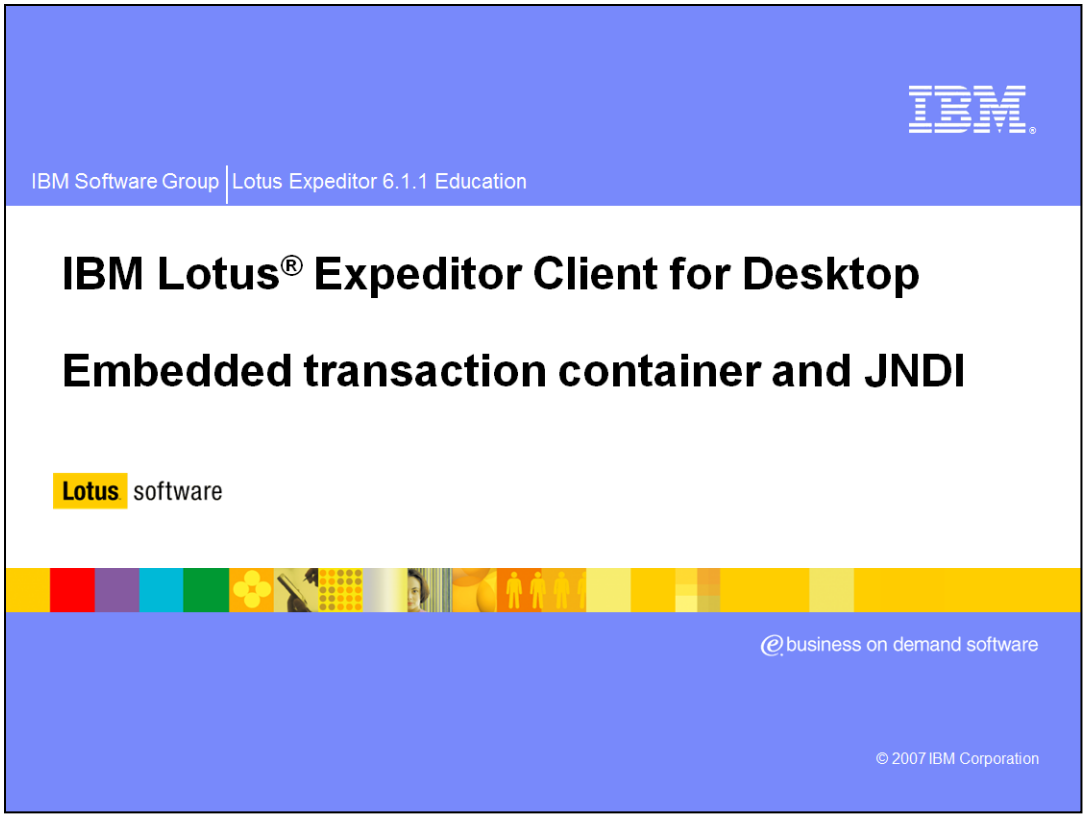
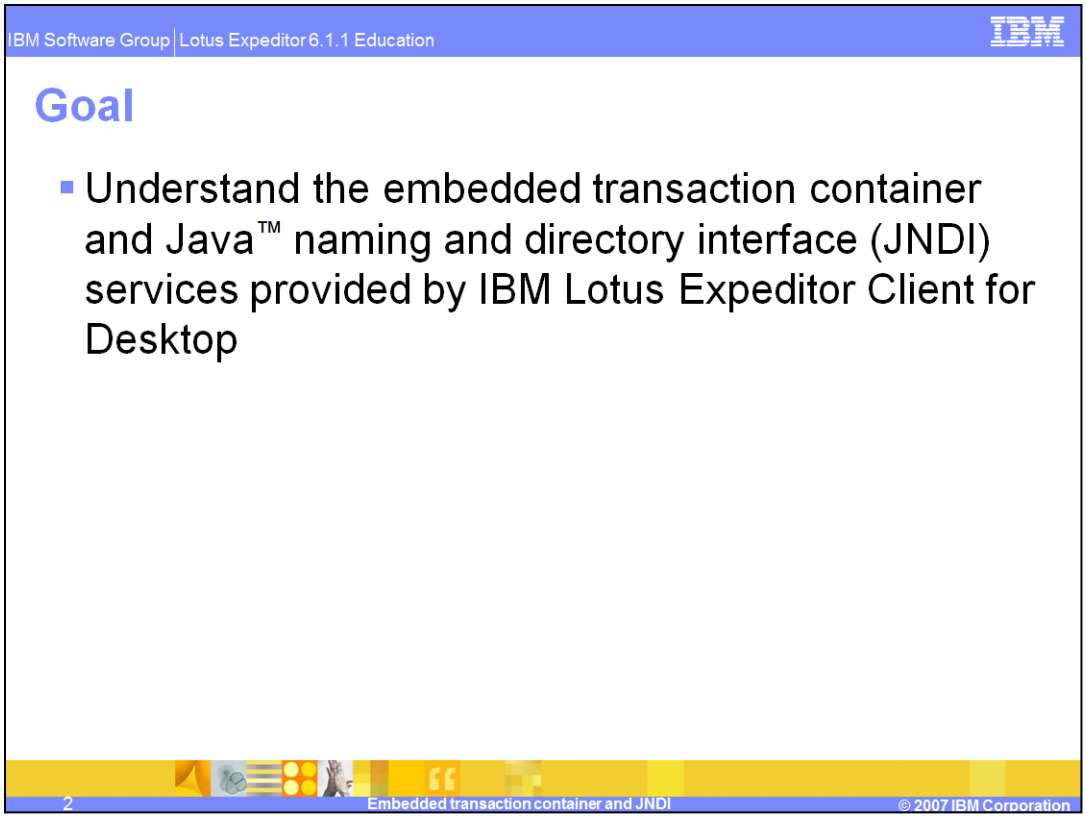
