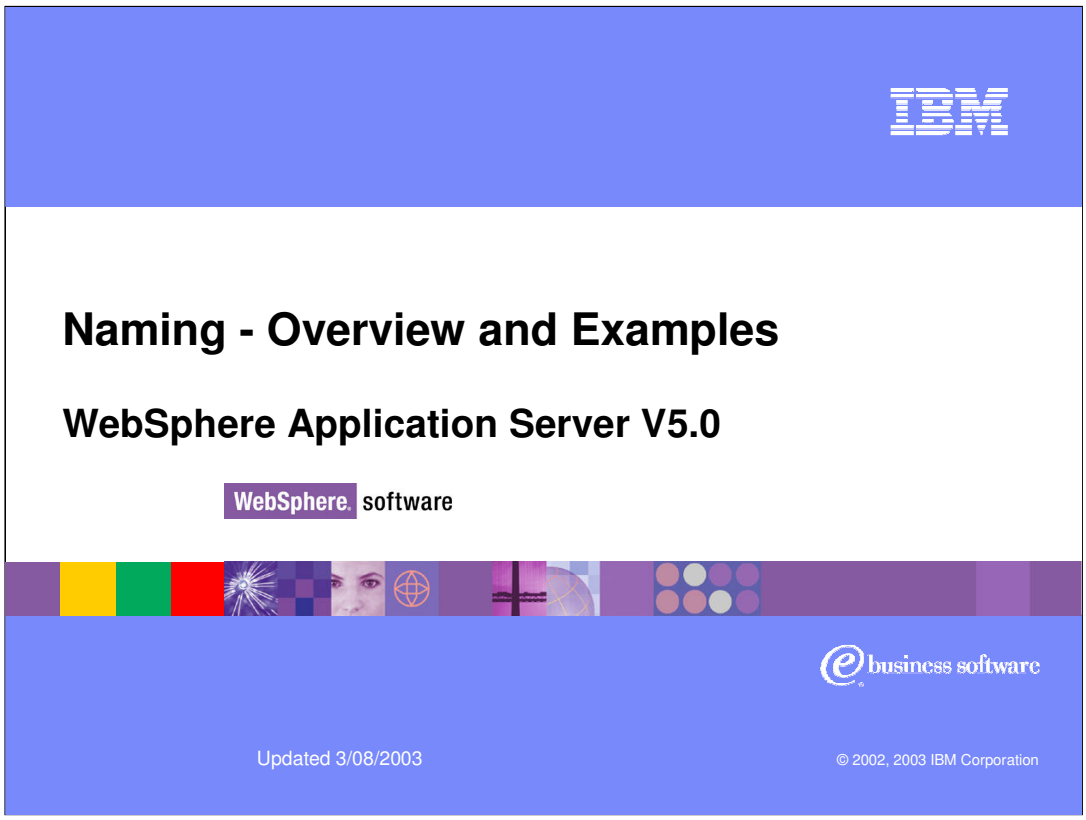
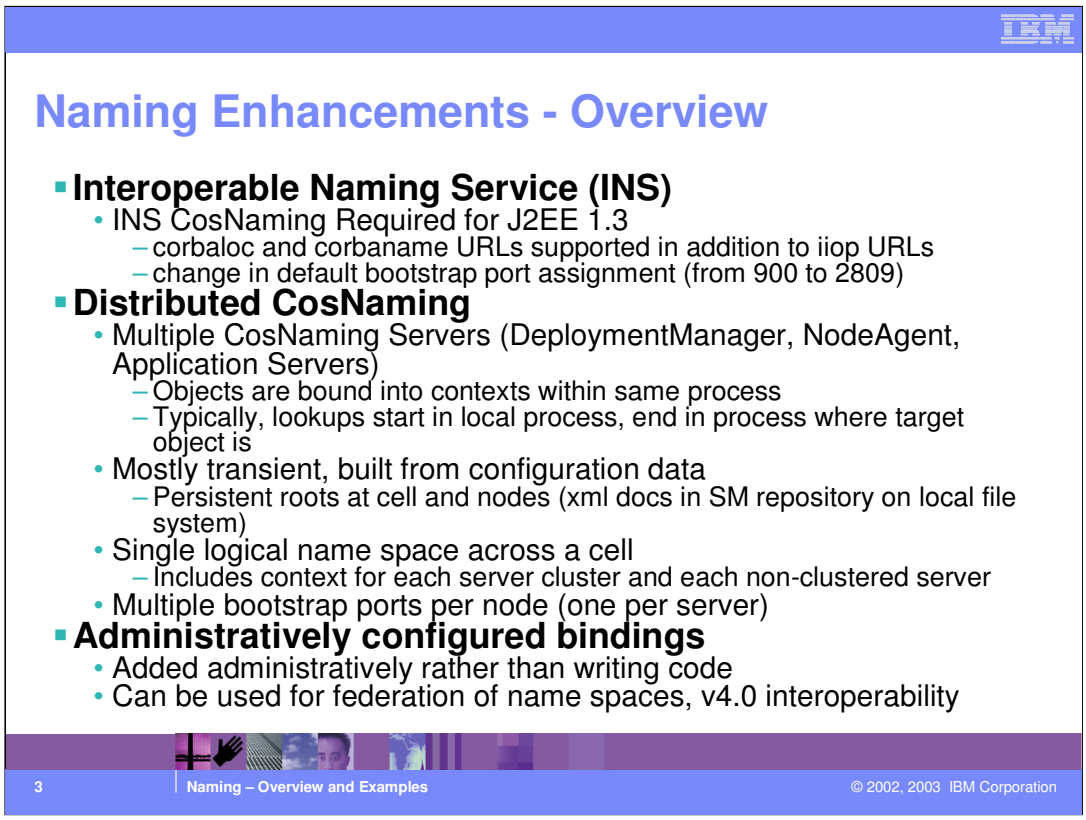

IBM
Naming - Overview and Examples
WebSphere Application Server V5.0
WebSphere. software
e business software
Updated 3/08/2003
© 2002, 2003 IBM Corporation

Agenda

- **What has changed in Naming?**
  - CosNaming functionality changes
  - Limited JNDI exposure of underlying changes
- **Interoperable Naming Service (INS)**
  - Improves interoperability with other App Servers and with CORBA servers
- **Distributed CosNaming**
  - Removes single name server bottleneck
- **Administratively configured bindings**
  - Increased ability to add objects to name space through admin interfaces



