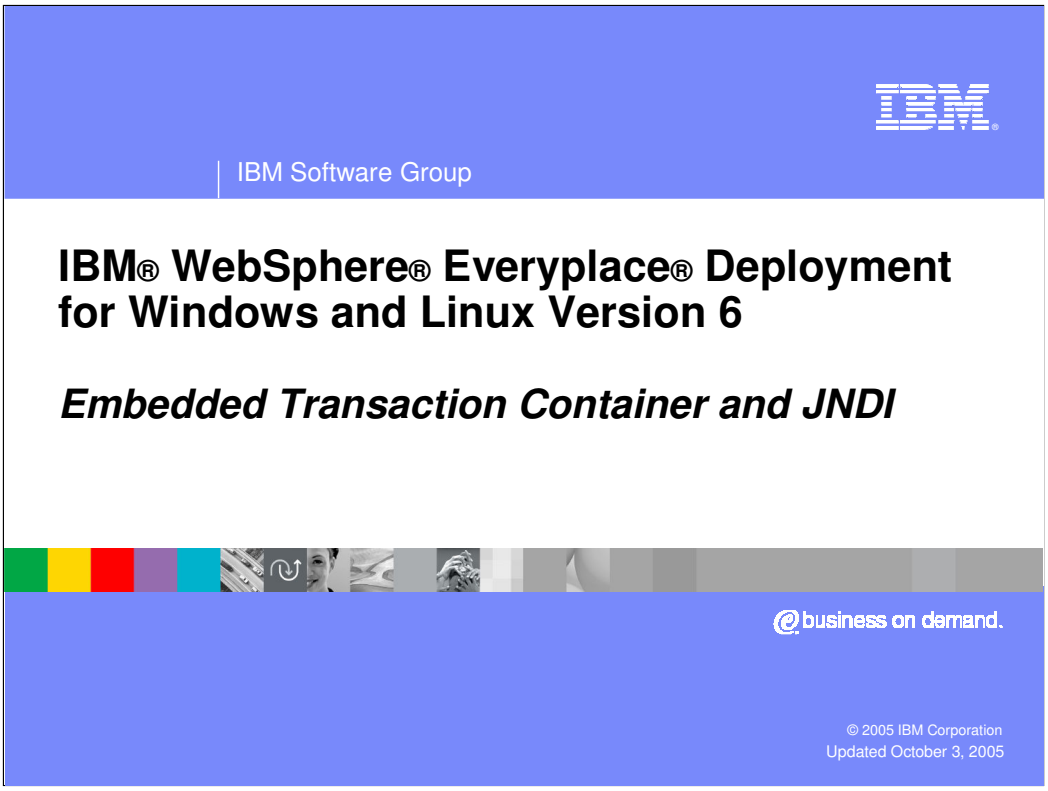
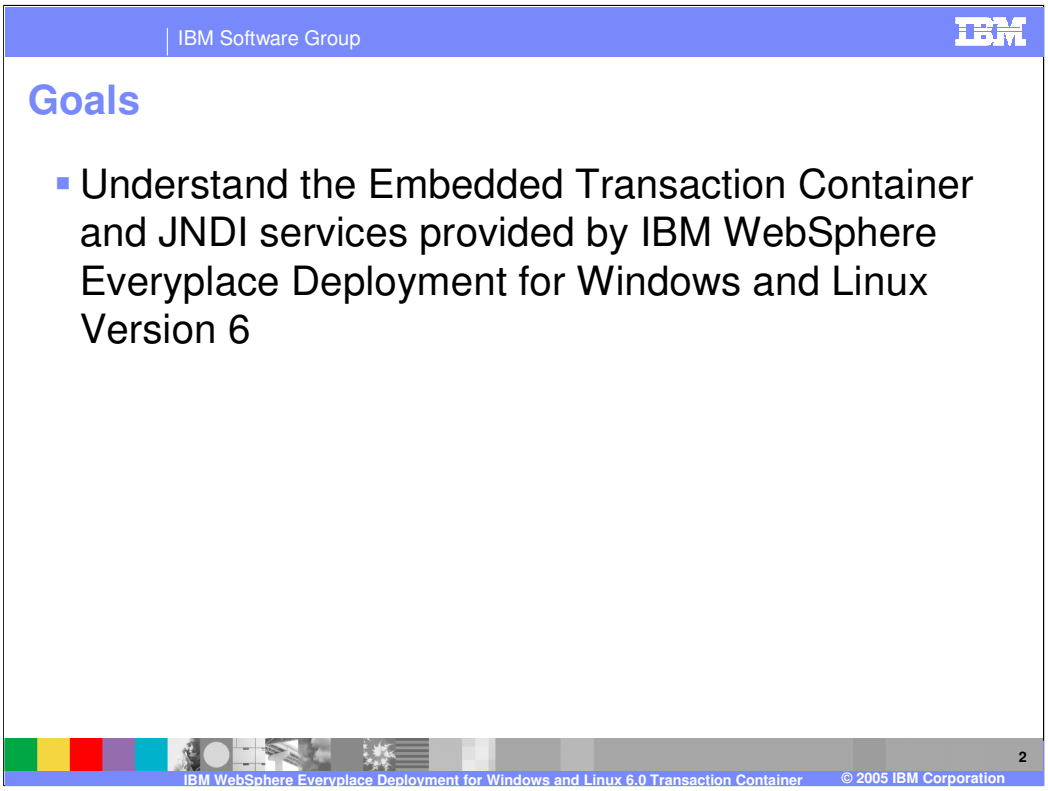
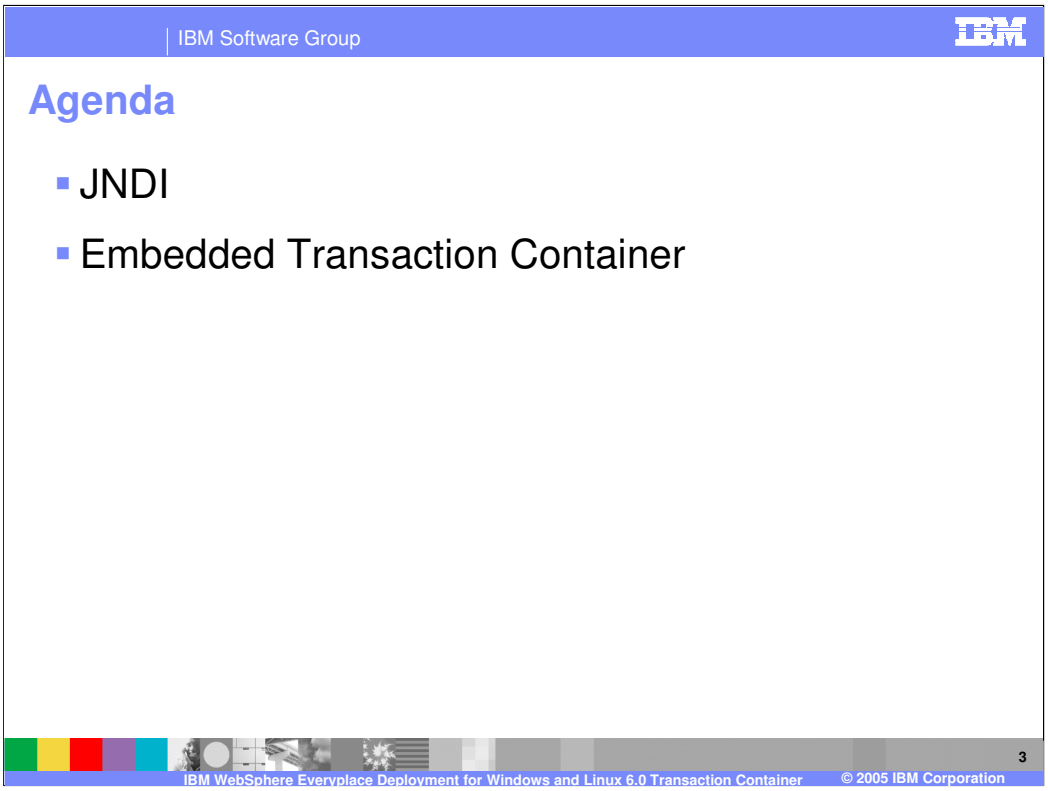