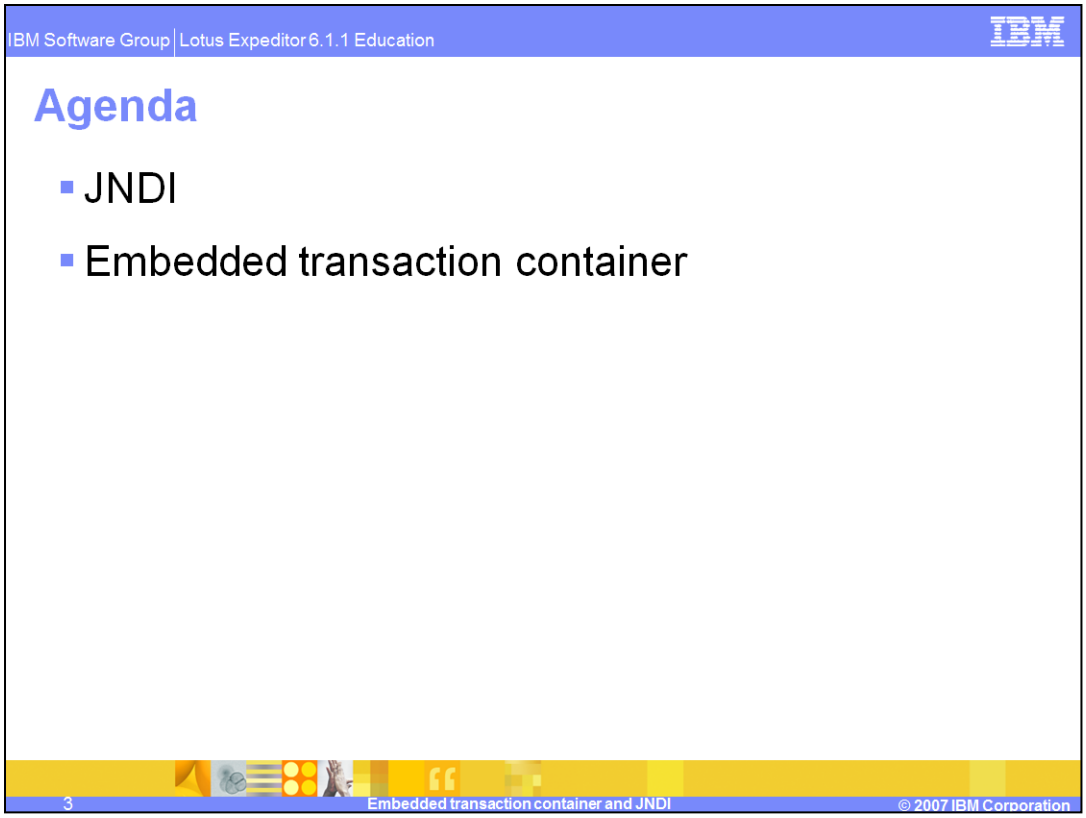