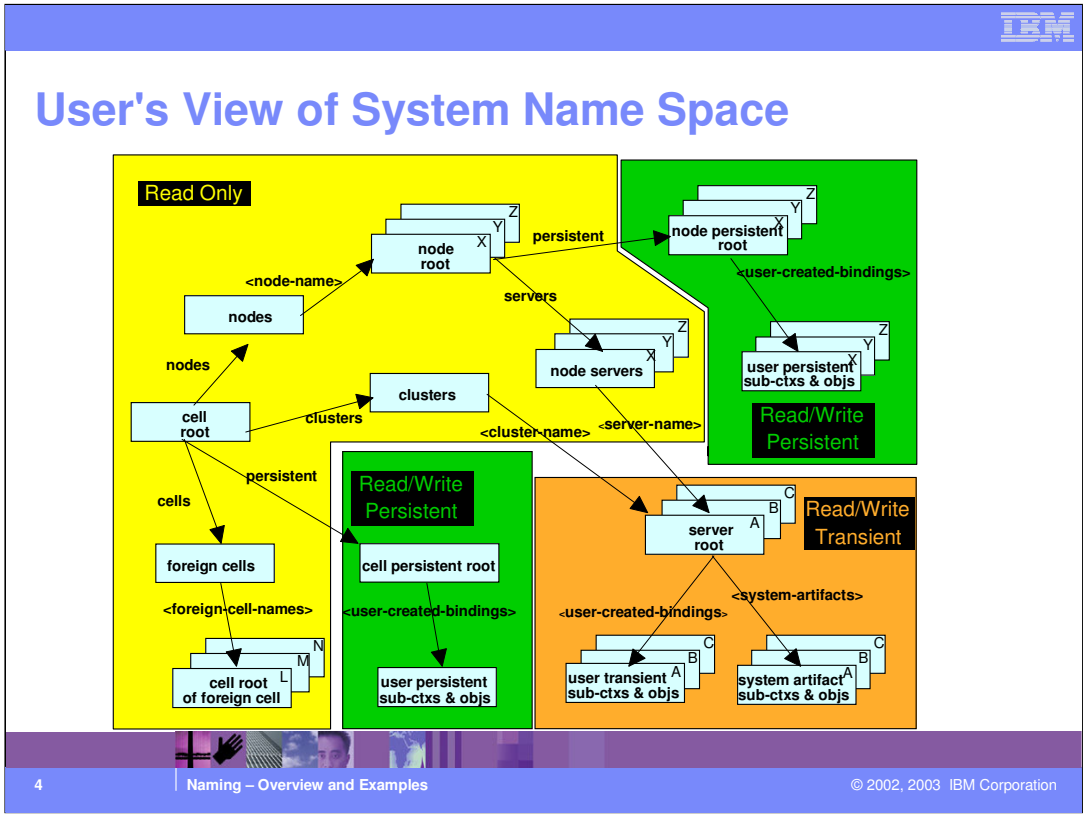
Servers to deal and understand names formulated according to the CORBA naming scheme (corbaloc designates an end-point, such as a host and corbaname designates an object's name) Notice that the port name has also changed (the standard requires it to be 2809).

The naming architecture uses the CORBA CosNaming as a foundation. In WebSphere 5.0 we have multiple CosNaming servers and this reduces the dependency of the WebSphere Application Server network on a single name server.

Objects are bound locally - a certain server is "responsible" for managing the names of the objects that are bound locally.

There is still a concept of single logical name space, accessible at the cell manager level. You will be able to navigate to specific sub contexts - each server or cluster of servers having its own context.

An interesting enhancement is the concept of "administratively configured bindings" In a nutshell, these offer the ability to configure an "alias" for names through the admin console. That way, you can create an additional level of indirection around your V5 names. That can be very useful in order to make v4 names compatible with the V5 naming structure.

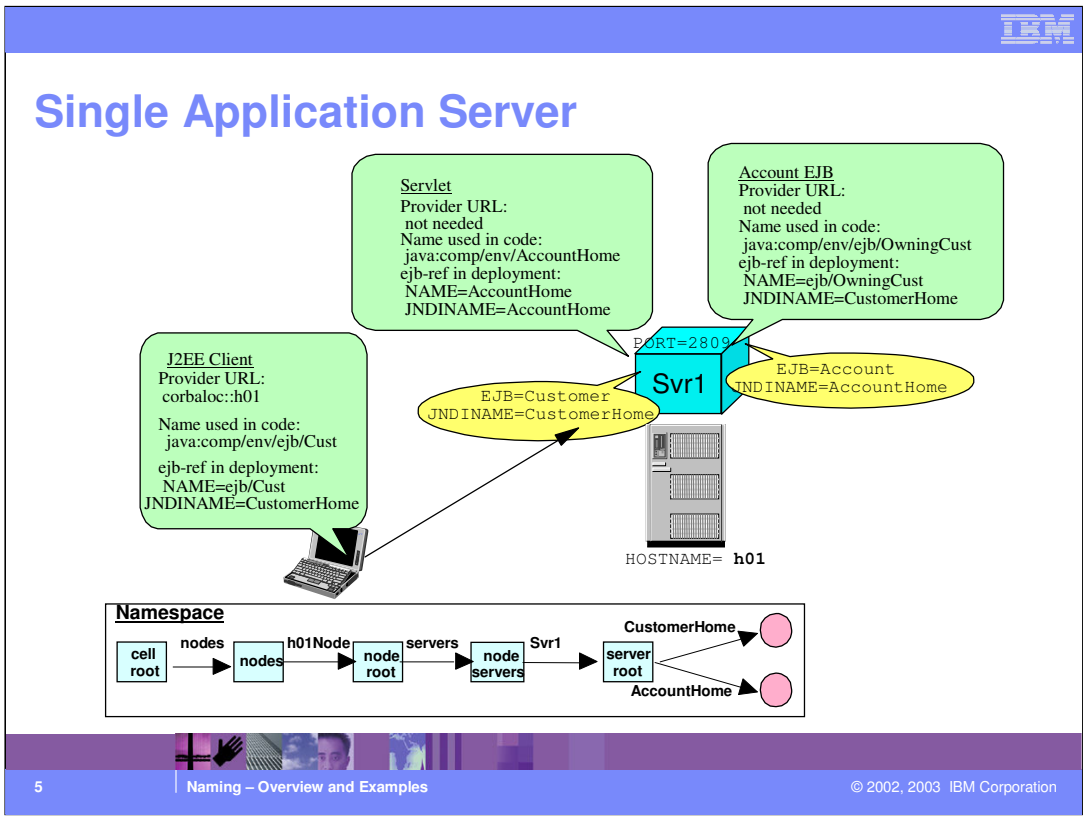


This chart gives you a map of the naming architecture in WebSphere Application Server 5.0

