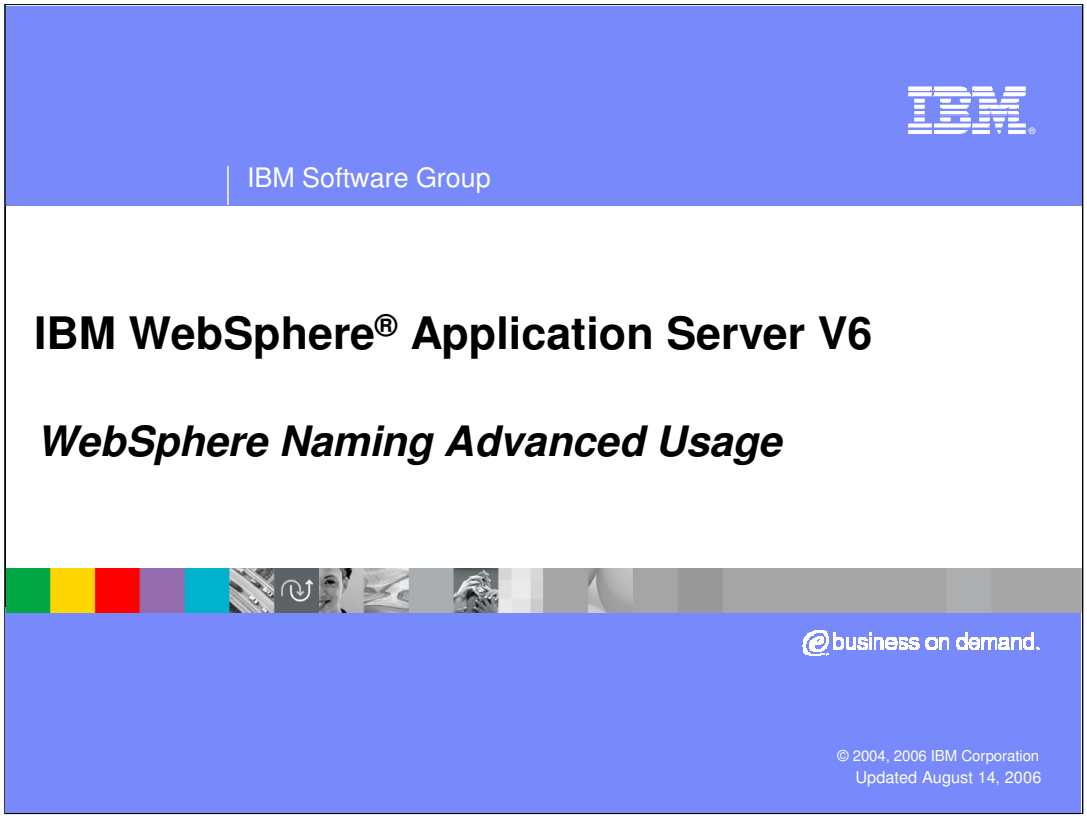


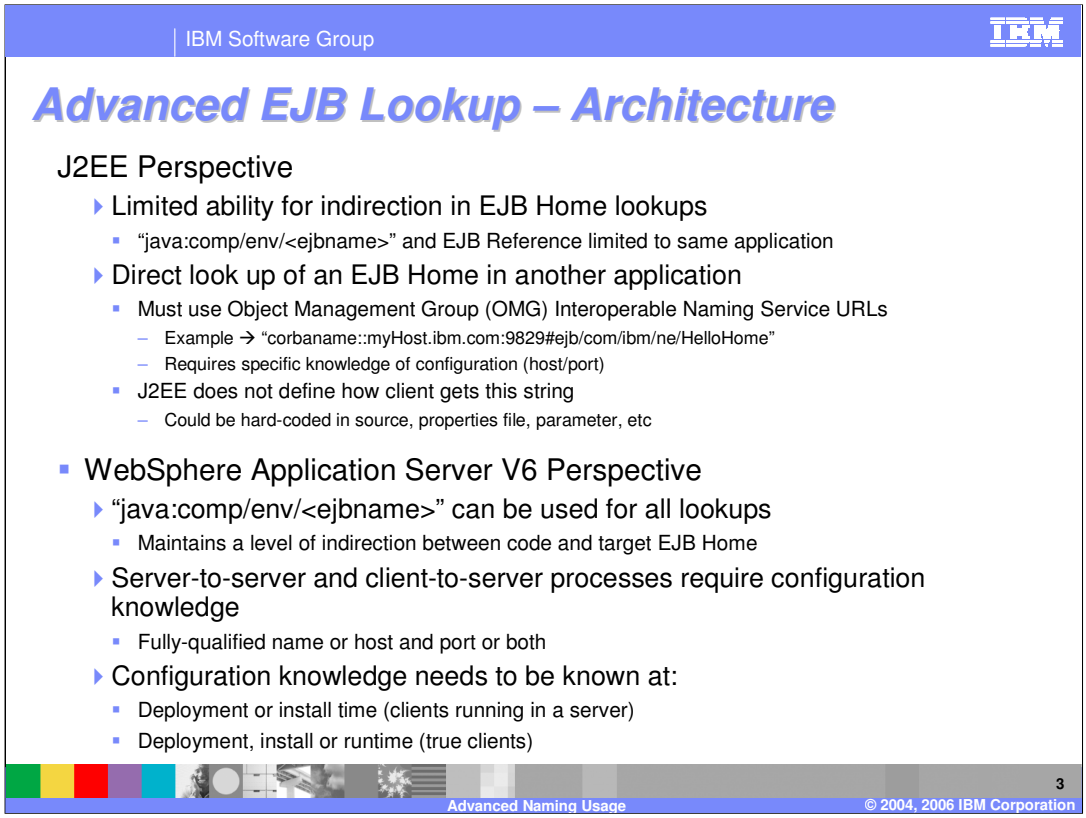
This presentation will explore advanced usage of Naming in WebSphere Application Server V6.

Goals

- Architectural view of Java™ 2 Enterprise Edition (J2EE) requirements and V6 extensions
  - From a multiple application and multiple server perspective
- Provide advanced understanding of how to use Naming
  - Approach from Network Deployment configured environment
- Understand characteristics of the global name space
  - Use of fully-qualified names
  - Host and port usage for bootstrapping
- Understand which server to connect to for naming
  - Simple versus fully-qualified names
  - Node agent bootstrapping
  - Clustered server bootstrapping
- Topology Dependent Name issue
  - Understand the problem
  - Best practices to avoid the problem

© 2004, 2006 IBM Corporation

The goal of this presentation is to present the in-depth knowledge required by developers and administrators regarding the use of Java™ Naming and Directory Interface (JNDI) names and naming functionality in a multi-server Network Deployment environment. It approaches this by first reviewing both the J2EE and WebSphere Application Server architecture that applies to performing naming lookups of Enterprise JavaBean (EJB) Homes across server processes. Knowledge of the architecture of the global name space is an important foundation to understanding this topic, so the information about the global name space from the Naming Introduction presentation is reviewed and expanded upon. When dealing with naming in the Network Deployment environment, the concepts of fully-qualified names and also bootstrap host and port need to be considered. These topics are examined in detail. When designing your use of naming for performing EJB Home lookups, an important consideration is which server you plan to bootstrap to when connecting to the global name space. The factors involved in this consideration are examined. Finally, the issue of how topology dependent names can hamper your ability to easily reconfigure your environment is explained along with some suggested best practices to avoid this issue.