This presentation explains the Embedded Transaction Container and JNDI services supported by IBM Lotus Expeditor Client for Desktops.

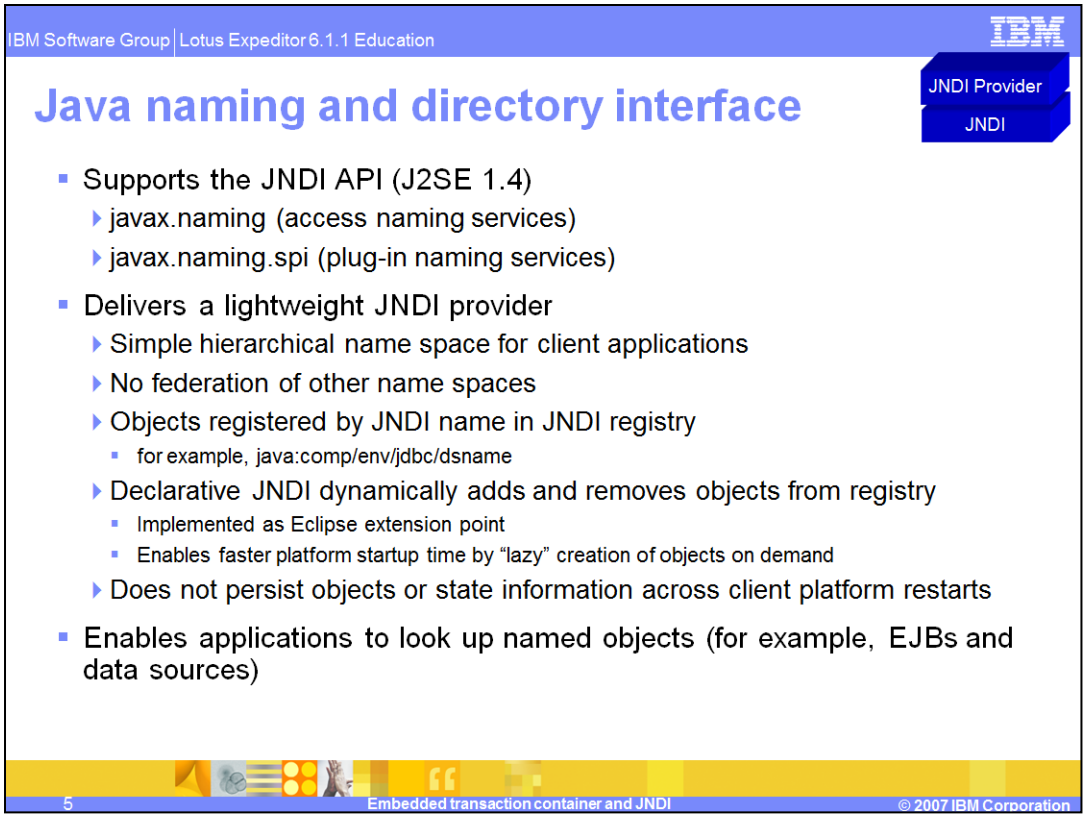


The goal of this presentation is to understand the Embedded Transaction Container and JNDI services supported by IBM Lotus Expeditor Client for Desktop.



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

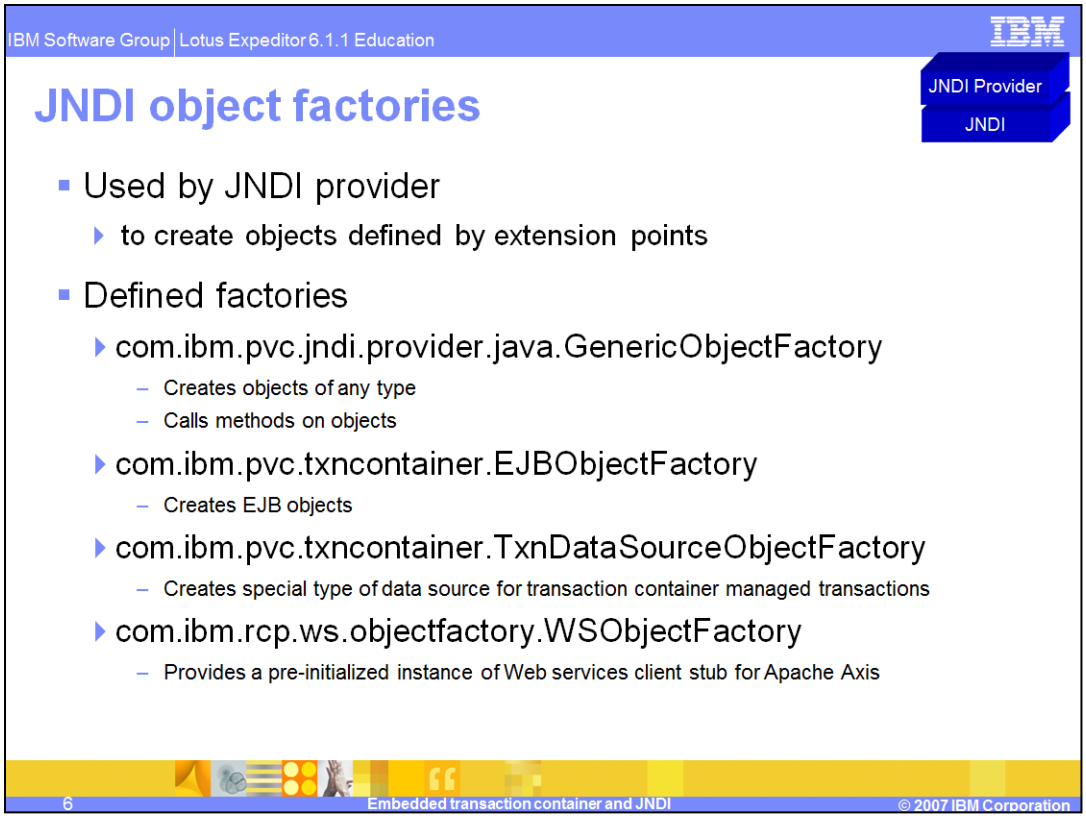
Let's start with an explanation of JNDI.



