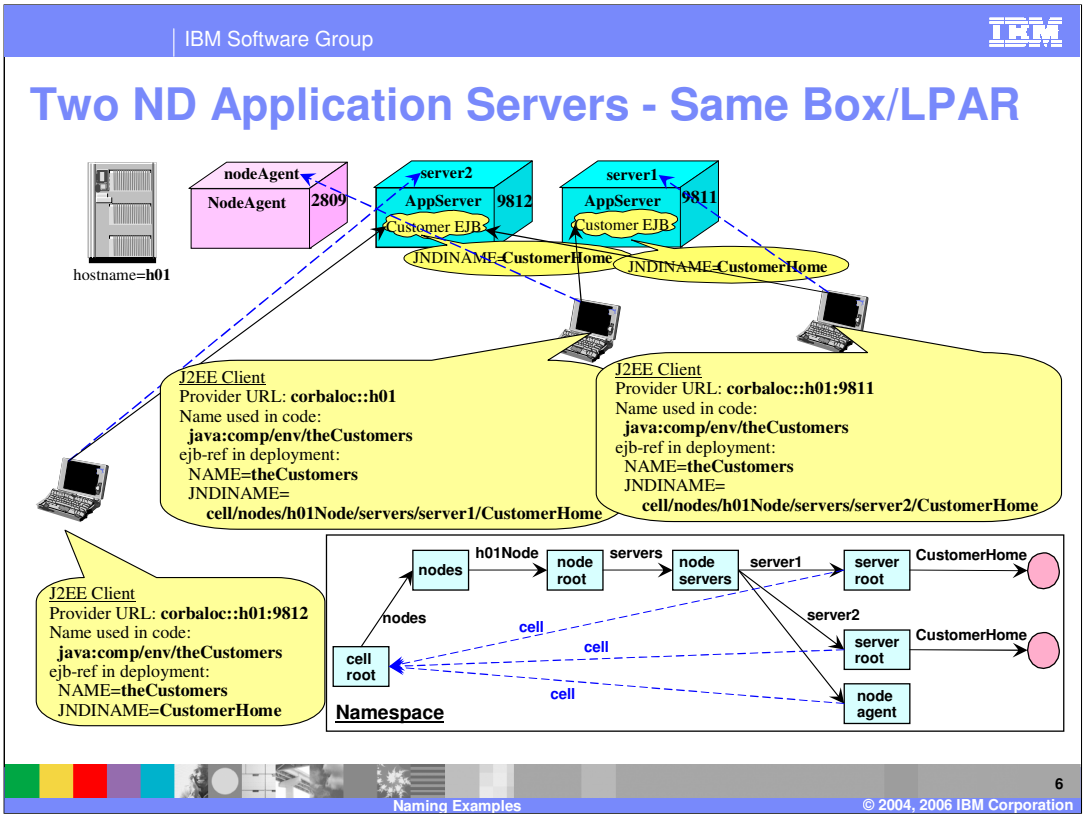
Next, look at two stand-alone application servers which co-reside in the same machine or same logical partition within a machine – in either case within a single operating system instance. These application servers are set up just like the example on the previous slide, with the exception that server2 is listening to a different bootstrap port, 9812. The two servers must listen on different bootstrap ports because they reside within the same operating system instance.

Notice the name space for server1 and server2. Since these are both stand-alone servers the two name spaces are independent of one another yielding two independent global name spaces.

Start with the J2EE client in the lower left. This is the identical scenario to the J2EE client on the previous slide. It is the port assignment in the provider URL which determines which server will be connected to and therefore which Customer EJB Home will be accessed.

In the upper right there is an EJB or servlet running in server1. Just as the previous slide, by not using a provider URL, it is connected with the name space in server1 and therefore will access the Customer EJB Home in server1.

So far the usage scenarios are no different from the previous slide. What is new with this slide is in the upper left side. Here there is an EJB or servlet running in server2, but it did use a provider URL with port 9811 specified and will therefore access the Customer EJB Home that is in server1.



Now move to a Network Deployment topology. As in the previous slide, server1 is listening on port 9811 and server2 listening on 9812, but this time they are federated into a cell and there is a Node Agent running which listens on port 2809 (the default bootstrap port).