J2EE defines a programming model for accessing EJB Homes which are not in the same application, possibly not even within the same server process. The level of indirection provided by use of "java:comp/env" names defined by J2EE for accessing EJBs in the same application does not apply in this case. Rather, the J2EE specification makes use of the OMG Interoperable Naming Service defined "corbaname:" URL strings. Using this mechanism requires knowledge of host and port information for contacting the global name space. The URL needs to be used directly in the JNDI lookup, and the J2EE specification provides no guidance on how that string is to be provided. Therefore, it is up to the user to determine how to isolate the source code from the host and port configuration information.

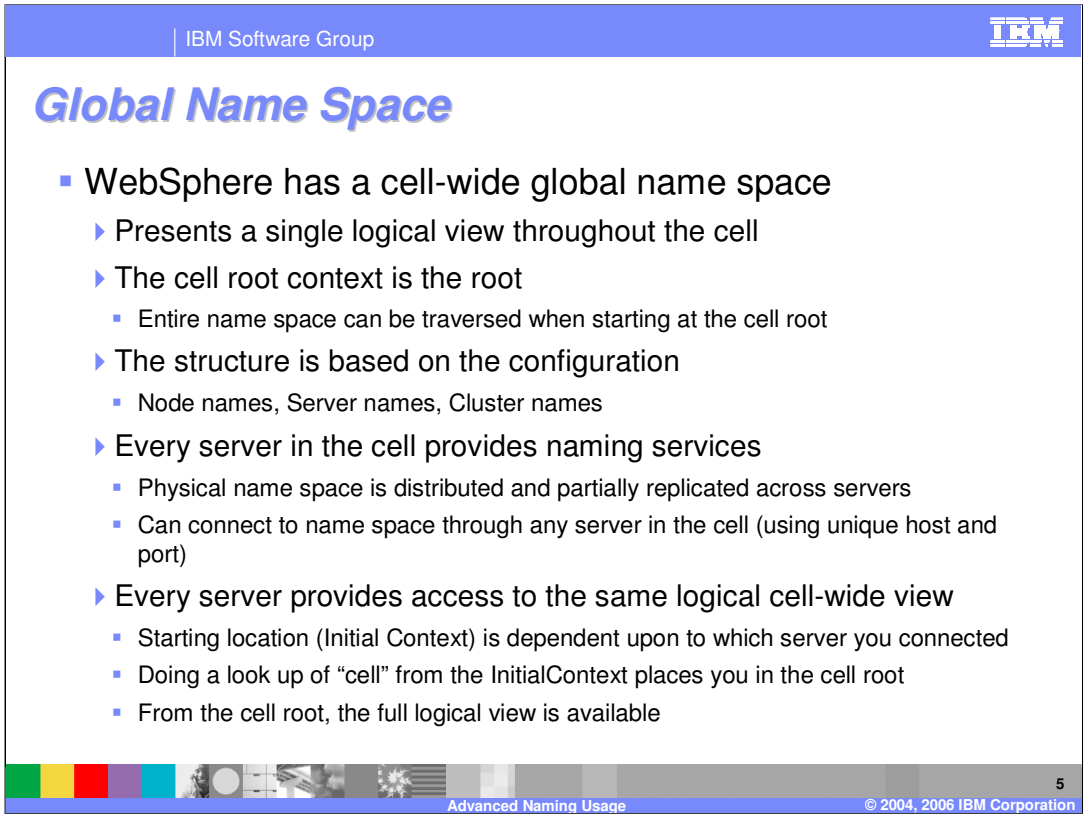
WebSphere Application Server extends this model to allow the use of "java:comp/env" names which maintains the same programming model as is used in the single application case, thus maintaining the same use of indirection and isolating the source code from potential changes. However, this does not totally eliminate the need to understand the configuration. The JNDI name used in the EJB Reference must either contain a fully-qualified name or host and port information. This information can be supplied at deployment of the application or during installation of the deployed application.

Section

Understanding Fully-Qualified Names

© 2004, 2006 IBM Corporation

The following slides discuss the global name space structure and the use of fully-qualified names to look up EJBs in the global name space.

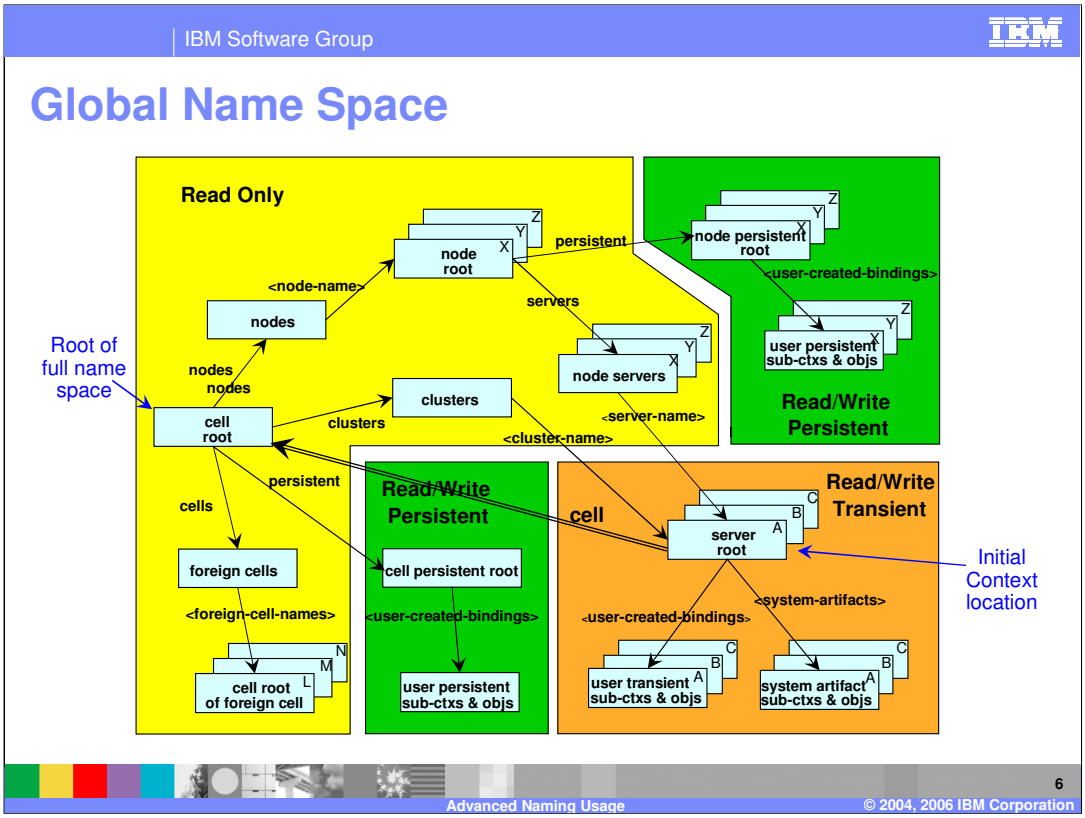


The architecture and concepts of the global name space were presented in the Naming Introduction presentation, but due to the importance in understanding these concepts they will be reviewed in the next couple of slides.