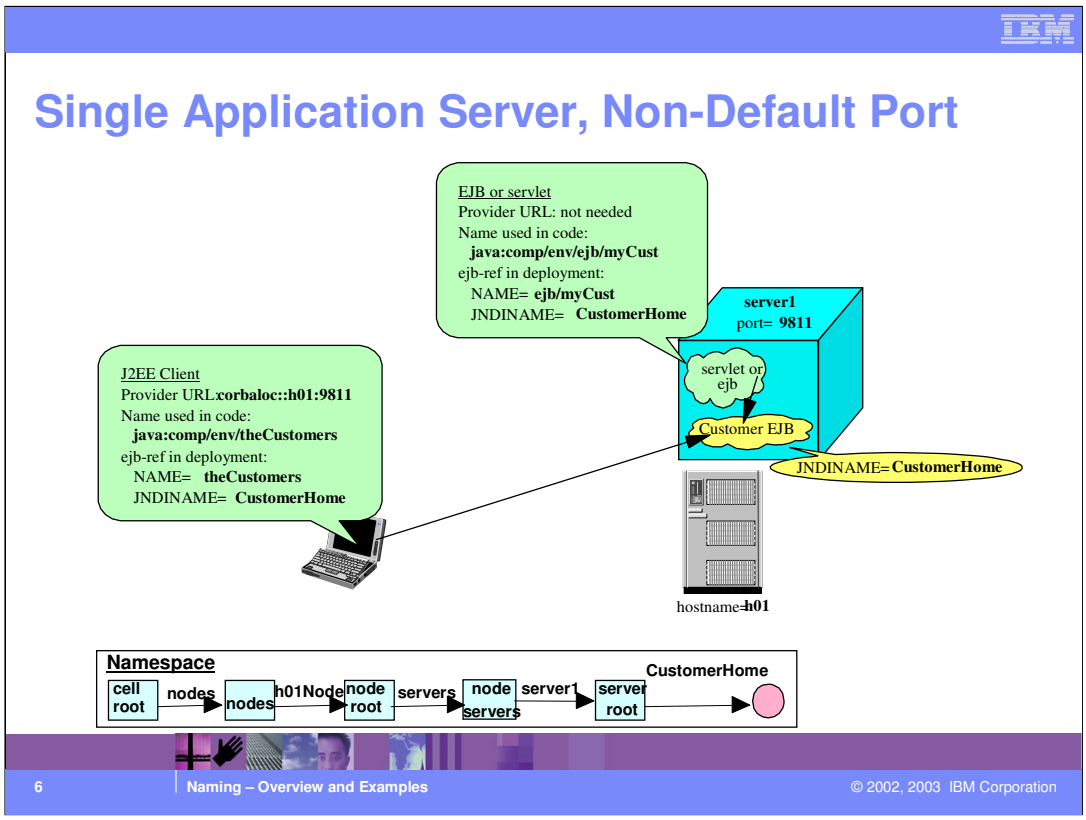
The yellow portion of the naming is basically a reflection of the topology - it's therefore read-only (can't be modified programmatically) and it is persistence - its persistent state resides in the xml repository that contains the topological information.

The green areas are meant to store configuration information primarily about resources, such as data sources, JMS destinations, etc. This data can be modified using the JNDI APIs (the admin interfaces will do that for us) and it is persistent - being stored in some other group of xml files out of the repository.

The remaining area is also updateable through APIs and it is meant to contain information such as the EJB bindings and JNDI names and so on. This part of the name space is transient, in that changes made through the JNDI APIs will not be persisted anywhere. In practice, this area of the name space gets created at server's startup - for instance, as the EJB deployment descriptors are processed.



In the single server environment, the naming works exactly the same way as it used to do in Version 4. There is only one root context - no ambiguity on the location of the named object - since there is only one server.

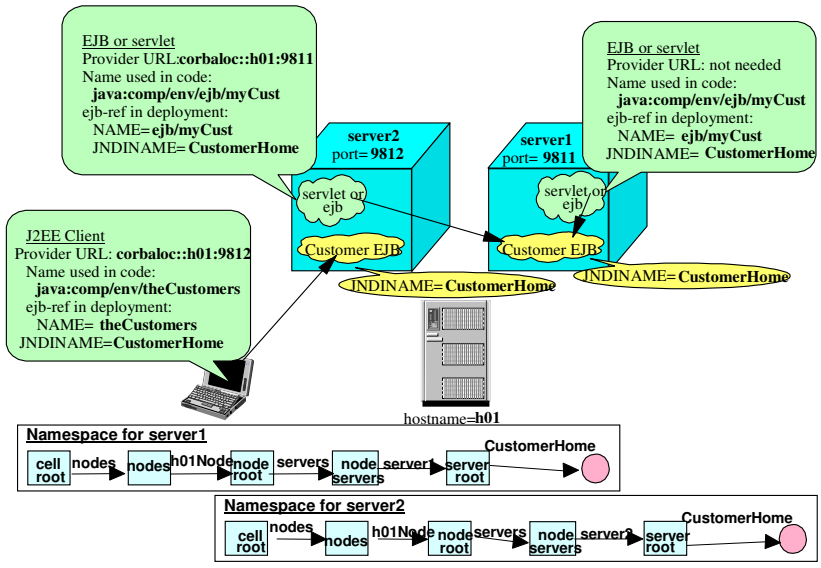


You may still configure the single server to use a different naming port - for instance if you need to have multiple instances of the server running on the same box