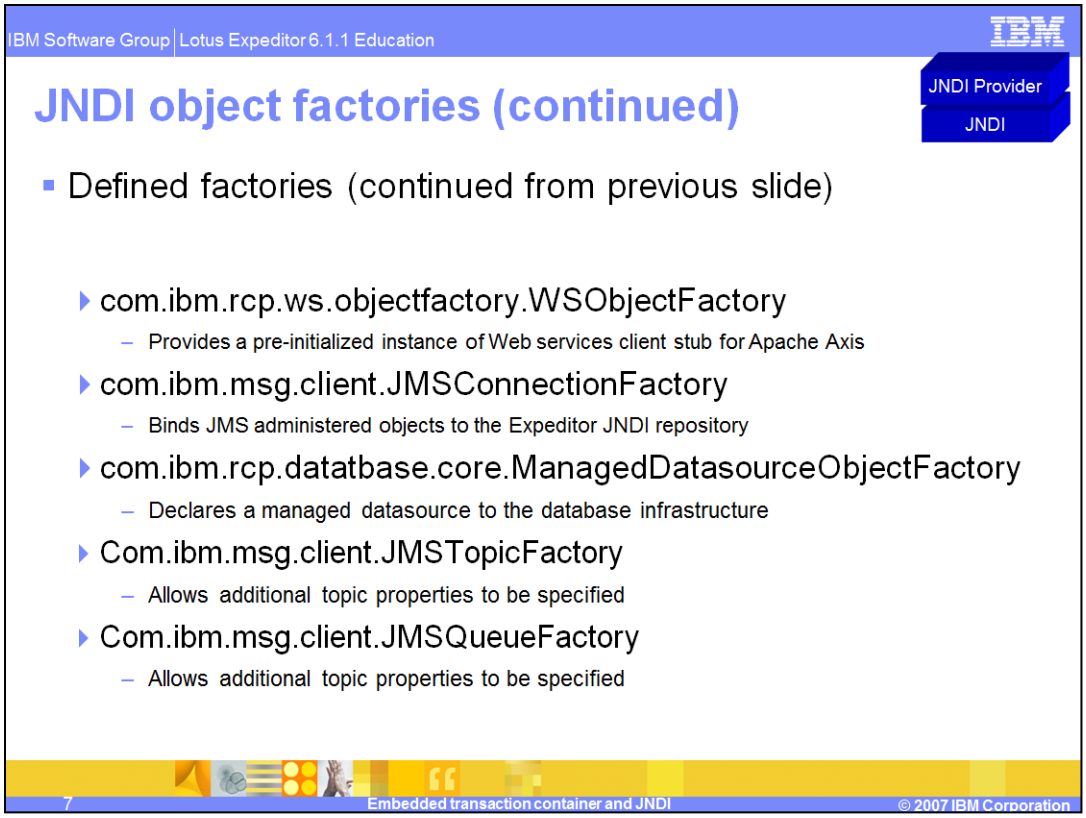
The client platform provides a simple Java object JNDI registry that enables applications to lookup named objects (for example, EJBs and data sources).

The JNDI provider enables a local naming directory for objects running in the client platform to communicate using 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 using 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.



The JNDI object factories are used by JNDI provider to create objects defined using extension points. The defined factories are listed on this slide and the next slide.



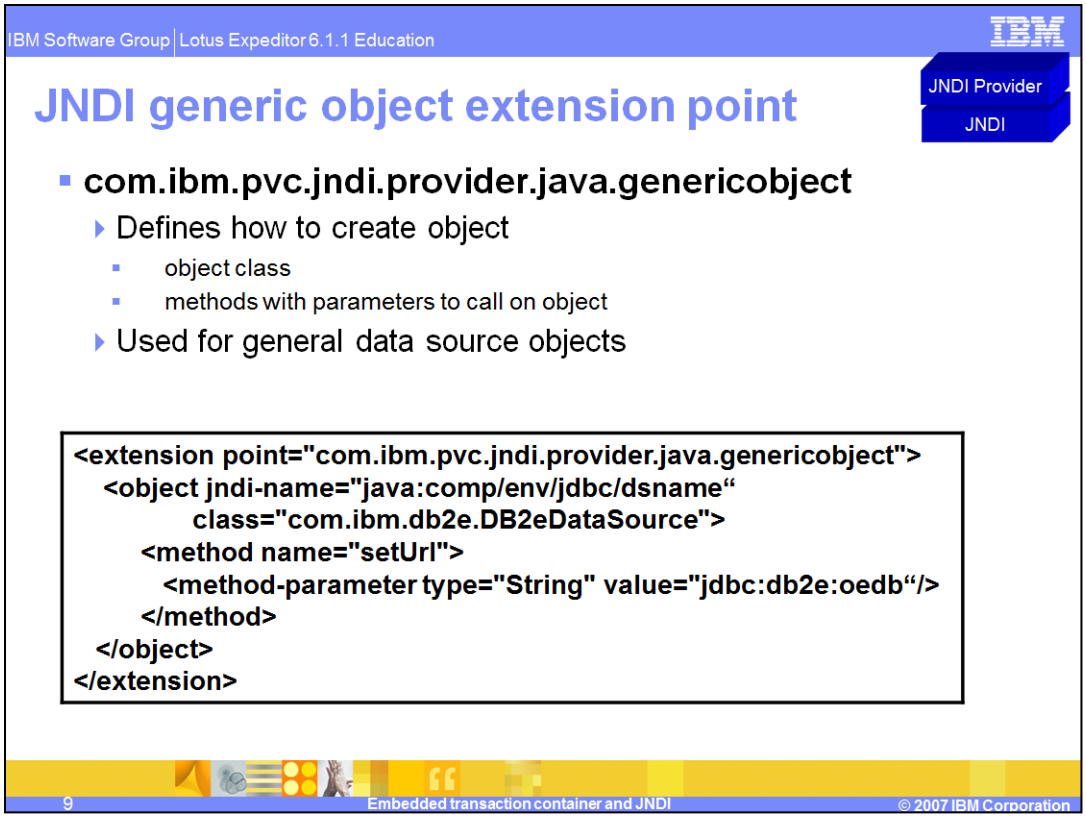
The JNDI object factories are used by JNDI provider to create objects defined using extension points. The defined factories listed on this slide are continued from those listed on the previous slide.