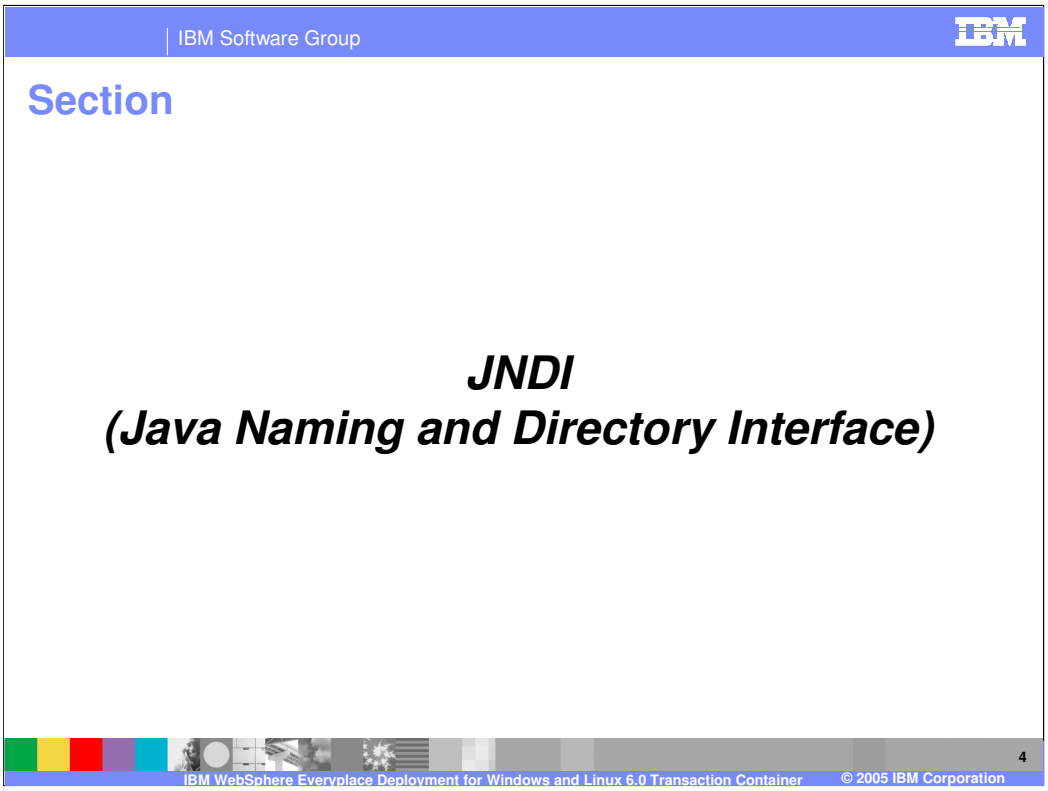
This presentation explains the Embedded Transaction Container and JNDI services supported by IBM WebSphere Everyplace Deployment for Windows and Linux Version 6.