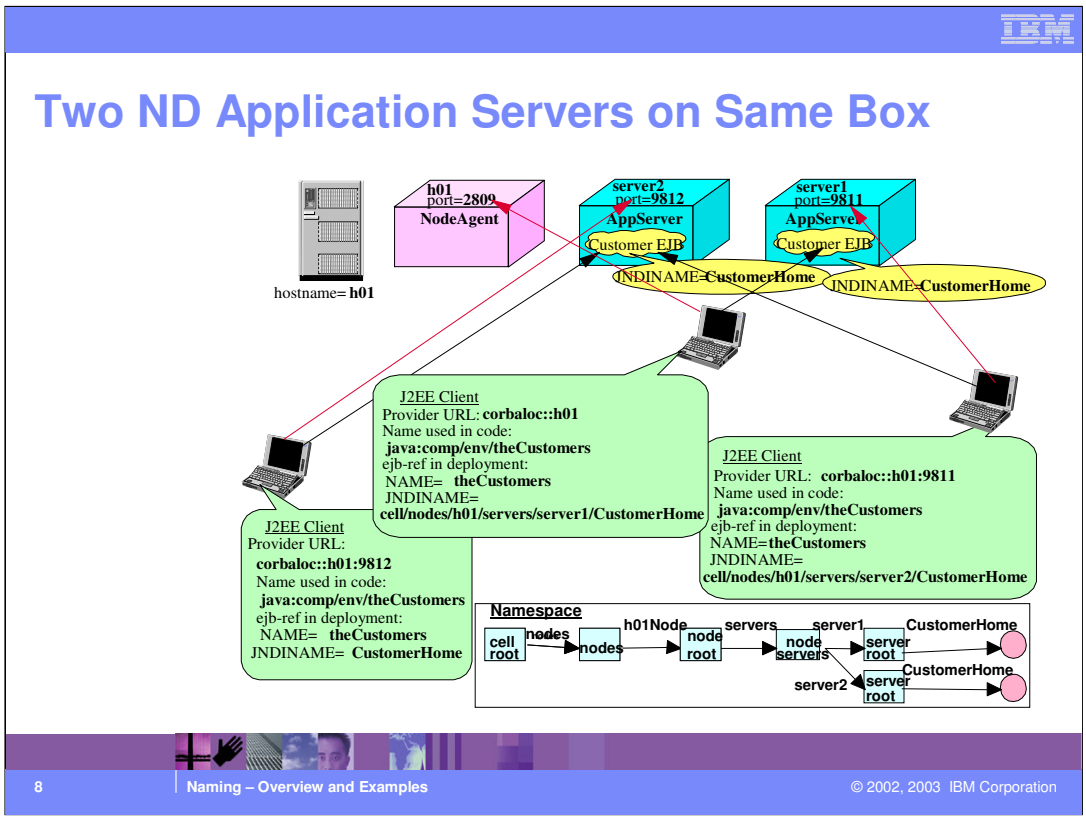
The chart illustrates how to get to that particular server. The only component really impacted is the J2EE client. Servlets and EJBs will by default look objects up in the same server as they are running - unless you specifically instruct them to look elsewhere.

Notice the provider URL expressed by means of a corbaloc name.

Two Single Application Servers on Same Box



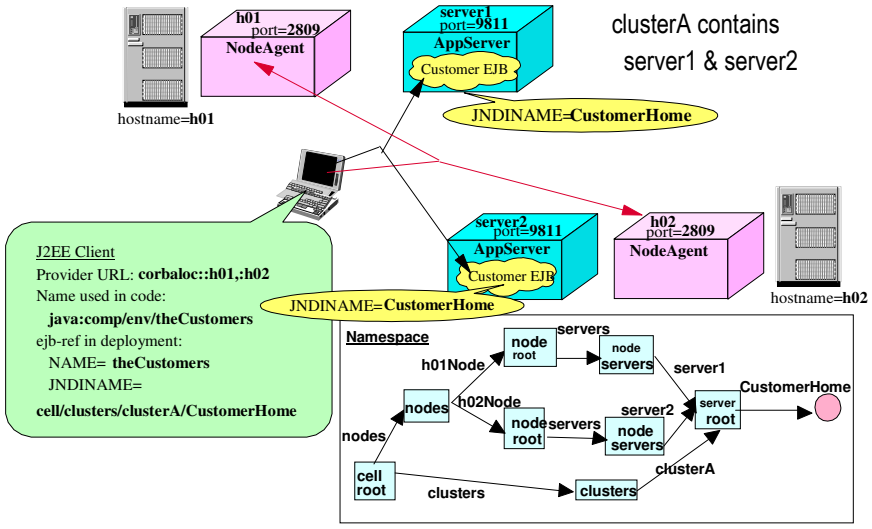
As we mentioned in the previous chart - you may have multiple instances of the single server coexisting and interoperating on the same box - in this case you must configure them with different bootstrap ports. The chart illustrates several combinations and lookup possibilities - including a J2EE component running on one instance of the single server looking up objects on the other instance.



Things become more complex as we move to network deployment. Here every server has its own sub context (every app server is a Provider URL) - including the node agent.

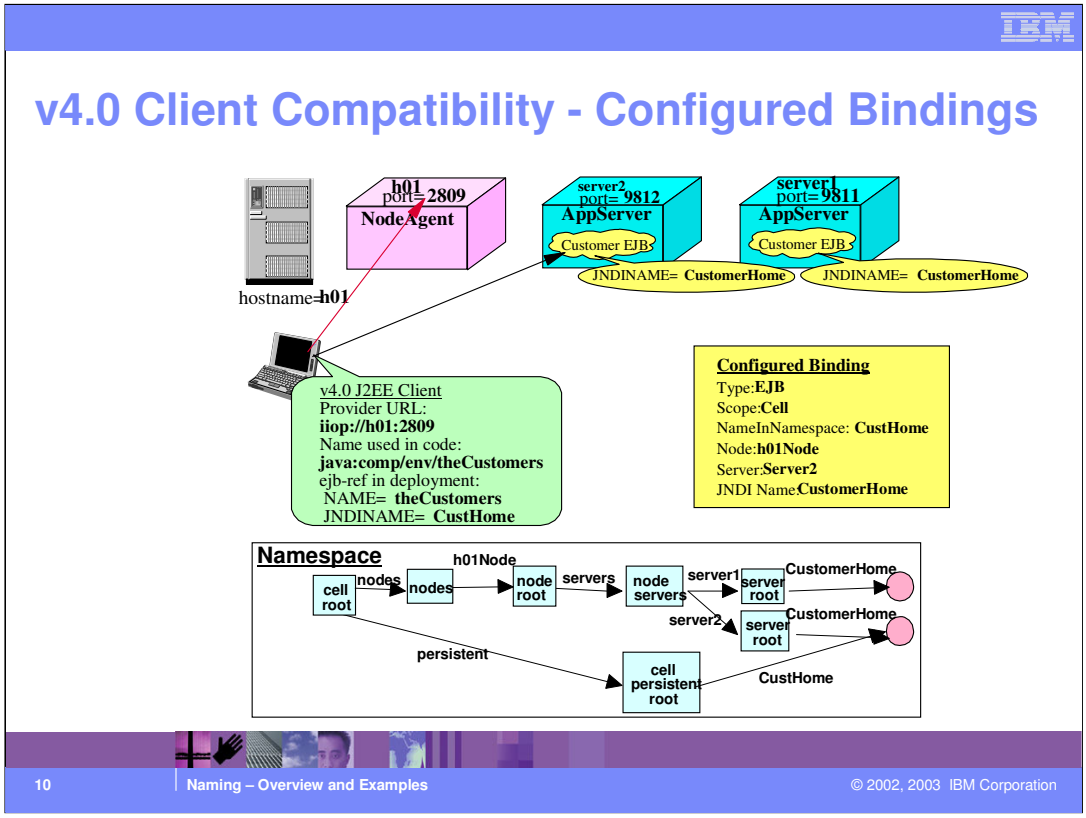
The node agent is going to be the default provider URL for a J2EE client running on that node. Therefore, unless a client specifies a certain appserver as a provider URL, the lookup will occur on the node agent. In order for the lookup to succeed, the bindings have to specify the correct fully qualified name of the object (cell/servers/<server>/<name of the obj>) - in other words, the client needs to specify WHERE the objects is located (big difference with the 4.0 situation).