JNDI Provider
JNDI

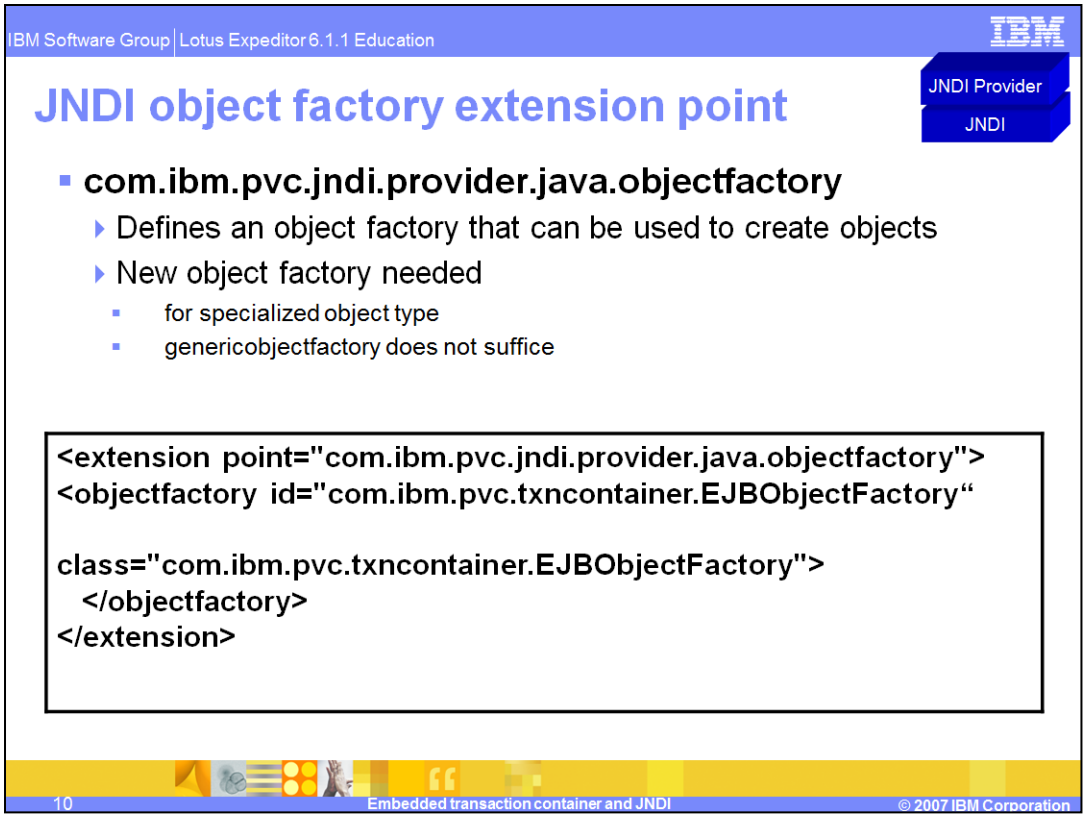
JNDI binding extension point

- com.ibm.pvc.jndi.provider.java.binding
  - Defines an object that can be located using JNDI
  - References an object factory that knows how to create the object
  - Extension registry searched on lookup request

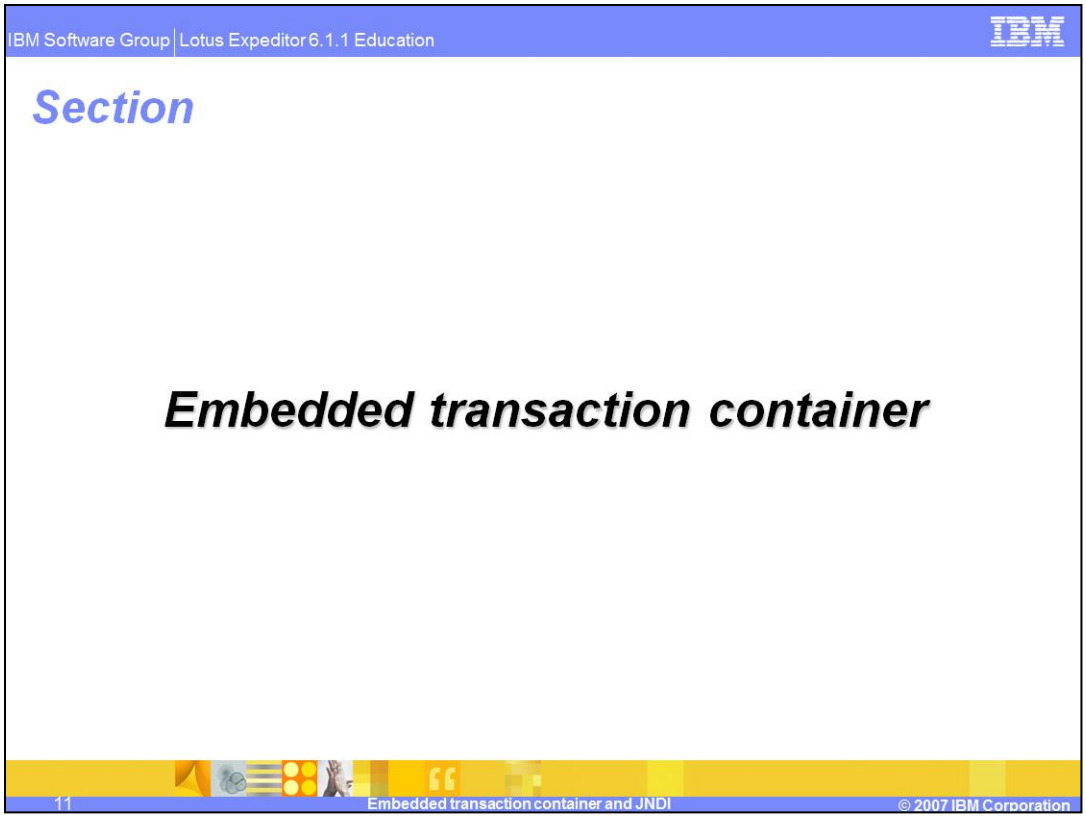
```
<extension point="com.ibm.pvc.jndi.provider.java.binding">
  <binding
     jndi-name="java:comp/env/jdbc/dsname“
    objectFactory-
id="com.ibm.pvc.jndi.provider.java.genericobjectfactory">
 </binding>
</extension>
```