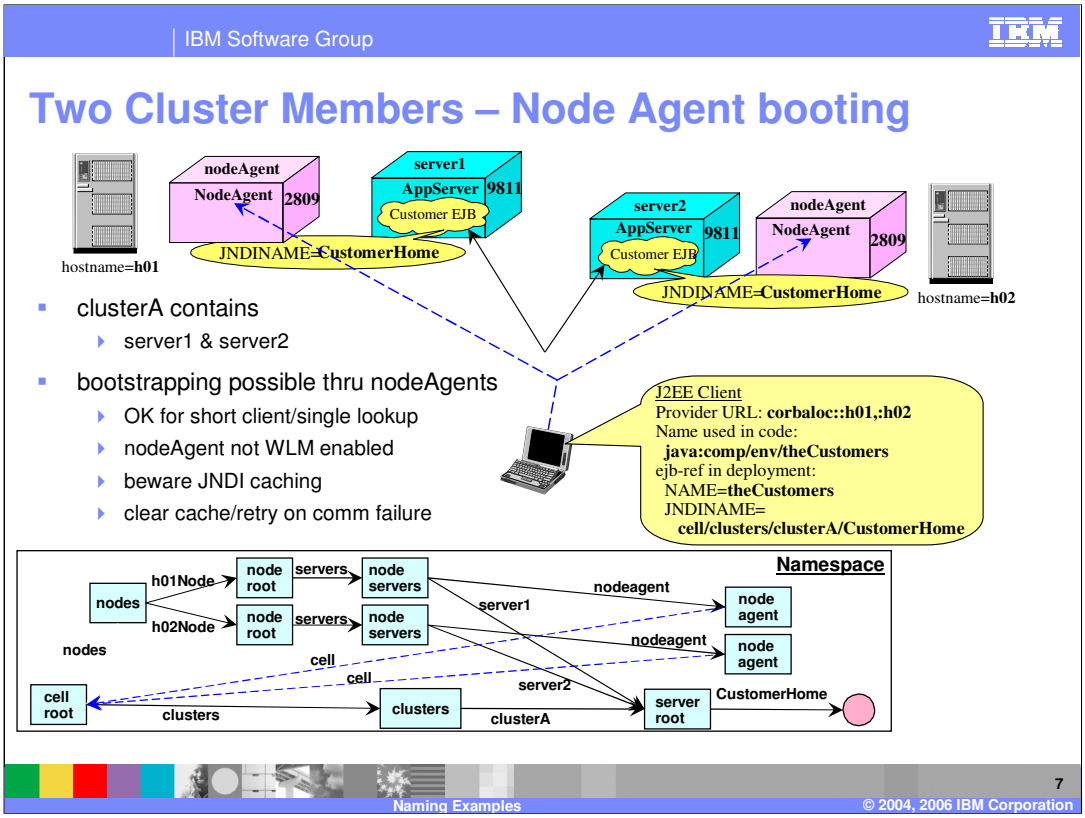
The first thing to note is the name space. Since this is a Network Deployment environment, the global name space is a single logical name space rather than independent name spaces.

It is now possible to bootstrap into one server and access an EJB Home from another server through the use of fully-qualified names.

The first J2EE client to look at is in the lower left. In the picture, the dashed blue arrows represent bootstrapping and the solid black arrows represent the Customer EJB Home that is being accessed. The provider URL uses port 9812 pointing to server2 and the JNDI name of “CustomerHome” is a simple name and therefore it is the Customer EJB Home in server2 that is accessed by this client. This scenario is no different than the ones examined on the previous slides.

The next J2EE client to look at is in the middle. The provider URL used is “corbaloc::h01”, and since no port is specified the client bootstraps to the Node Agent as it is listening on the default port of 2809. Notice that the EJB Reference in the deployment descriptor uses a fully-qualified name of “cell/nodes/h01Node/servers/server1/CustomerHome“ and will therefore access the Customer EJB Home that is in server1. Notice that the fully-qualified name started with “cell” which points to the cell root context, thus making the entire name space accessible.

The J2EE client on the right bootstraps into server1 because the provider URL specified port 9811. However, even though bootstrapped into an Application Server, the EJB Reference contains the fully qualified name “cell/nodes/h01Node/servers/server2/CustomerHome“ and will therefore access the Customer EJB Home that is in server2. Basically, the key point to take away from here is