Cluster of Two Application Servers



Things become more complex as we move to network deployment. Here every server has its own sub context (every app server is a Provider URL) - including the node agent.

The node agent is going to be the default provider URL for a J2EE client running on that node. Therefore, unless a client specifies a certain appserver as a provider URL, the lookup will occur on the node agent. In order for the lookup to succeed, the bindings have to specify the correct fully qualified name of the object (cell/servers/<server>/<name of the obj>) - in other words, the client needs to specify WHERE the objects is located (big difference with the 4.0 situation).

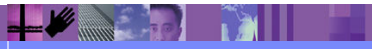


In 4.0 - there was no need to specify a path to the object - and while this feature was convenient, it also lent itself to producing naming conflicts.

However - in order to preserve the 4.0 naming structure, you may want to create configured bindings that map the 4.0 style into a fully qualified name in the 5.0 name space.

Bootstrap Ports, CORBA URLs

- **Every process has a bootstrap server/port assignment**
  - Unique port assignment per server on a given node
  - Default port assignments
    - Single server environment- default to 2809
    - Managed environment - NodeAgent defaults 2809
    - Administrators make explicit assignments for others (defaults 9810, 9811, ...)
- **JNDI and CORBA URLs**
  - used for providerurl on InitialContext - generally a corbaloc
  - used for direct URL lookup - ic.lookup("urlstring"); - generally a corbaname

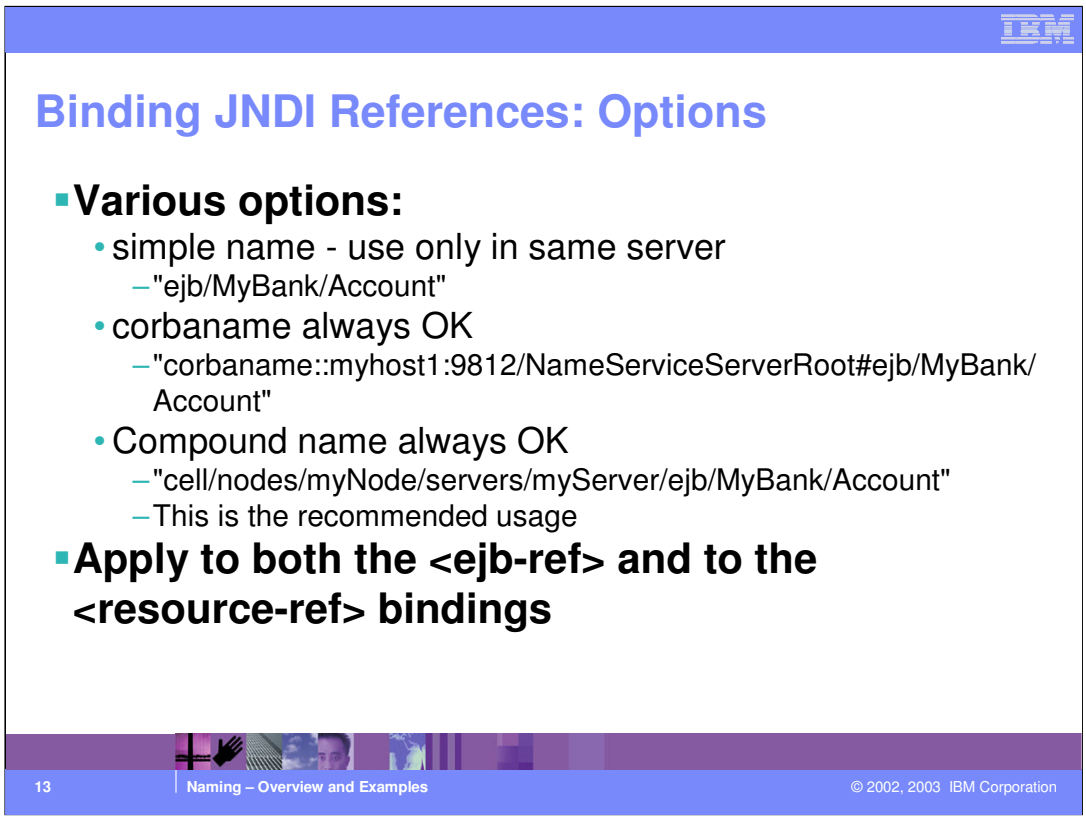


This chart summarizes the changes that involved the bootstrap port in 5.0.

CORBA URLs – Syntax and Examples

- **CORBA URL Syntax is also supported**
  - corbaloc:<protocol>:<addresslist>/<Key>
  - corbaname:<protocol>:<addresslist>/<key>#<INS-string-formated-name>
- **CORBA URL Examples**
  - corbaloc::myhost
  - corbaloc:iiop:1.2@myhost.austin.ibm.com:9344/NameServiceCellRoot
  - corbaloc::myhost1:9333,:myhost2:9333,:myhost2:9334/NameService ServerRoot
  - corbaname::myhost:9333#nodes/myNode1/servers/server5/someEjb
  - corbaname::myhost1:9333,:myhost2:9333#clusters/myCluster2/some Ejb
  - corbaname::myhost:9333/NameServiceServerRoot/someEjb

This slide provides some examples of CORBA names.



What are then the options available to the J2EE developer for object names?

The simple bindings (4.0 style) is guaranteed to succeed only in the single server case. You may use it in a servlet or EJB if you are sure the object you are looking up is going to be located on the same app server.

Alternatively the corbaname is always going to succeed. Of course you need to know the correct path to the object.

Similarly - the fully qualified JNDI name is also always successful.

Binding JNDI References: Options

- **Controlled by property com.ibm.websphere.naming.namespaceroot**
  - cellroot
  - cellpersistentroot (also legacydomainroot for v4.0 compatibility)
  - bootstrapnoderoot (also bootstraphostroot for v4.0 compatibility)
  - treeinfrastructureroot (naming implementation internal usage only)
  - **bootstrapserverroot** (this is the default)
- **From default position of bootstrapserverroot:**
  - from within a server, EJBs / resources can be treated as simple names
  - bootstrapping to specific server or cluster, EJBs/resources can be treated as simple names

If you do not specify a fully qualified name or a corbaname, a property allows you to control which root context is going to be assumed. The default is that the name is going to be resolved based on the context associated with the server whose bootstrap port you are connected to.