In WebSphere Application Server, there is no separate Name Server process. Rather, every server in the environment provides the functionality for the global name space, including all application servers, the node agents, and the deployment manager. From the perspective of a WebSphere Application Server Network Deployment cell, these servers work together to provide you with a single logical view of the name space although the physical representation of the name space is not fully contained in any one server. The physical representation of the name space is partially replicated across the server processes and partially distributed across the server processes.

When a WebSphere Application Server starts, the subset of the logical name space which will be physically represented within that server is built in the server’s memory. When accessing the global name space, any server can be used as a starting point. Each server in the cell has a unique host and port combination which can be used to connect (bootstrap) into the name space in that server. For code already running in a server, by default it will be connected to the name space within the same server.

Once connected to the name space, any portion of the logical name space can be traversed without regard to which server or servers host that part of the name space. There is a cell root context from which the entire name space is accessible. However, by default, when bootstrapping to the name space, the JNDI InitialContext will be positioned at the server root context for the server you bootstrap into. From this location, a look up of



This is a pictorial representation of the global name space. This picture is rather extensively described in the Naming Introduction presentation. At this point it is useful to review some of the highlights and important points which will serve as a basis for understanding the use of fully-qualified names in WebSphere Application Server Network Deployment environments.

The light blue rectangles represent naming contexts in the name space. The text within a rectangle is a label which describes the purpose of the context. The arrows represent bindings in the name space and the text by an arrow is the name of the binding. This would be the actual name used when traversing the name space. The logical name space starts at the context in the center of the left-hand side of the picture, with the label “cell root”. Starting from the cell root context, the entire logical name space can be traversed. The structure of the name space is based on the topology of the WebSphere Application Server cell. There is a context for each node in the cell (the “node root” shown in the upper center). There is also a context for each server in the cell (the “server root” shown in the lower right). For example, if there was a “server1” on “nodeA”, from the cell root context the name “nodes/nodeA/servers/server1” would be used to traverse to the server root context for server1. Another example would be a cluster named “clusterX” with cluster members “x1” and “x2”. From the cell root context, a look up of “clusters/clusterX” would be used to traverse to the server root context for one of the cluster members, either “x1” or “x2”.

Examine the portion of the name space within the yellow background. As you can see, the contexts within this area all pertain to the topology of the cell, reflecting information about the cell, its nodes, clusters, and servers. This yellow part of the name space is replicated in every server in the cell and is built using configuration data from the system management repository. Now examine the portion of the name space with the brown background in the lower right-hand corner of the picture. This shows the “server root” context. There is a unique server root context for every server in the cell. The server root context is contained in the server that it represents, and it is not replicated in any other server. The context labeled “system artifact sub-ctxs & objs” refers to the EJB Homes and Resources for the server.

Fully-Qualified Names - Introduction

- When can simple names be used?
  - Lookups within the same server
  - When connected directly to the name space of the server containing the target of the lookup
- When can fully-qualified names be used?
  - Server to server within a cell
  - When connected to the name space of any server in the cell
  - Fully-qualified names can also be used in the simple name scenarios
    - If you are not sure, using a fully-qualified name will work
- A fully-qualified name starts at the cell root context
  - "cell/……." from the Initial Context

© 2004, 2006 IBM Corporation

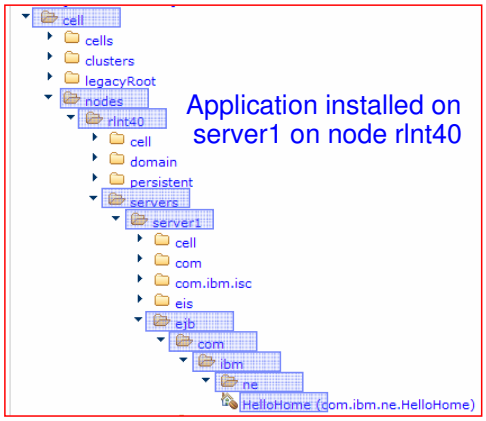
In the WebSphere Network Deployment environment, there are times when simple names can still be used, such as is done with a stand-alone application server. For example, code running in a server which is looking up an EJB Home which is installed in the same server, can use a simple name. Also, client code in a different process can use a simple name if bootstrapped directly into the server which contains the EJB Home being looked up.

Fully-qualified names must be used when doing a lookup of an EJB Home that is in another server from the one where the lookup was initiated. Also, a client process that bootstraps into a different server than the one where the target EJB Home is located must use a fully-qualified name. An example of this would be a client that chooses to use a default bootstrap port assignment. This client will bootstrap into a node agent, and then the fully-qualified name must be used to look up the EJB Home in an Application Server. As discussed in the previous slide, starting a name with "cell" will position you in the name space for use of a fully-qualified name.

It should be noted that even in cases where simple names will work, fully-qualified names will also work.

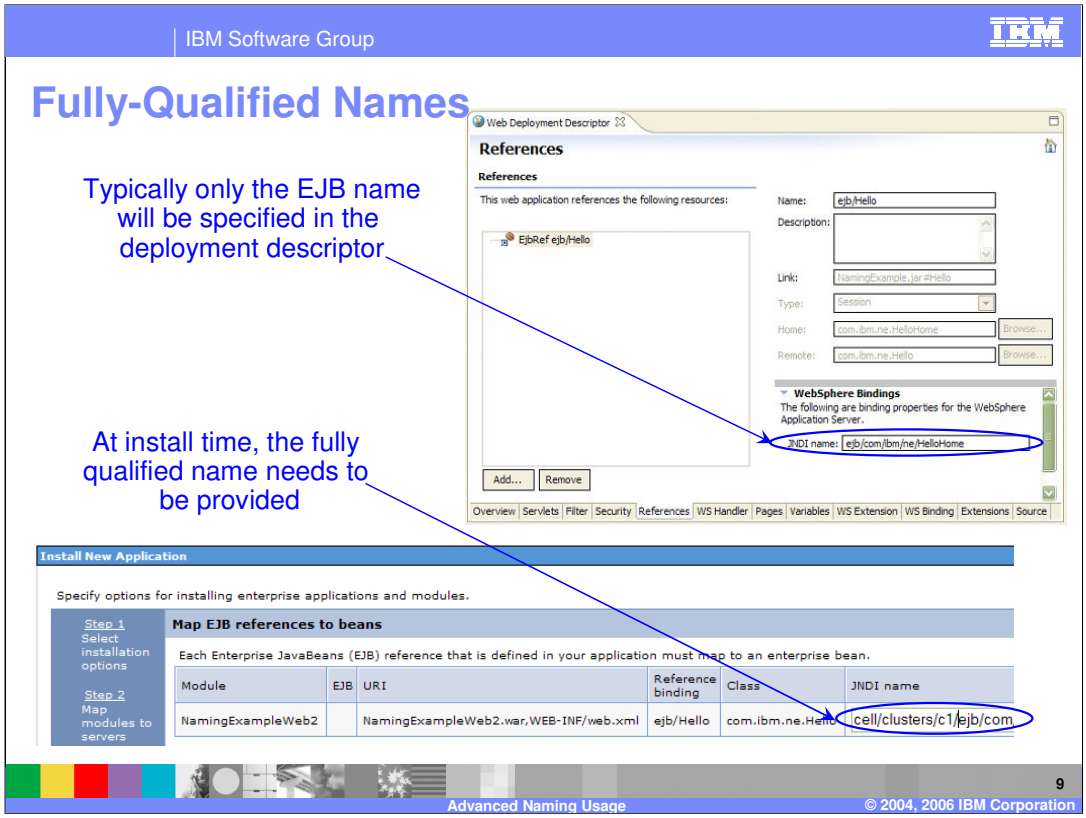
Fully-Qualified Names - Examples

JNDI name in EJBs deployment descriptor	ejb/com/ibm/ne/HelloHome
Application installed on server1 on node rlnt40	cell/nodes/rlnt40/servers/server1/ejb/com/ibm/ne/HelloHome
Application installed on cluster5	cell/clusters/cluster5/ejb/com/ibm/ne/HelloHome