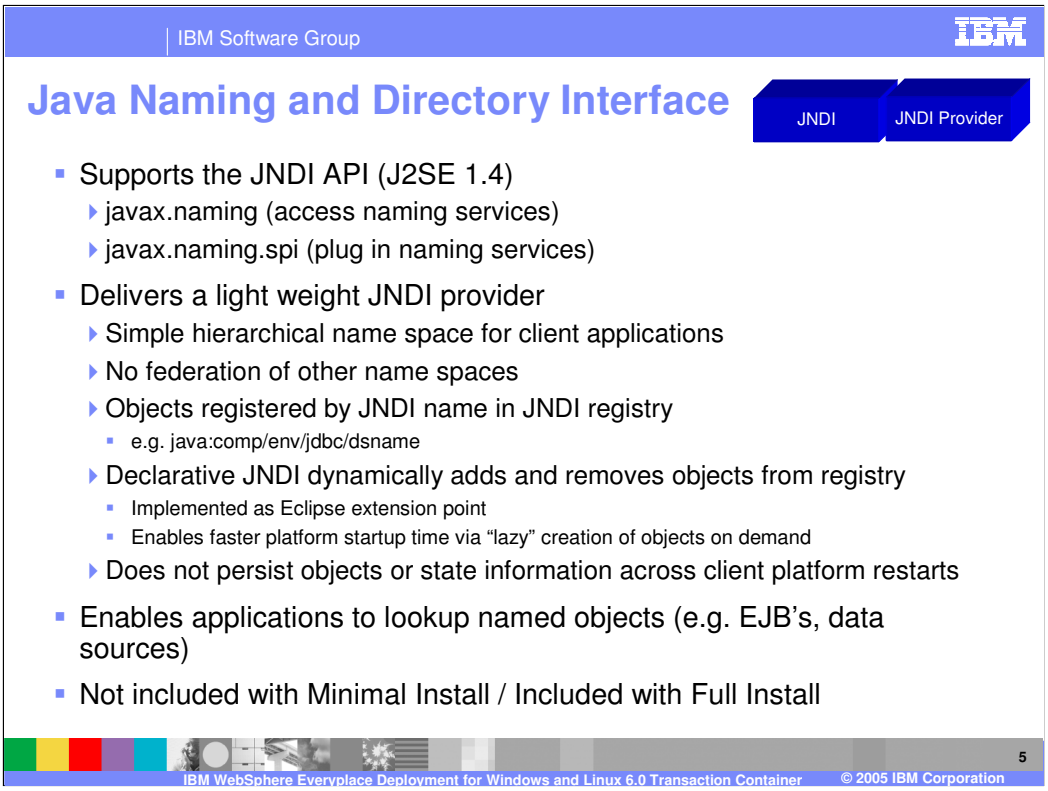
The goal of this presentation is to understand the Embedded Transaction Container and JNDI services supported by IBM WebSphere Everyplace Deployment for Windows and Linux Version 6.