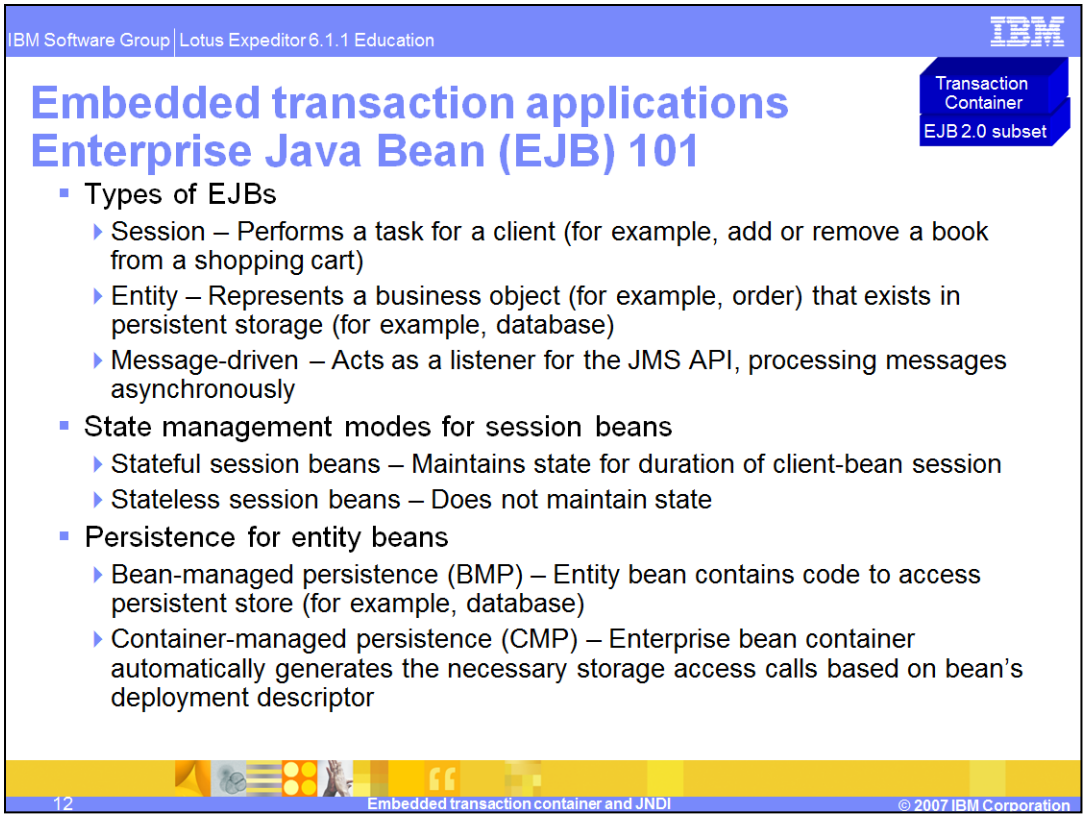
This slide describes the JNDI generic object extension point.



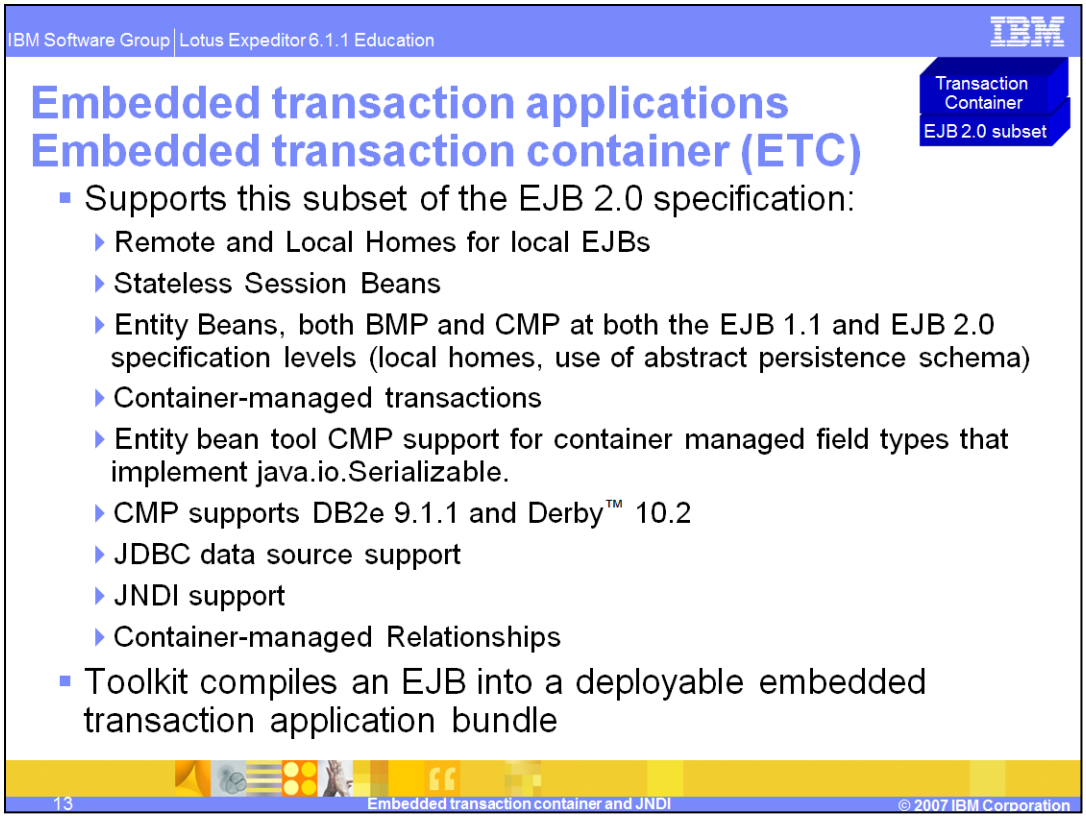
While this capability is provided, it is not expected that you will create your own object factory instance.