© 2004, 2006 IBM Corporation

This slide shows an example of a fully-qualified name. There is an EJB whose deployment descriptor defines the JNDI name for the EJB to be “ejb/com/ibm/ne/HelloHome”. Assuming that this gets installed on a server named “server1” on the node “rlnt40”, a fully-qualified name for this would be “cell/nodes/rlnt40/servers/server1/ejb/com/ibm/ne/HelloHome”. The screen capture at the bottom of the slide further illustrates this example. Alternatively, assume that the EJB has been installed into a cluster named “cluster5”; the fully-qualified name would be “cell/clusters/cluster5/ejb/com/ibm/ne/HelloHome”.



In most cases, a deployed application EAR will have an EJB Reference in the deployment descriptor that only specifies the target EJB's simple JNDI name. It would normally be during the installation of the application that the administrator would modify this to be a fully-qualified name, as that is when it is most likely known where the target EJB has been installed.

Section

Understanding Host and Port Usage

© 2004, 2006 IBM Corporation

The following set of slides will consider the specification of host and port information for bootstrapping into the global name space.

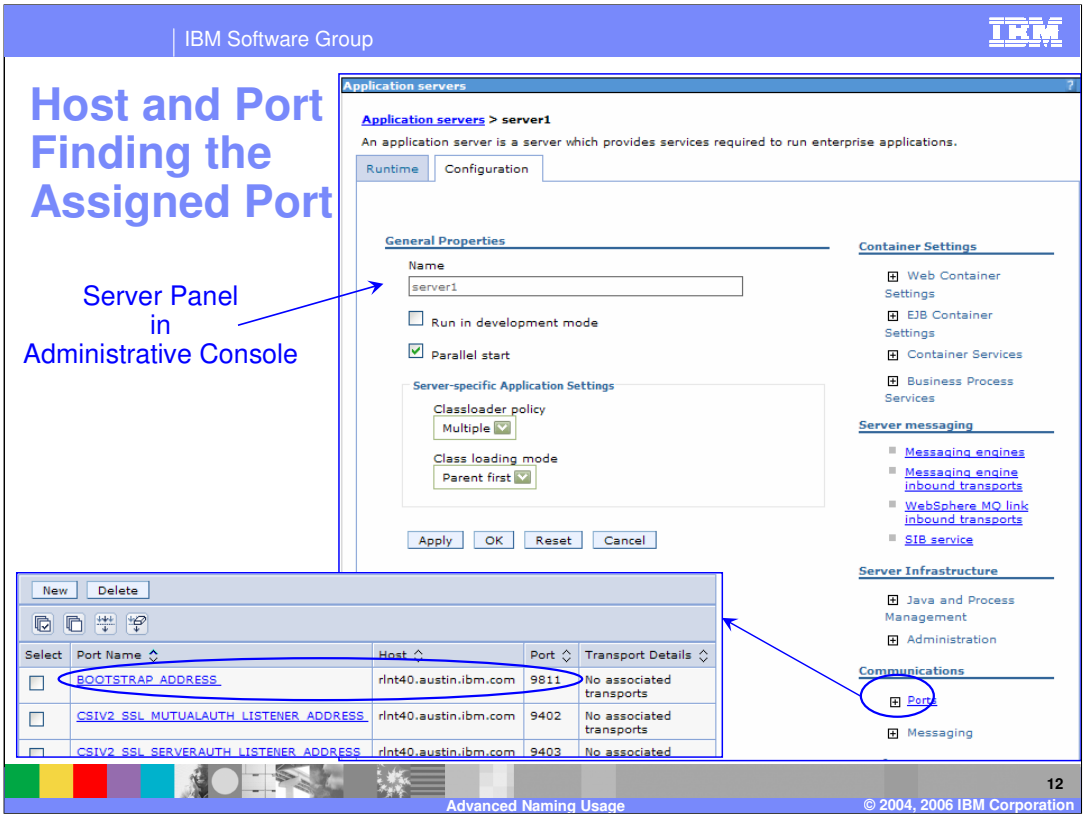
Host and Port

- When are host and port needed?
  - When outside the cell of the target EJB, such as from a:
    - J2EE Client
    - Pure Java client
    - stand-alone server
    - Server in another cell
  - Exception:
    - Not needed if server is at default host and port → “localhost:2809”

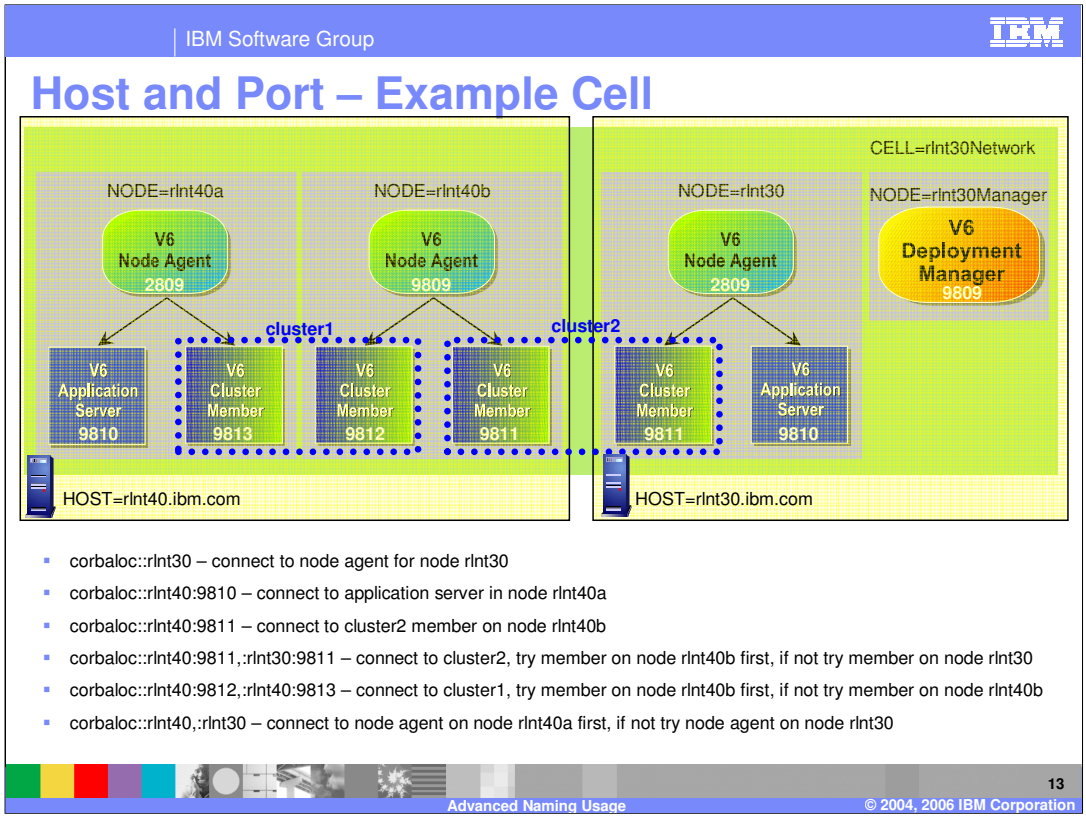
© 2004, 2006 IBM Corporation

This slide discusses when you are required to specify a host and port to connect to the global name space. If you are running any code that is outside of the target cell you need host and port information to identify where you would like to connect to the global name space. This would be true for client processes such as a J2EE client or a pure Java client. It also applies when connecting to a cell from a stand-alone Application Server or an Application Server running in a different cell.