Configured Bindings

- **Administrators can configure bindings to exist in name space**
- **Types of objects which can be bound:**
  - EJB in a server in the same cell
  - CORBA Object that can be identified with a corbaname URL
    - must be bound into some INS compliant CosNaming server
  - Any object bound in WebSphere name space accessible with JNDI
    - uses IndirectJndiLookup object
  - String constant
- **Configured bindings can be relative to:**
  - server root
  - node persistent root
  - cell persistent root

There are four types of configured bindings

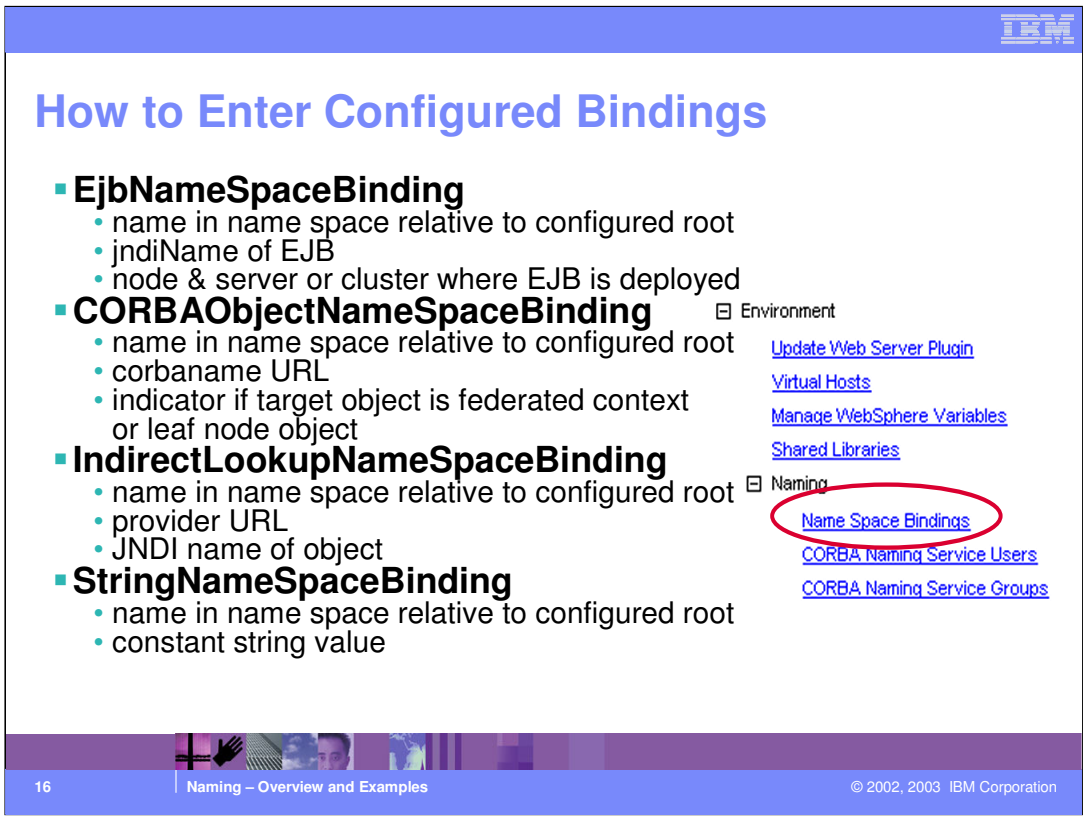
EJB located in an app server within the same cell

CORBA object identified by a corbaname

Another JNDI name

A string

The configured binding allows for yet another level of indirection when it comes to name resolution - they can be very effective when migrating 4.0 clients to the V5 platform.



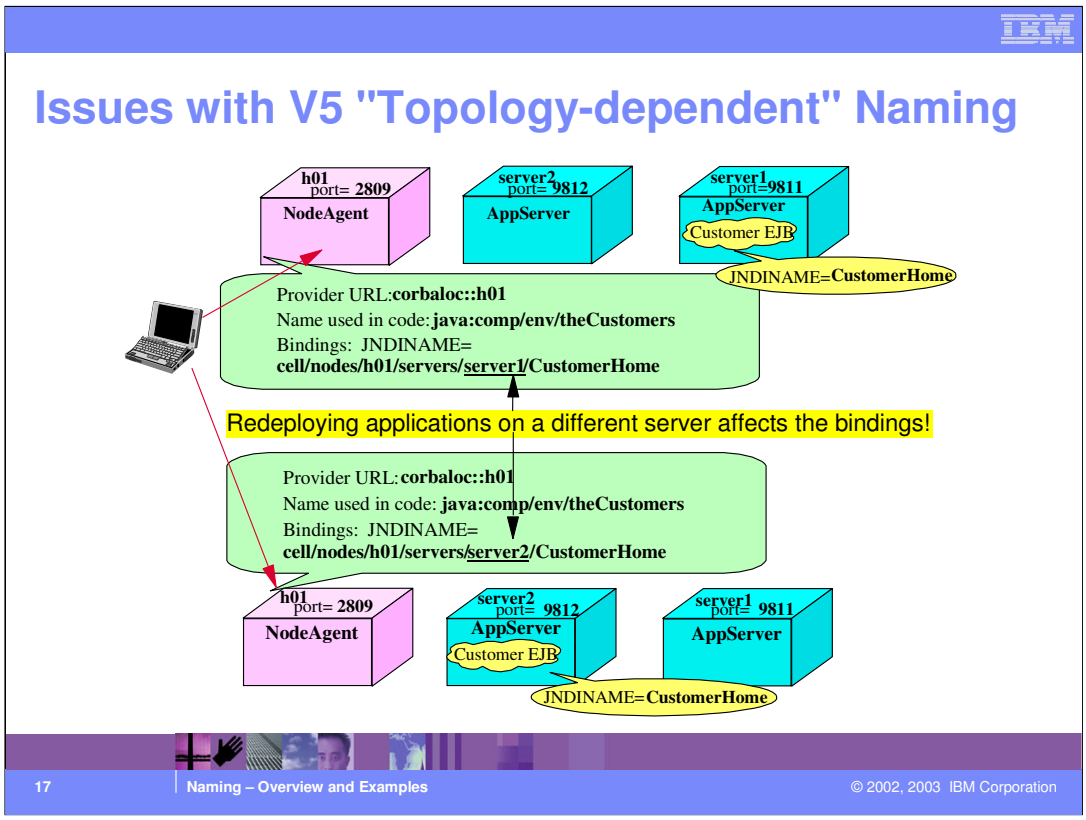
The information required to create a configured binding in WebSphere 5.0 varies based on the type of the binding itself.