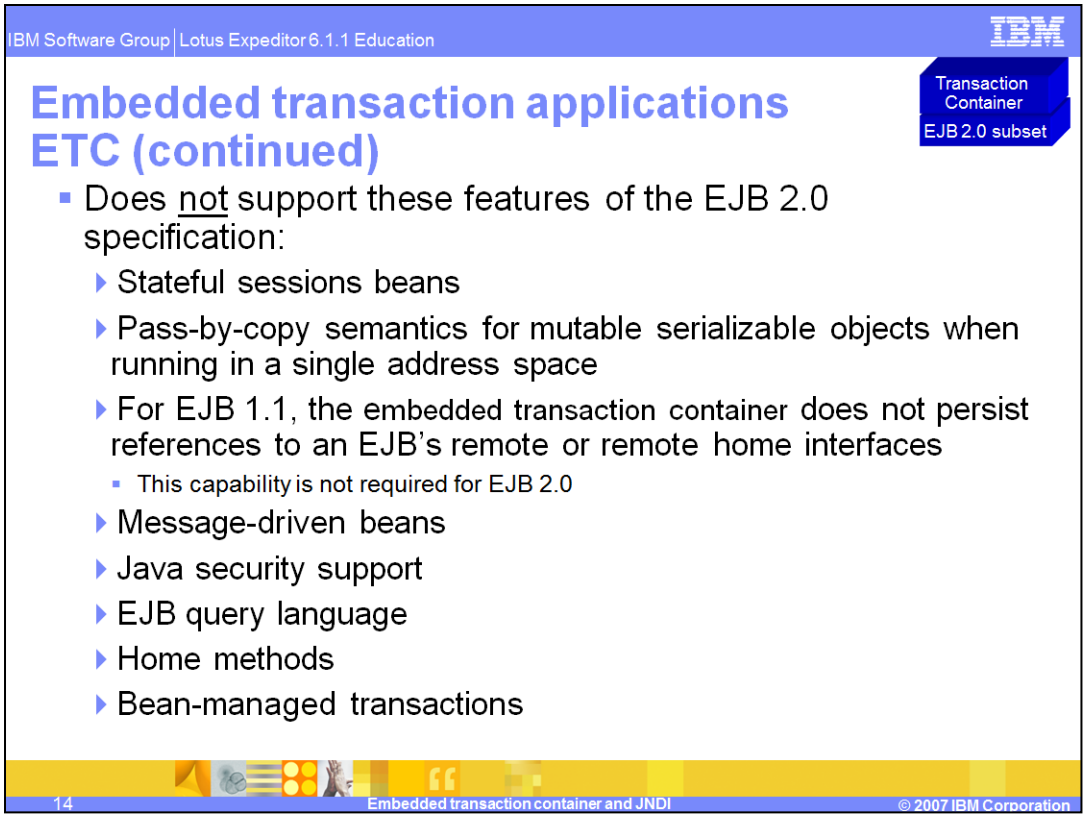
Next, let's study the Embedded Transaction Container.



The Embedded Transaction Container (ETC) provides the ability to deploy Enterprise Java Beans to the client platform. Before describing the details of the Embedded Transaction Container, here is a quick review of key concepts for Enterprise JavaBeans (or EJBs).



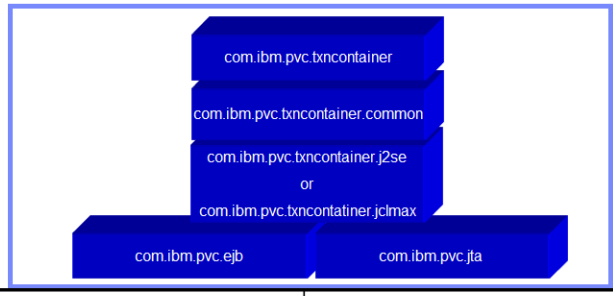
The Embedded Transaction Container provides tools and runtime support for local Enterprise Java Beans. The Embedded Transaction Container supports a subset of features in the EJB 2.0 specification as shown in this slide. The toolkit compiles and packages an EJB into a deployable bundle, called an Embedded 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 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. Keep these restrictions in mind as you develop your Embedded Transaction Applications.