The agenda of this presentation is to explain the JNDI and Embedded Transaction Container services.



Let’s start with an explanation of JNDI.



The client platform provides a simple Java object JNDI registry that enables applications to lookup named objects (e.g. EJB's, data sources).

The JNDI provider enables a local naming directory for objects running in the client platform to communicate via standard Java naming APIs. The runtime client JNDI implementation is very lightweight and does not support federation of other name spaces, rather it provides a simple hierarchical name space for client applications. In most cases, applications leveraging JNDI do not need to interact directly with JNDI Name objects and simply use String representations of the names to be bound or located.

The JNDI provider does not persist objects or their state information across platform restarts, so the platform administrator is responsible for binding the objects each time the platform starts and configuring those objects as needed before binding them into the JNDI registry. While the application itself could programmatically register the objects that it needs each time the platform starts, the client platform provides another declarative model for JNDI bindings. Objects that need to be bound into JNDI can be declared using Eclipse extension points, so that when a lookup request is made for a specific object via its JNDI name the JNDI provider will locate the declarative definition, create the object and return it to the client application on-demand. This "lazy" creation of objects provides for faster platform startup and memory allocation based on actual need, rather than expected need.

JNDI is not included with the Minimal Install – it is included with the Full Install of the client platform.

JNDI Object Factories

- Used by JNDI Provider to create objects defined via Extension Points
- Three defined factories
  - com.ibm.pvc.jndi.provider.java.GenericObjectFactory
    - creates objects of any type
    - calls methods on objects
  - com.ibm.pvc.txncontainer.EJBObjectFactory
    - creates EJB Objects
  - com.ibm.pvc.txncontainer.TxnDataSourceObjectFactory
    - creates special type of DataSource for Transaction Container managed transactions

JNDI Binding Extension Point

- com.ibm.pvc.jndi.provider.java.binding
  - defines an object that can be located via JNDI
  - references an object factory that knows how to create the object
  - Extension Registry searched on lookup request

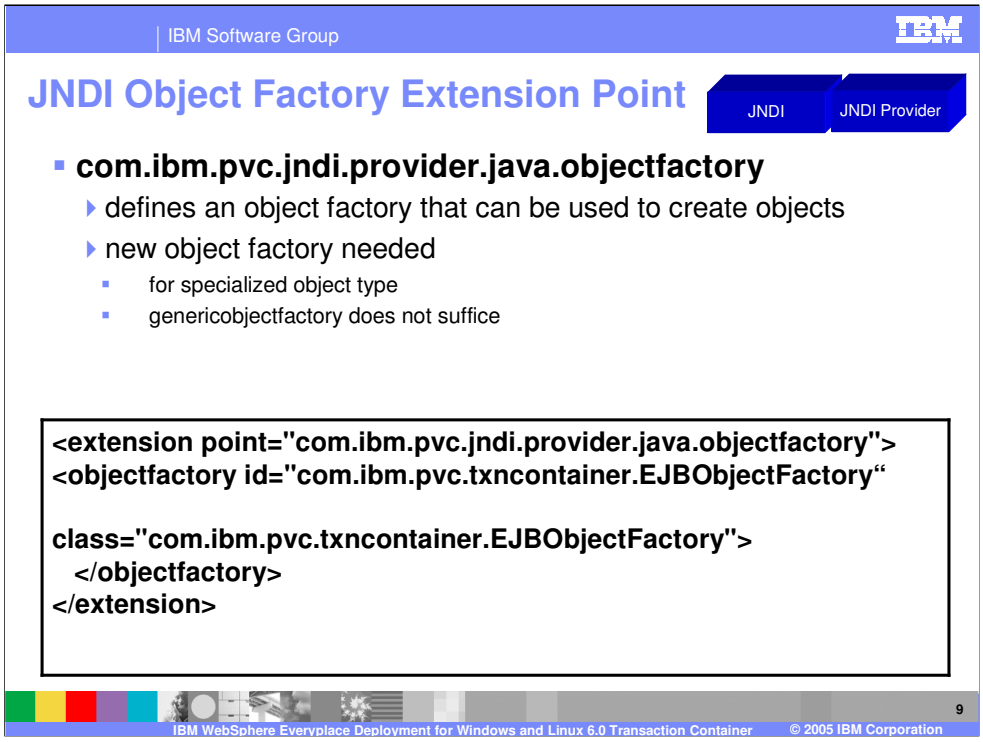
```
<extension point="com.ibm.pvc.jndi.provider.java.binding">
  <binding
     jndi-name="java:comp/env/jdbc/dsname“
    objectFactory-id="com.ibm.pvc.jndi.provider.java.genericobjectfactory">
 </binding>
</extension>
```