The default host and port are localhost and 2809. When the target satisfies these, they are implicit and do not have to be explicitly specified.



The Administrative Console can be used to find the host and port used for bootstrapping to a particular server. From the server panel, opening “Posts” displays a panel which contains the “BOOTSTRAP_ADDRESS”, as is illustrated in these screen captures.



This slide shows an example configuration. The example “corbaloc:” URL strings illustrate different ways to bootstrap into the global name space for this cell.

In the example there are two host machines, “rlnt40.ibm.com” and “rlnt30.ibm.com”. There is a cell named “rlnt30Network” with nodes “rlnt30Manager” and “rlnt30” on host “rlnt30.ibm.com” and nodes “rlnt40a” and “rlnt40b” on host machine “rlnt40.ibm.com”. There are two non-clustered Application Servers in the cell and two clusters, each with two cluster members. The bootstrap ports for each of the Servers is shown with each Server.

The example URL strings are:

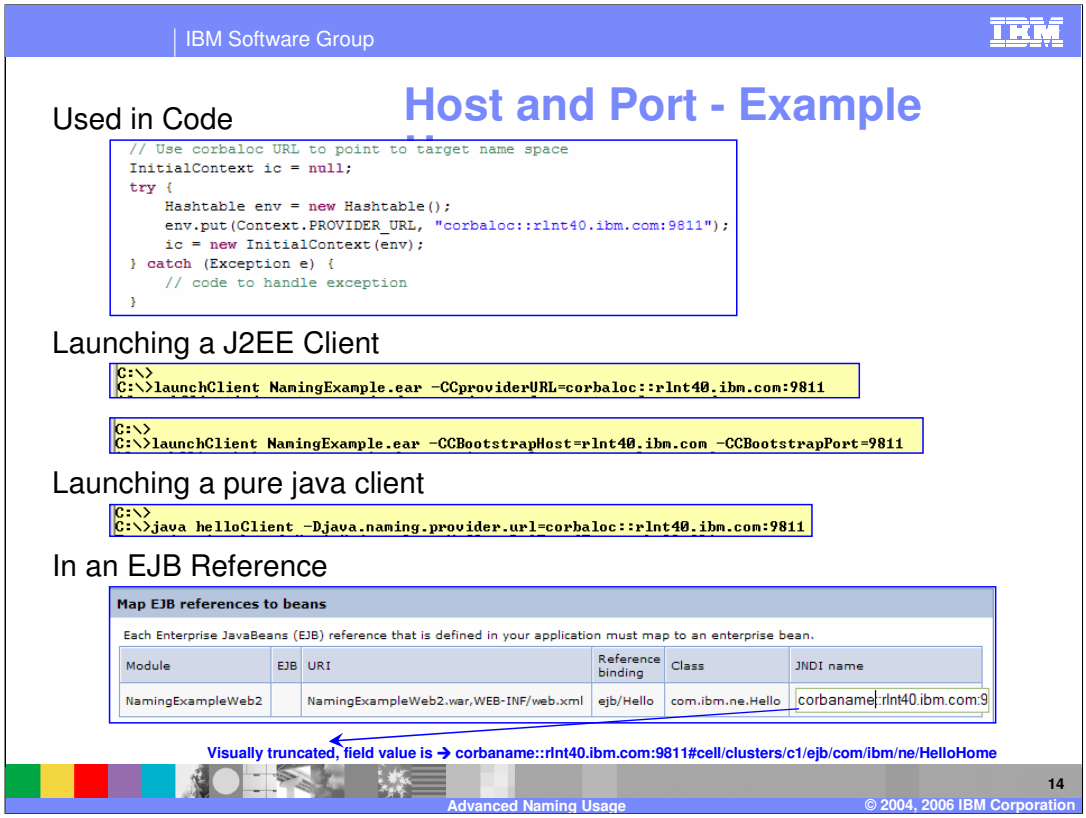
“corbaloc::rlnt30” – In this URL, the host is specified but the port is not, and therefore the default port of 2809 is implied. This identifies the node agent in node “rlnt30”.

“corbaloc::rlnt40:9810” – In this URL, both host and port are specified. This identifies the non-clustered application server on node “rlnt40a”.

“corbaloc::rlnt40:9811” – This URL also identifies both host and port. This identifies the cluster member of “cluster2” which is running on node “rlnt40b”. Note that clustering does not come into play for the bootstrapping operation, so this will only attempt to connect to that specific cluster member.

“corbaloc::rlnt40:9811,:rlnt30:9811” – This URL specifies two host and port pairs. This identifies the two cluster members of “cluster2”. An attempt will be made to bootstrap into the first one specified which is on node ”rlnt40b”, and if not successful will attempt to bootstrap into the one on node “rlnt30”.

“corbaloc::rlnt40:9812,:rlnt40:9813” – This URL also specifies two host and port pairs which identify the members of “cluster1”. An attempt will be made to bootstrap into the first



This slide illustrates the different ways that the host and port can come into play.

The first example shows a “corbaloc:” URL being used as the provider URL directly in the code when obtaining a JNDI InitialContext.

The second example shows how to pass host and port information to a J2EE client. There are two ways to do this, the first being the use of the “providerURL” parameter which takes a “corbaloc:” formatted URL string. The second way is to pass the host and port using the “BootstrapHost” and “BootstrapPort” parameters.

The third example is for a pure Java client. In this case, the environment variable “java.naming.provider.url” must be set to a “corbaloc:” formatted URL string.

Lastly, the JNDI name in an EJB Reference can be set to a “corbaloc:” formatted URL string as an alternative to using a fully-qualified name. Normally this would only be done if the target EJB was in another cell.

Section

Considerations for Deciding Where to Bootstrap into the Name Space

© 2004, 2006 IBM Corporation

There are multiple options for where to bootstrap into a WebSphere Application Server Network Deployment cell. This section takes a look at some of the things to consider when determining which is the best approach for your application and configuration.

Which Server to Connect To?