For EJB Name Space Bindings, you need to specify the name of the EJB relative to the configured root sub context. This will be the name used by the programs to look up the EJB. You then need to specify the real JNDI name of the EJB and the server or cluster where the EJB is installed. By creating an EJB configured binding, you allow applications to look up EJBs using a name that is not dependent on the topology of the WebSphere cell. We will talk about the importance of EJB configured bindings later on in this presentation.

If you want to configure a CORBA Object Name Space binding, you need to specify its name relative to the configured root context. You then need to specify the corbaname URL that indicates the actual location of the name. Last - you need to specify an indicator that tells whether the binding refers to a sub context or to an actual leaf object.

For the indirect lookup name space binding, you need to specify its name relative to the root sub context. Next you need to enter the provider URL and the JNDI name by which that object is known at the provider URL.

Finally, for a String name space binding, you need to specify its name and the correspondent string value.



The new structure of the name space allows for increased flexibility. However - if you bind your EJB references directly to the compound names of resources and EJBs, you expose client applications to potential maintenance issues.

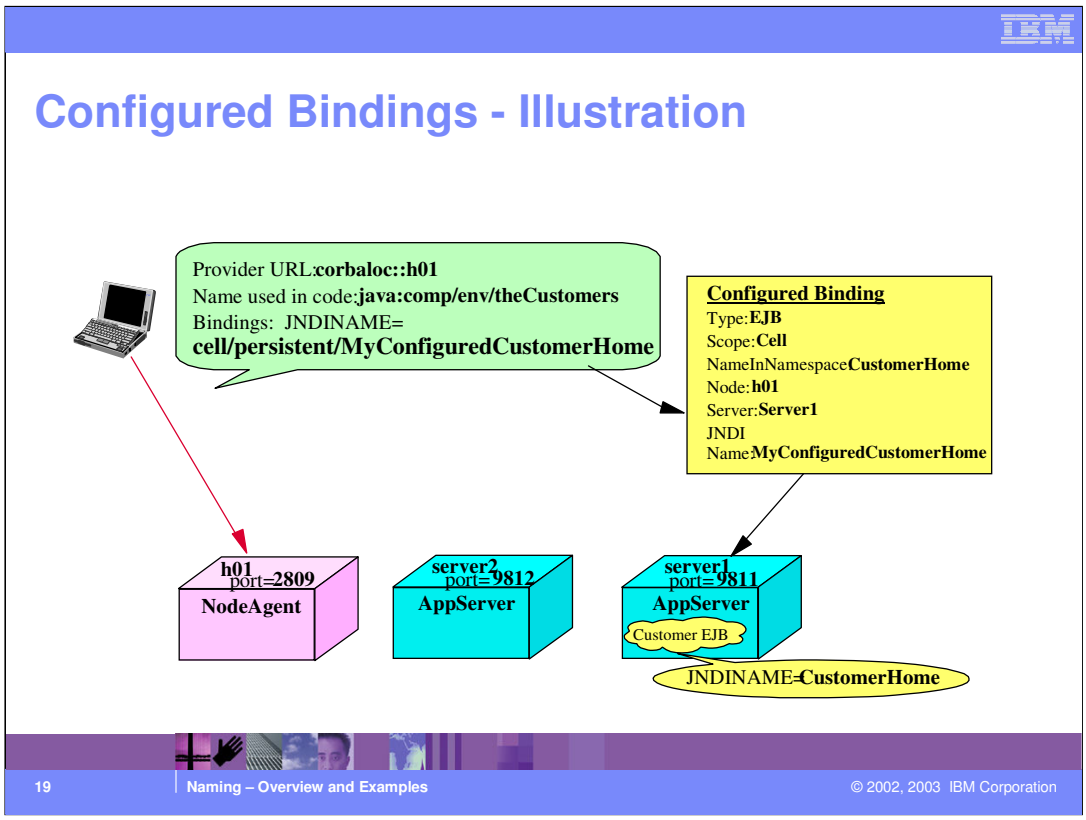
In this chart, we illustrate an application where a client is using the Customer EJB. Initially, the Customer EJB was installed on server1. The client exposes an EJB reference that is bound using the Version 5 compound name for that EJB. As we have discussed, the compound name is "topology-dependent", because it includes the name of the node and the name of the server where the EJB is installed.

This implies that if the EJB is "moved" from server1 to server2 (for instance, as part of reorganizing an existing application environment) we have to update the bindings of the J2EE client to have them reflect the new application topology. In large multi-tiered applications, this may be a significant maintenance issue.

Reducing Topology Dependence

- **Create configured bindings for your EJBs**
  - Configured bindings point to the topology-dependent name
- **Specify names of configured bindings to bind clients' EJB references**
  - If EJB is moved to a different server, you only need to update the configured binding, rather than EJB references in all the EJB clients

The impact of topology-dependent names on application maintenance can be strongly reduced by using configured bindings.



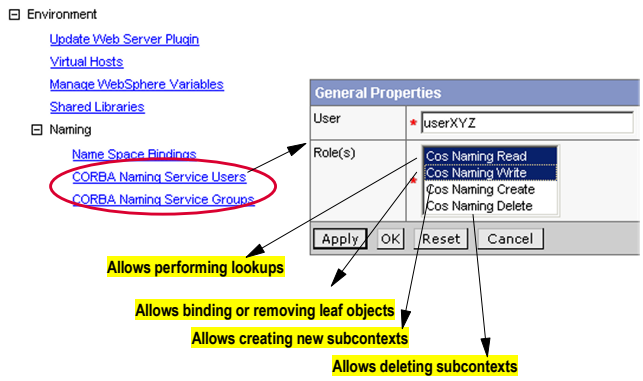
This chart illustrates how you could make the application client described in the previous chart completely resilient to changes in the application configuration.

The client's EJB reference is bound to a configured binding (cell/persistent/MyConfiguredCustomerHome) instead of being bound to the topology-based name.

The configured binding, in turn, points to the topology dependent name. This solution makes it a lot easier to redeploy the Customer EJB on a different server or on a cluster: the only place that needs updating is the configured binding - the client wouldn't have to change.

Naming Security

- **Allows you to associate user and user groups with "roles"**
- **Roles determine what a user or a group is allowed to do**



In WebSphere Version 5, you can secure naming operations.

There are four different roles that can be assigned to each user or user group - the same user can be assigned multiple roles if necessary.