JNDI Generic Object Extension Point

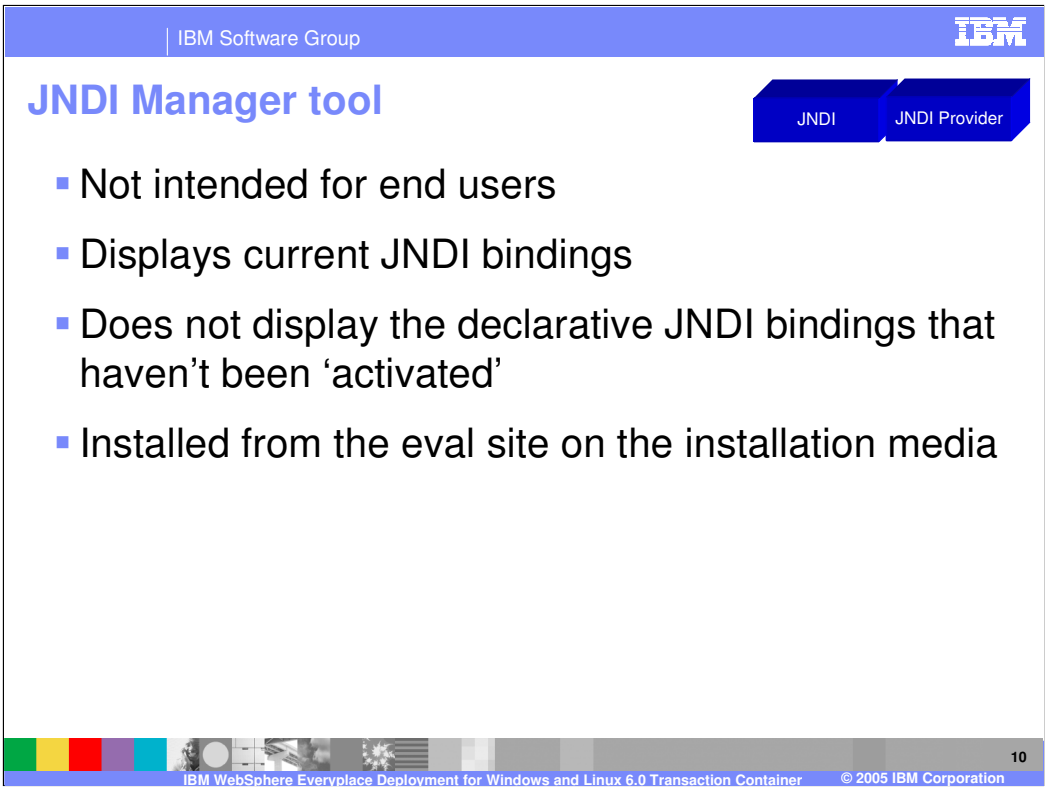
JNDI | JNDI Provider

- **com.ibm.pvc.jndi.provider.java.genericobjectfactory**
  - defines how to create object
    - object class
    - methods with parameters to call on object
  - used for general data source objects

```
<extension
point="com.ibm.pvc.jndi.provider.java.genericobjectfactory">
   <object jndi-name="java:comp/env/jdbc/dsname“
            class="com.ibm.db2e.DB2eDataSource">
      <method name="setUrl">
        <method-parameter type="String" value="jdbc:db2e:oedb“/>
      </method>
  </object>
</extension>
```



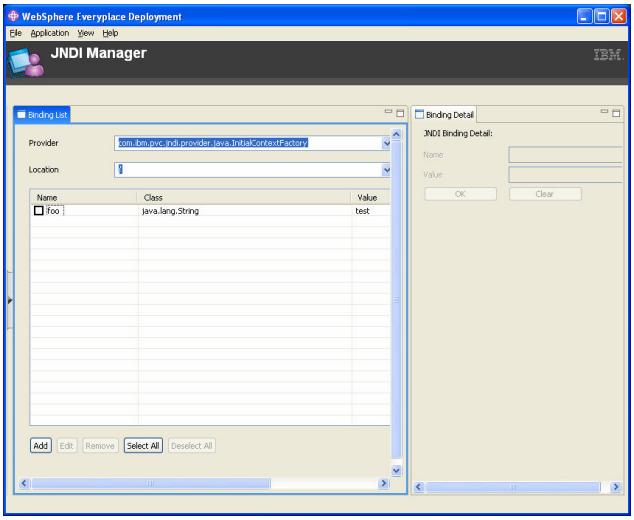
While this capability is provided, it is not expected that you will create your own object factory instance.