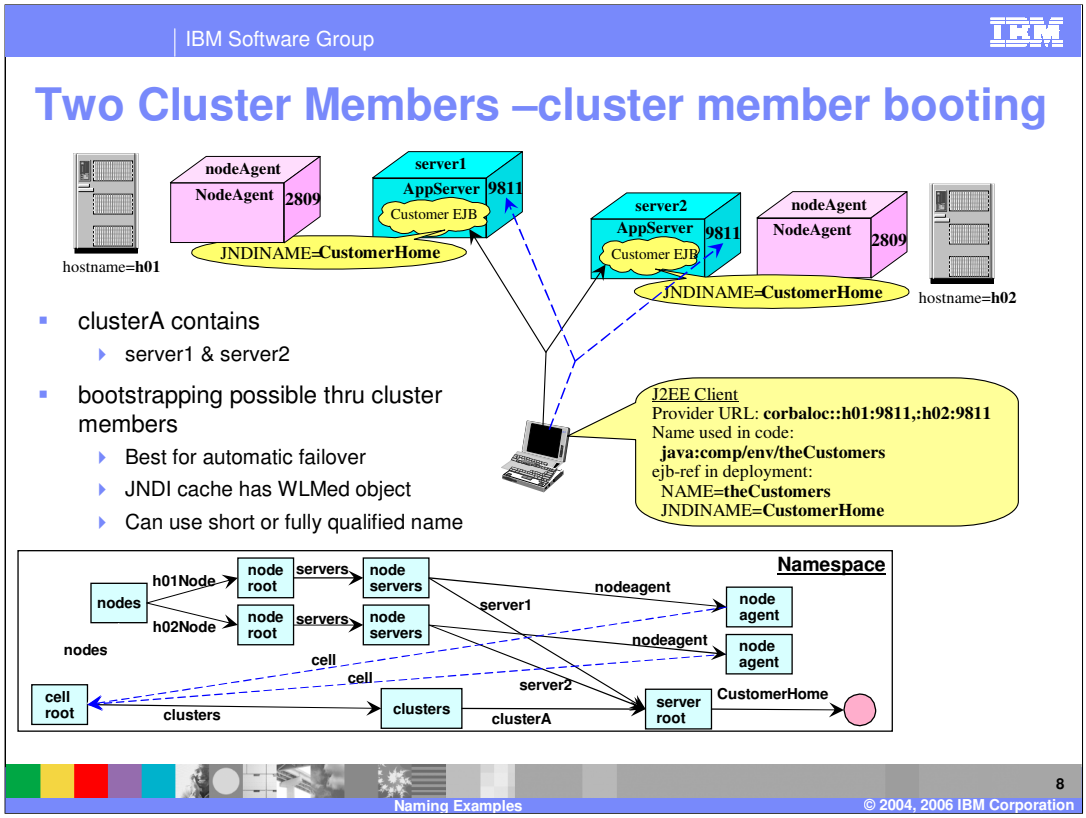
This slide looks at servers that are cluster members. The configuration is different from the previous slide in that there are now two machines (or LPARs) with a WebSphere Application Server node and corresponding Node Agent. The nodes are federated into the same cell. There are still server1 and server2 but now they are members of the same cluster. Because they are running in different operating system instances, they can use the same port. They both are listening on port 9811 and both node agents are listening on port 2809.

The next thing to look at is the name space. Since both server1 and server2 are members of the same cluster, they have the same server root context in the logical global name space (although each server has its own physical instance of this server root). Also notice the addition of the “clusters” context off of the cell root. Basically, with this configuration, a lookup of the Customer EJB Home will result in a reference which would be workload-managed and provide access to the home using either server1 or server2.

This type of configuration is used for workload balancing and for high availability with failover. The question is, how does bootstrapping into the name space fit in this environment? Can it be highly available and can there be failover? This slide and the next slide will look at two approaches to this.

This slide discusses the use of bootstrapping through the Node Agents. Notice the provider URL is “corbaloc::h01,:h02” which has two hosts specified and no ports specified, thus using the default port of 2809. What this results in is an attempt to bootstrap into the node agent running on h01 and, if for some reason that should fail, the attempt is made to bootstrap into the node agent on h02. This provides a level of high availability in the bootstrapping operation.

The JNDI name that is used is “cell/clusters/clusterA/CustomerHome” and once that is looked up, the reference to the customer home is workload-management enabled and will allow for failover.



This configuration is the same as the previous slide. However, this example examines cluster member bootstrapping.

Notice the provider URL is now specified as “corbaloc::h01:9811,:h02:9811”. This causes the bootstrapping to initially try to access server1, and if that does not work, to access server2. The NamingContext that the JNDI cache saves in the cache is from one of these servers. Since they are workload managed there will be failover when using this NamingContext from the JNDI cache. This is the recommended pattern for clients which are long running or need to perform multiple lookups from the name space.

Summary

- Examined multiple topologies
- Illustrated naming usage in each topology
  - Provider URL, host/port
  - java: name
  - Reference
  - Global name
- Topologies:
  - stand-alone application server
  - stand-alone application server on non-default port
  - Two stand-alone application servers on the same box
  - Two network deployment servers on the same box
  - Two cluster members, node agent bootstrapping
  - Two cluster members, cluster member bootstrapping

This presentation provided examples of the use of naming in increasingly complex scenarios. Each scenario had a defined topology and then used sample clients to illustrate the use of the provider URL, the use of the “java:comp/env” name, and the EJB Reference which defines the name of the target EJB in the global name space. The scenarios covered stand-alone servers through network deployment configurations including strategies for high availability of the name space.

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.