- Things to consider:
  - Coming from client process or server process in the same cell
  - Simple names versus fully-qualified names
    - Can use simple names when connecting to server containing target EJB Home
    - Illustrated in an upcoming slide
  - Knowing versus not needing to know port assignments
    - Can connect to Node Agent as normally assigned default port (2809)
  - Only one server involved when connecting to server containing target EJB Home
  - Do you want to have failover?
    - Multiple host and port pairs in corbaloc URL
    - Connecting to a cluster versus connecting to a node agent
      - Issues addressed in two upcoming slides

© 2004, 2006 IBM Corporation

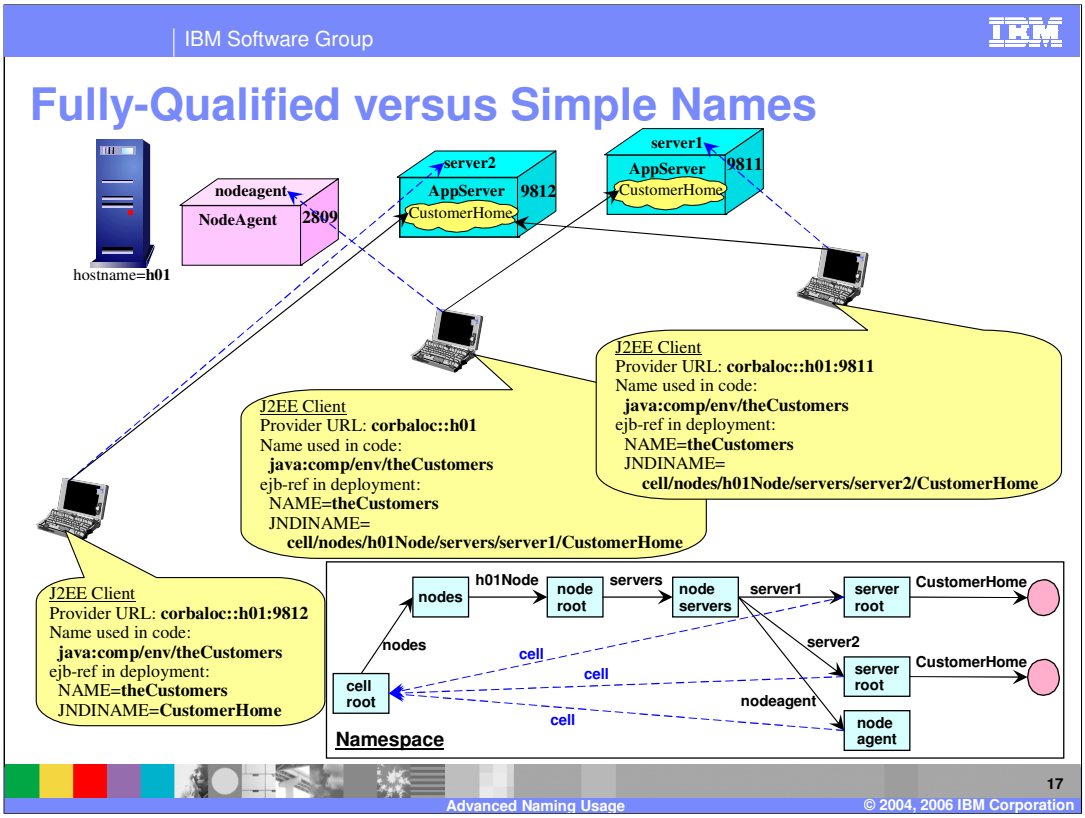
Although not an all inclusive list, these are the factors to consider when determining your strategy for bootstrapping into the global name space.

When the client code is running within an Application Server in the same cell as the target, fully-qualified names are almost always the best route to take. This bypasses the need for any host and port information and provides for failover when the target is in a cluster.

When the client code is in a client process or in a server in a different cell, the decision is more complicated. The questions basically are:

1) Would you rather deal with knowing host and port information so you can use simple names, or would fully- qualified names without having to know host and port be more appropriate?
2) Is it worth having to deal with host and port information so that you can optimize the lookup by connecting directly to the server containing the target?
3) Do you want to enable failover?

The following slides will provide information to help with understanding these decision points.

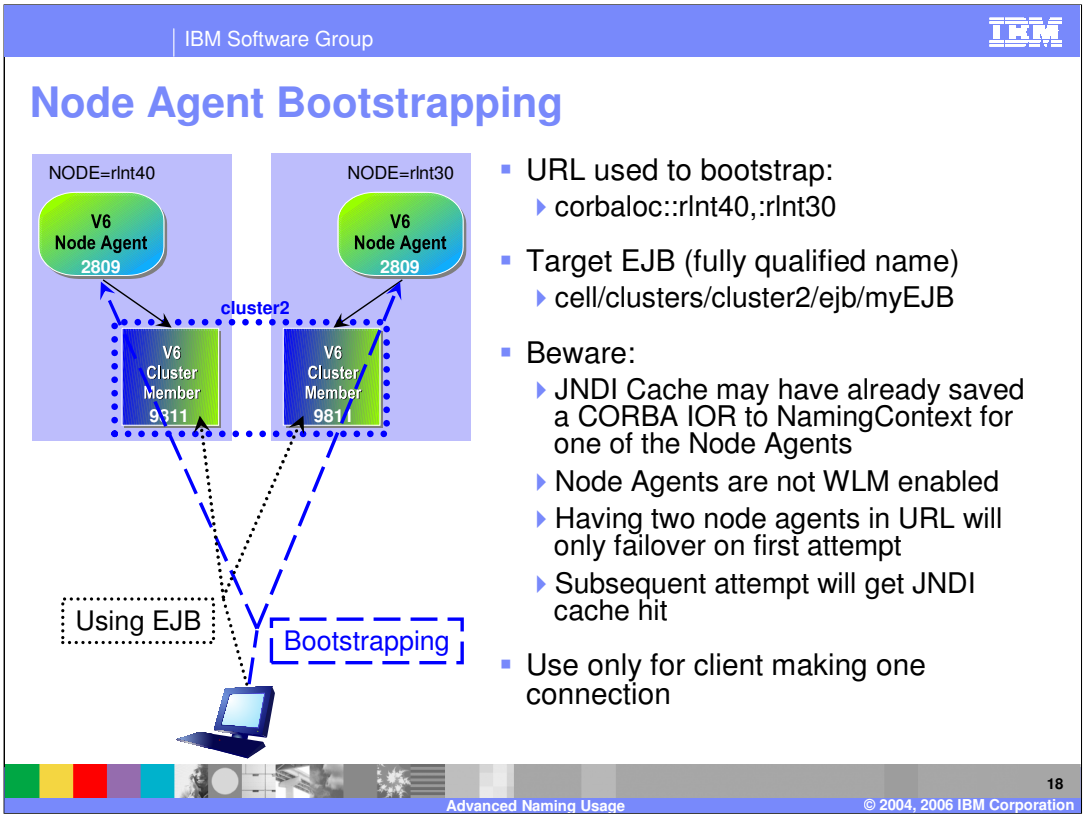


This slide is not quite as complicated as it appears at first glance. It's purpose is to illustrate different scenarios for clients and how fully-qualified names and host and port specifications come into play.