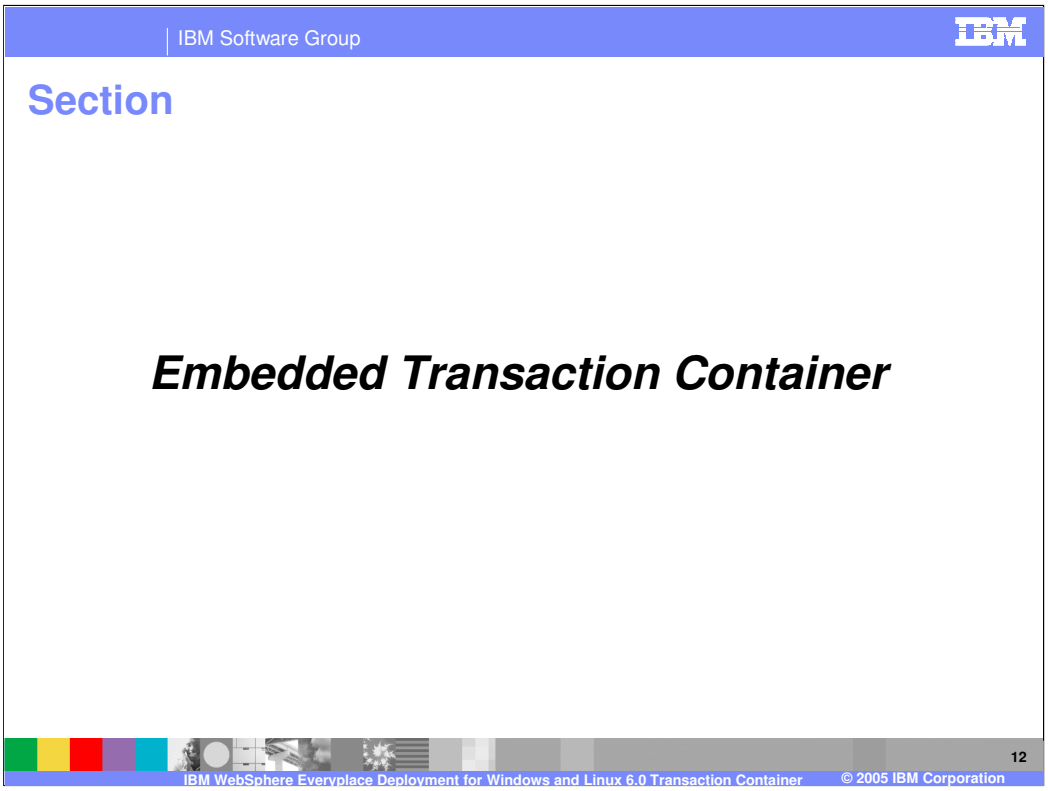
The JNDI Manager is a rich client application tool consisting of one perspective and two views – the Binding List view and the Binding Detail view. The purpose of the Binding List view is to provide easy access to controls which allow for looking up objects bound within the JNDI on the platform. It also provides controls to add, edit or remove those objects. The Binding Detail view is where specific binding information is entered if a binding is to be added or edited.