CosNamingRead is the role that enables a user to perform lookup operations on the name space

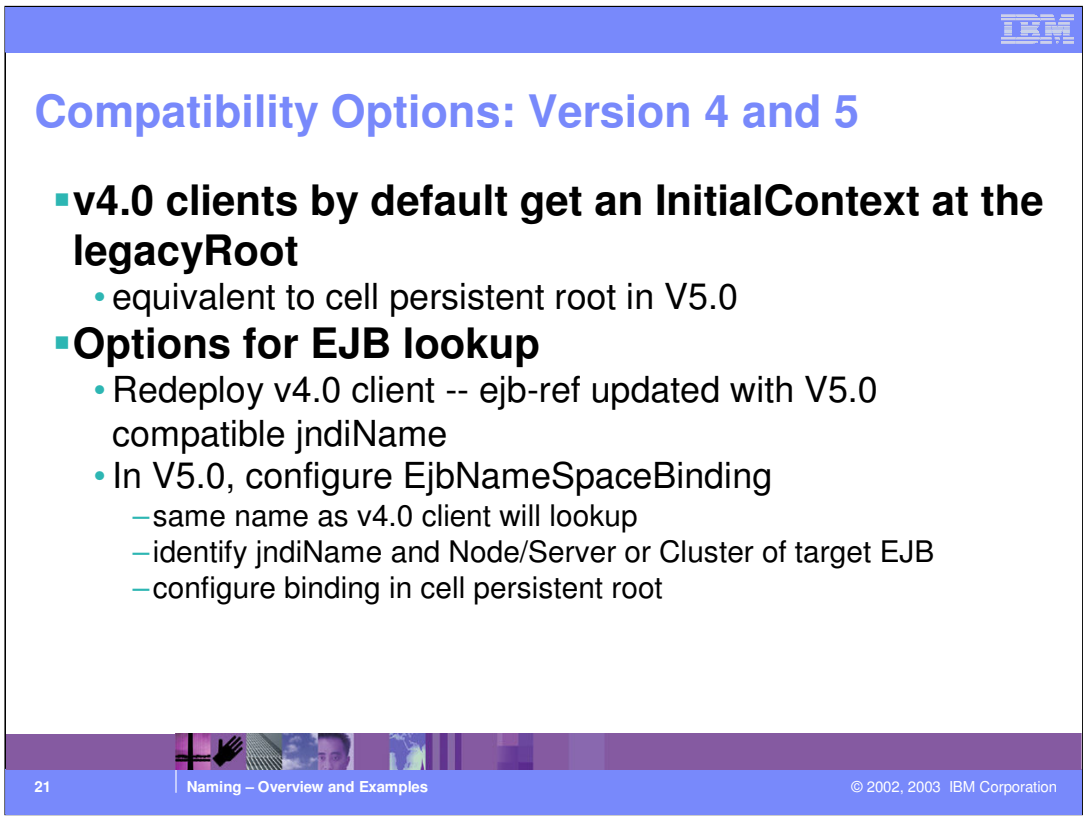
CosNamingWrite allows the users to add or remove bindings, but only for leaf objects - sub contexts cannot be manipulated.

CosNamingCreate allows the users to add sub contexts

CosNamingDelete allows the users to remove sub contexts.

Naming security was introduced back in Version 4.0.2 and it was carried over into Version 5. Currently, the level of granularity of naming security does not allow you to protect specific sub contexts or specific leaf objects - if you authorize a user to perform lookups, that user will be able to perform lookups anywhere in the name space.

By default, all users are assigned to a group called Everyone which has Cos Naming Read authority. Also - all authenticated users can perform all operations if security is on.



In WebSphere Version 5, interoperability with prior releases has been given a lot of focus.

Version 4 clients are bootstrapped at a special initial context - the "cell persistent root" sub context. That way, Version 4 clients will be presented with a monolithic namespace that resembles the V4 name space. In other words, they will not be directly exposed to the V5 topology-dependent name space.

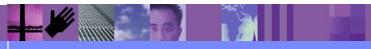
If your Version 4 client needs to look up EJBs, you could choose between two alternatives:

You could redeploy the client and bind it using the new topology-dependent names

or - you could create EJB configured bindings under the cell persistent root. The migration tooling will create those bindings during the migration process.

Compatibility Options: Version 4 and 5

- **Options for Resources bound in external name space**
  - Redeploy v4.0 client -- resource-ref updated with V5.0 compatible jndiName
  - In V5.0, run program to bind resource in V5.0 cell persistent root
  - In V5.0, configure IndirectLookupNameSpaceBinding
    - same name as v4.0 client will lookup
    - specify provider URL & jndi name of name space where resource is already bound (some v4.0 name space)
    - configure binding in cell persistent root



If your Version 4 client looks up resources that are bound to an external name space outside of WebSphere, you have three alternatives:

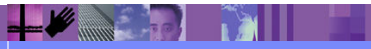
You can redeploy the client to use the Version 5 naming conventions

or, you could write and run a utility program that rebinds those external resources under the cell persistent root

or, you could create valid indirect lookup name space bindings for those resources.

dumpNameSpace - You Can't Live Without It

- **Extremely helpful for debugging any name space related problems**
- **Dump contents of name space for a single server**
  - does not present the full logical view of the name space
  - shows CORBA URLs where name space transitions to another server
- **Need to use correct host/port to dump the server you are interested in**

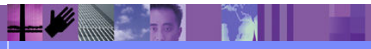


The dumpNameSpace utility is also updated. You can run it against any bootstrap port and get a listing of the names bound with that Provider URL.

The dumpNameSpace will indicate that a certain name points to a context which is external to the current server - showing the transitions that are necessary to perform a look up. In other words you won't be able to have a dump of the entire name space with one single command.

dumpNameSpace - Examples

- **Get help on options:**
  - dumpNameSpace -?
- **Dump server on localhost:2809 from cell root**
  - dumpNameSpace
- **Dump server on localhost:2806 from cell root**
  - dumpNameSpace -port 2806
- **Dump server on yourhost.ibm.com:2811 from cell root**
  - dumpNameSpace -port 2811 -host yourhost.ibm.com
- **Dump server on localhost:2809 from server root**
  - dumpNameSpace -root server



Summary

- **WebSphere Version 5 naming improves:**
  - Interoperability with other providers of J2EE and CORBA servers
  - Independence of individual WebSphere processes
    - No dependency on a single, central name server
- **Administrative bindings are also supported**
- **New format for formulating bindings**
  - Variety of options for migrating or interoperating with Version 4 clients

Trademarks and Disclaimers

References in this document to IBM products or services do not imply that IBM intends to make them available in every country. The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

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

Lotus, Domino, Freelance Graphics, and Word Pro are trademarks of Lotus Development Corporation and/or IBM Corporation in the United States and/or other countries.

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 trademarks of Microsoft Corporation in the United States, other countries, or both. 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. Other company, product and service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. 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.