The picture illustrates a WebSphere Application Server Network Deployment node which contains a Node Agent and two Application Servers. The Node Agent is listening on port 2809, server1 on port 9811 and server2 on port 9812. Both server1 and server2 contain the Customer application with the associated CustomerHome. The picture at the lower right is an illustration of the global name space. There are three different client scenarios illustrated that show the providerURL (and therefore host and port specified), the "java:comp/env" names used in the client code, and the contents of the EJB Reference in the client deployment descriptor. You will notice that the "java:comp/env" names are all the same, so the variable factors in the scenarios are the provider URL and the JNDI Name found in the EJB Reference. The scenarios use the dashed blue arrow to indicate bootstrapping and the solid black arrow to indicate the target CustomerHome EJB accessed.

First, look at the client in the lower left side of the slide. Notice that the provider URL used both host and port so that the bootstrapping occurs directly to server2. By using the simple name CustomerHome, the client accesses the CustomerHome in server2.

Next, notice the client in the middle. There is only a host specification but no port specification which implies the use of port 2809, and the bootstrapping occurs to the node agent. Using the fully-qualified name in the EJB Reference, the CustomerHome accessed is the one from server1.

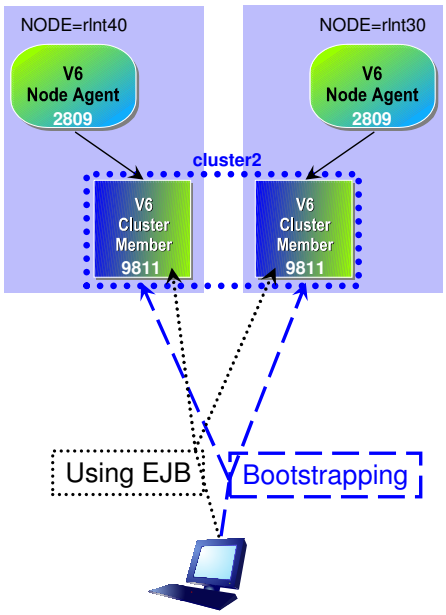


This slide illustrates achieving some level of failover by having a provider URL that points to multiple node agents. The picture shows two nodes, each with a node agent and a cluster member from “cluster2”. The URL used points to the two node agents, and the target EJB is identified using a fully-qualified name. With this configuration, even if one of the node agents is down, the bootstrapping will attempt to contact the other node agent and the look up of the EJB Home can proceed. When using this approach to obtain bootstrapping failover, you need to be aware of the following limitations:

1) On the client side, there is a JNDI cache which will cache information about the global name space.
2) Node agents themselves are not Workload Management enabled, and therefore do not display any clustering behavior
3) By having the two node agents in the URL, both will be attempted only if there is no JNDI cache hit
4) Subsequent lookups will most likely get a JNDI cache hit which will cause the bootstrapping to occur to the same node agent contacted previously. If that node agent has failed but the other is still active, there will be no failover of the bootstrapping operation.

As a guideline, bootstrapping into multiple node agents to get failover behavior should only be done when making a single connection from the client to the global name space.

Cluster Member Bootstrapping



- URL used to bootstrap:
  - corbaloc::rlnt40:9811,:rlnt30:9811
- Target EJB (simple name)
  - ejb/myEJB
- Safest for failover:
  - CORBA IOR to NamingContext is WLM enabled
  - Having two cluster members in URL will failover on first attempt
  - WLM enabled IOR in JNDI cache allows failover on subsequent attempts

© 2004, 2006 IBM Corporation

This slide illustrates achieving a high level of failover by having a provider URL that points to multiple cluster members. This is the same configuration as the previous slide. The target EJB can be looked up using simple names because the bootstrapping occurs directly into the cluster member. The bootstrapping operation itself will failover if the first cluster member is not running. In addition, the JNDI cache will contain a global name space reference which is Workload Management enabled, so on subsequent attempts to access the global name space there will be failover and load balancing between the two cluster members.

This is the recommended scenario to use for any client that is long running and making multiple accesses to the global name space.

Section

Topology Dependent Names Potential Problem and Proposed Solution

© 2004, 2006 IBM Corporation

It may be clear to you by now that having to deal with fully-qualified names that contain node, server, or cluster information and having to deal with specific host-port pairs has the potential to lead to difficulties in managing an environment. This section gets more specific about this issue and looks at a best practice to address this.

Topology Dependent Names – Define the Problem

- A scenario illustrating the problem
  - "VeryPopularEJB" is installed in "cluster1"
  - There are numerous other applications using this EJB
  - Every using application is installed with an EJB Reference that has a fully-qualified name
    - "cell/clusters/cluster1/ejb/VeryPopularEJBHome"
  - There is a need to move this EJB to "cluster2"
  - Every using application must be updated
    - Change name to → "cell/clusters/cluster2/ejb/VeryPopularEJBHome"
    - Must be able to identify all using applications
    - Must be done simultaneously with VeryPopularEJB being moved
- Depending upon your application deployment model
  - Topology dependent names could cause serious issues
  - It is better addressed up front before facing a crisis

© 2004, 2006 IBM Corporation

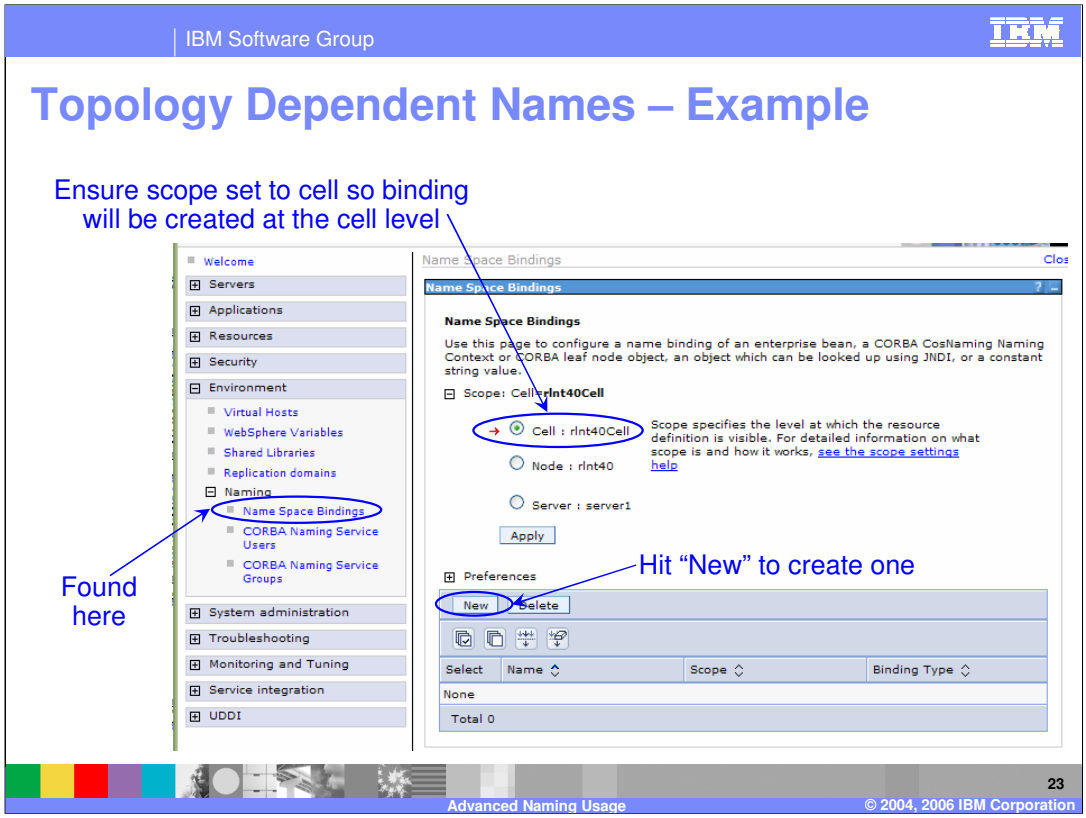
Here is an outline of a specific scenario which illustrates the kind of problems using a topology based fully-qualified name can create. To make the scenario concrete, assume there is a "Very Popular EJB" which has been installed into "cluster1". Because this EJB is very popular, there are many other applications which use it, and each have EJB References containing its fully-qualified name of "cell/clusters/cluster1/ejb/VeryPopularEJBHome". Everything is fine until it is decided that it would be better for this very popular EJB to be installed into "cluster2" which would change its fully-qualified name to "cell/clusters/cluster2/ejb/VeryPopularEJBHome". Now you must be able to identify every single application using this EJB, and modify the EJB Reference in every one simultaneous with moving the EJB. This could be a difficult, and in some cases, impossible task to accomplish.