JNDI Manager

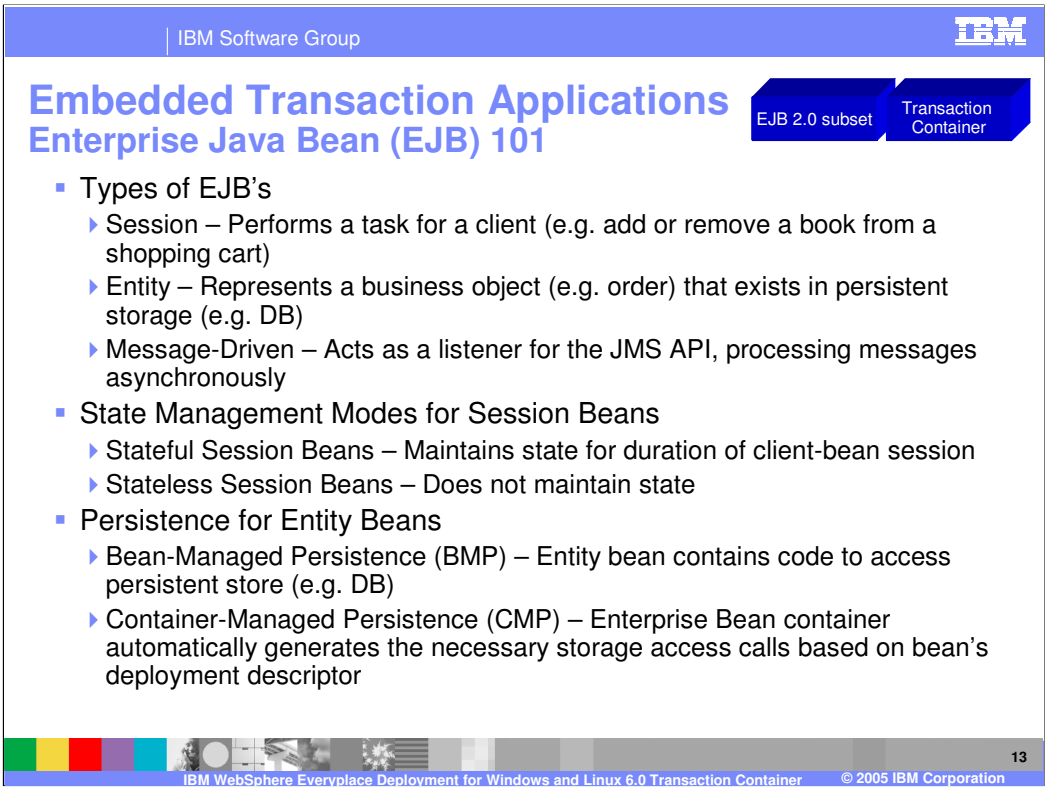
JNDI | JNDI Provider



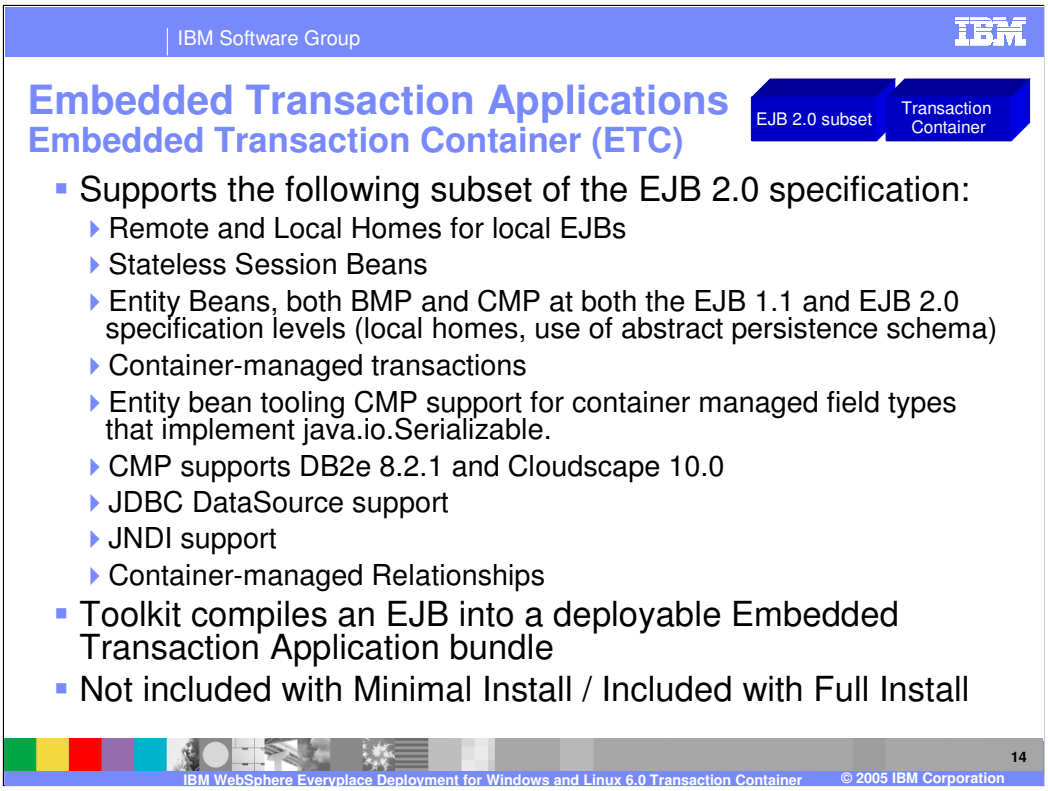
Here is a screen shot of the JNDI Manager tool.



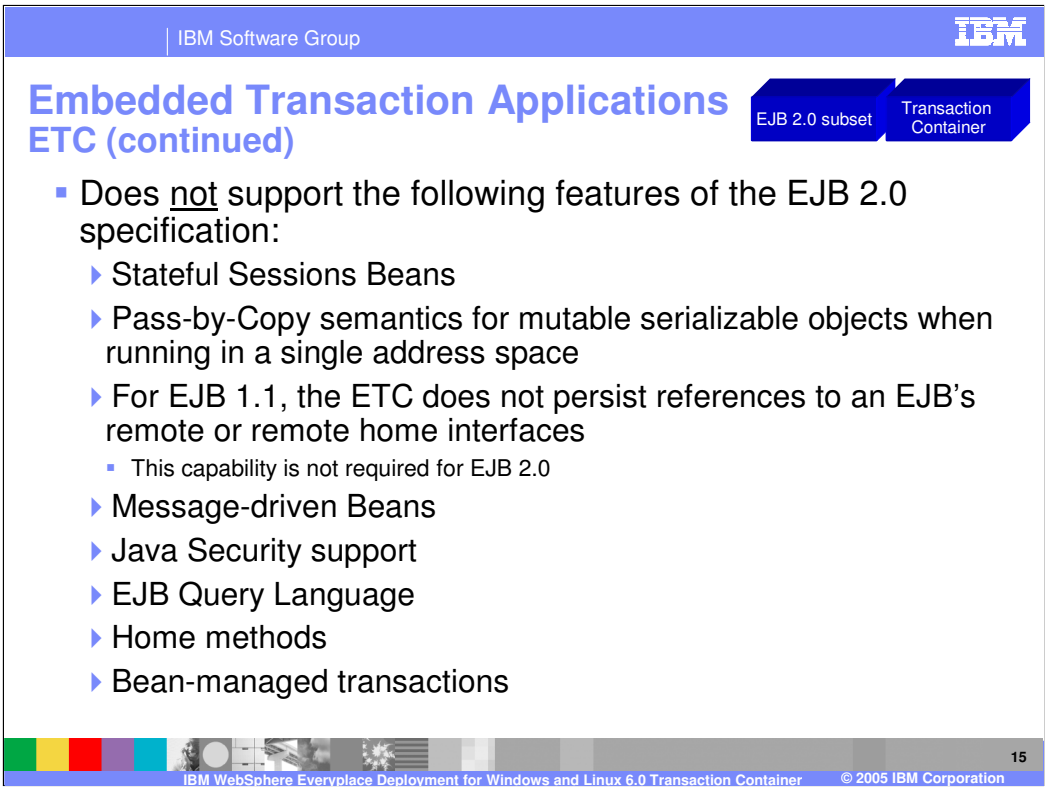
Next, let's study the Embedded Transaction Container.



The Embedded Transaction Container (ETC) provides the ability to deploy Enterprise Java Beans to the client platform. Before we describe the details of the ETC, here is a quick review of key concepts for Enterprise JavaBeans (EJB's).

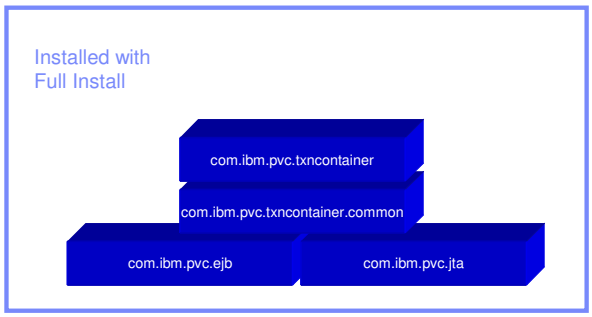


The Embedded Transaction Container (ETC) provides tooling and runtime support for local Enterprise Java Beans (EJBs). The ETC supports a subset of features in the EJB 2.0 specification as shown in this slide. The toolkit compiles and packages an EJB into a deployable bundle, called an Embedded Transaction Application bundle. The Embedded Transaction Application can be deployed onto any platform that is supported by the Embedded Transaction Container. The Embedded Transaction Container is not included with the Minimal Install – it is included with the Full Install of the client platform.



The ETC is designed to be a light weight container and, therefore, does not support certain features in the EJB 2.0 specification as shown in this slide. Please keep these restrictions in mind as you develop your Embedded Transaction Applications.

Transaction Container infrastructure



Plug-in ID	Description
com.ibm.pvc.ejb	EJB APIs
com.ibm.pvc.jta	JTA APIs
com.ibm.pvc.txncontainer.common	Common classes shared between runtime and tools
com.ibm.pvc.txncontainer	Transaction Container Implementation

This slide shows the Transaction Container infrastructure on the client platform, including the plug-ins (components) installed with the Full Install of the client platform and a description of each of these plug-ins.

Transaction Container Serviceability

EJB 2.0 subset | Transaction Container

- Tracing capabilities provided specific to Transaction Container
- Set system properties to enable tracing
- Output typically written to OSGi Log Service
- Documented in Application Developer's Guide

EJB Packaging

EJB 2.0 subset | Transaction Container

- One or more beans exist in each bundle
- Definitions for JNDI binding generated by tools
- Bundle containing bean is started when bean lookup occurs

Template Revision: 9/28/2005 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.