Topology Dependent Names – A Solution

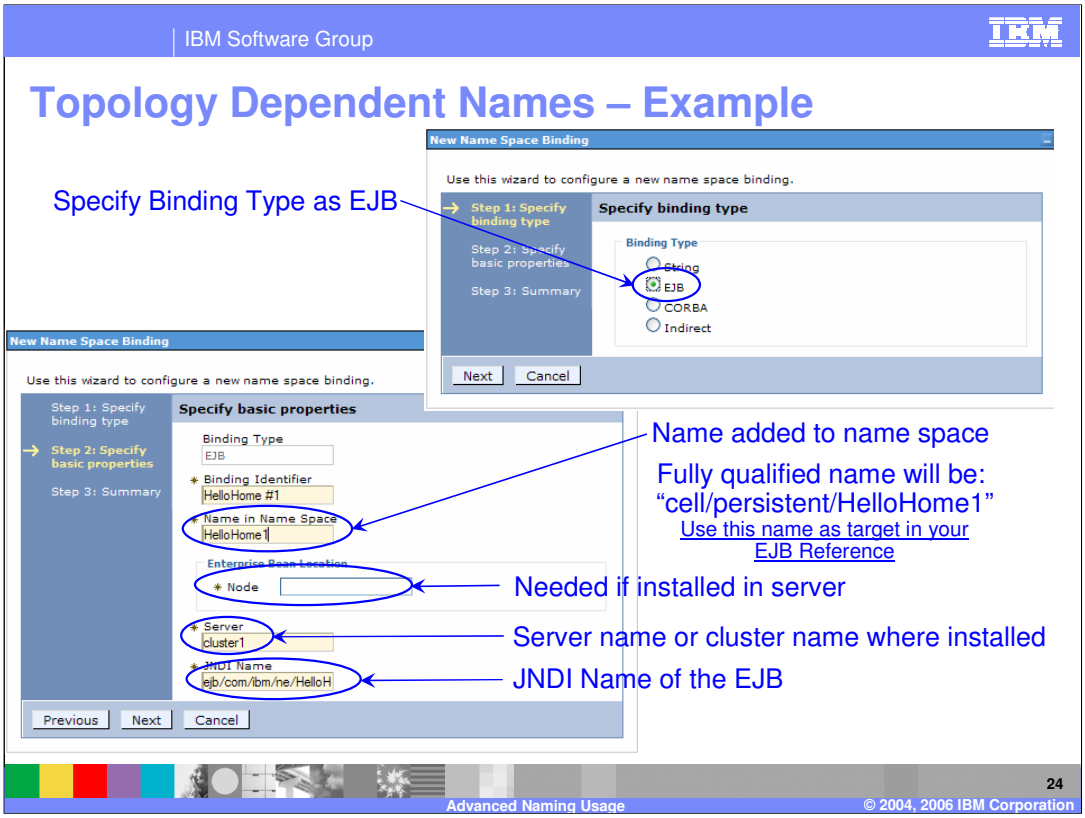
- Problem:
  - java:comp/env/<ejbname> lookups
    - Provide a level of indirection
    - Protects code from having to change
  - Topology dependent names
    - Are embedded in installed application EJB References
    - Require the application to change when configuration changes
- Solution:
  - Add another level of indirection
  - Remove topology dependent names from the application
- How? → Name Space Bindings
  - Administratively defined bindings can isolate topology dependence
  - Create an associated Name Space Binding for each EJB
  - EJB References point to the Name Space Binding
  - When the EJB needs to move, only the associated Name Space Binding needs to change

© 2004, 2006 IBM Corporation

Although the “java:comp/env” names provide a level of indirection that protects the source code from having to change when the target changes location, the name embedded in the EJB Reference may still have to change when the configuration changes. To address this issue, another level of indirection is needed to protect the JNDI name specified in the EJB Reference from having to change. This can be done using the Name Space Bindings which were discussed in the Naming Introduction presentation. Basically, by using an EJB Name Space Binding for each EJB, it can provide the level of indirection needed and hide the topology dependent information so the EJB References do not need to contain topology dependent names. The next few slides illustrate this.



This is a screen capture of the Administrative Console. You will see that under "Environment" and "Naming" there is a Name Space Bindings panel. To make use of this you need to make sure you have the scope set to "Cell" so that there is no topology information needed to traverse to this binding. Then select "New".



You are then presented with a panel that asks what kind of Name Space Binding you want to configure, which would be "EJB" in this case. You are then presented with a panel in which you can define the EJB Name Space Binding. The "Name in Name Space" value is the name by which this binding will be looked up, in this example it is set to "HelloHome1". Since the scope is set to cell, this will create a binding whose fully-qualified name will be "cell/persistent/HelloHome1". Next, specify the node and server name or cluster name on which the target EJB is actually installed, in this example being "cluster1". There is also a need to specify the JNDI name by which the target EJB is bound into the global name space, in this example "ejb/com/ibm/ne/HelloHome".

Having defined this EJB Name Space Binding, you can now use the name "cell/persistent/HelloHome1" rather than having to use "cell/clusters/cluster1/ejb/com/ibm/ne/HelloHome". This removes the topology information from the EJB References. When moving the Hello EJB from cluster1 to cluster2, all that is needed is to update the EJB Name Space Binding at the same time the application is uninstalled from cluster1 and installed on cluster2. All the EJB References in the using applications can remain unchanged.

As a best practice, it is recommended that this technique be used for most production Network Deployment environments that make use of topology dependent names.

Summary

- Addressed Naming in Network Deployment Environment
- Understand characteristics of the global name space
  - Use of fully-qualified names
  - Host and port considerations
- Determine where to bootstrap
  - Examined the trade offs
- Topology Dependent Name issue
  - Define the problem
  - Define best practices to avoid the problem

© 2004, 2006 IBM Corporation

This presentation took an in-depth look at Naming in the WebSphere Network Deployment environment. This was done by looking at the global name space and learning about fully-qualified names and host and port considerations. It also looked at how to make decisions about what scenario you want to use for your application clients to connect to the name space. Lastly, this presentation looked at best practices for avoiding the use of topology dependent names and thus making the environment more manageable.

Template Revision: 11/02/2004 5:50 PM

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, 2006. 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.