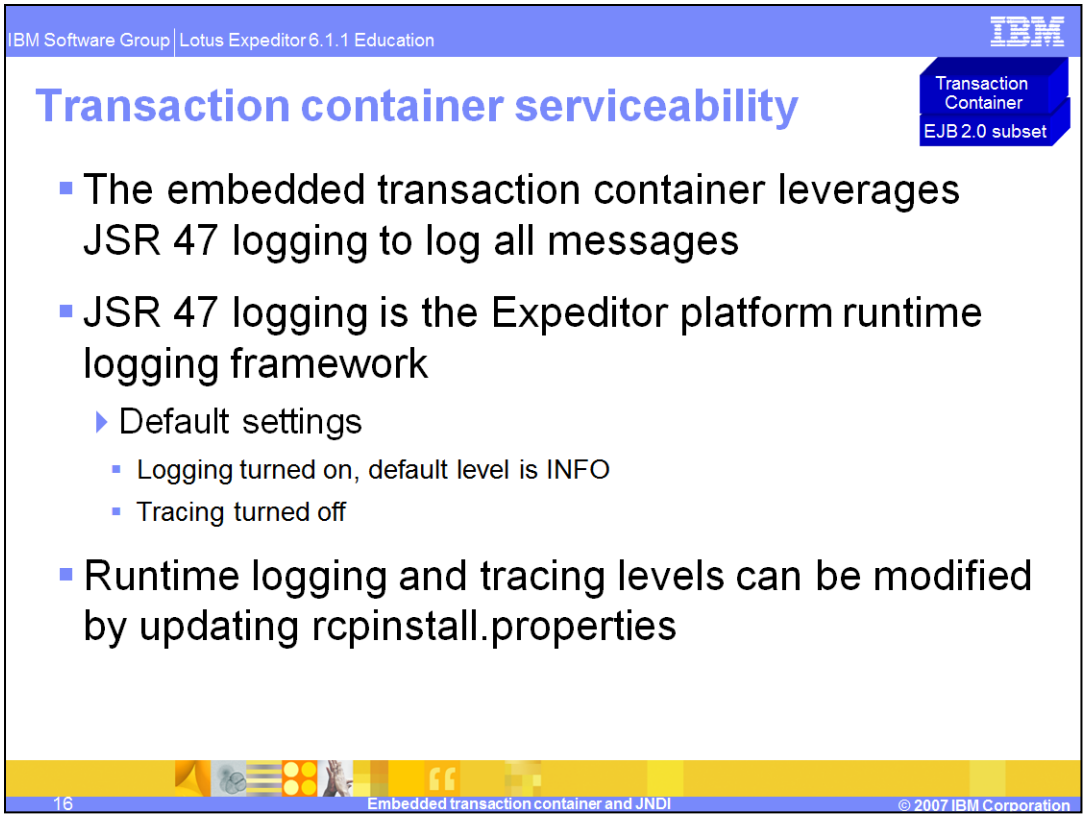
Transaction container infrastructure

Transaction Container
EJB 2.0 subset

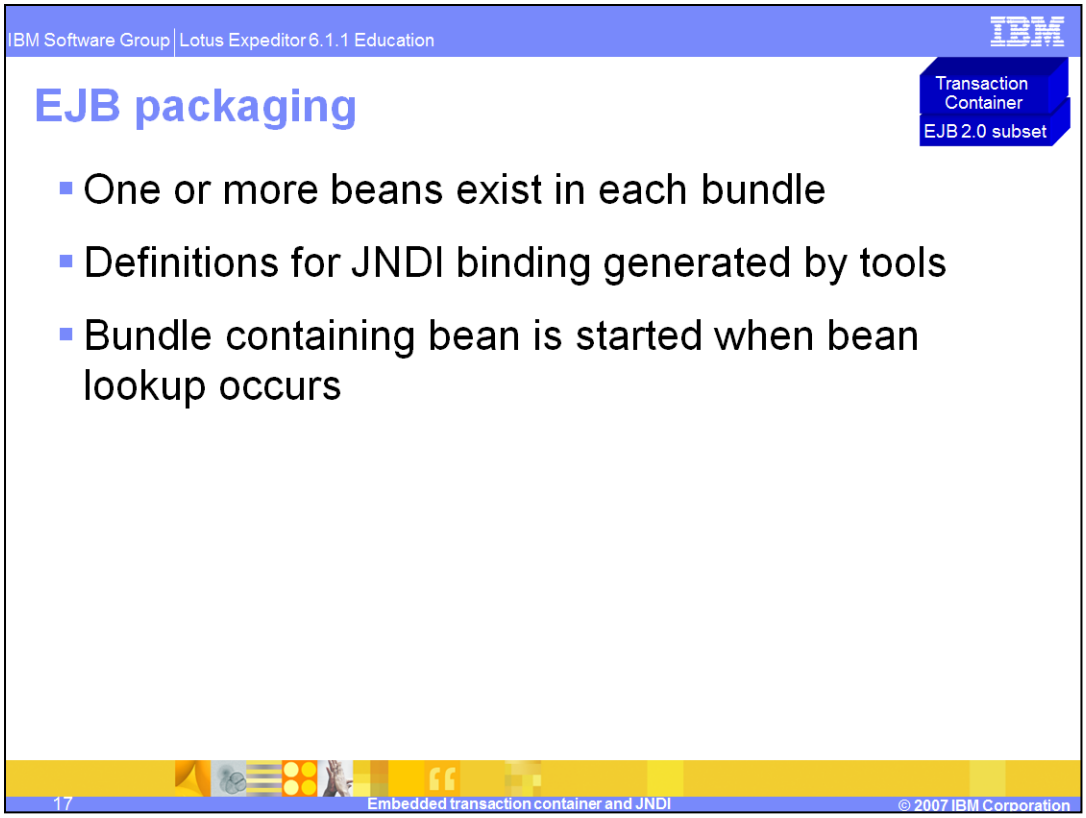


Plug-in ID	Description
com.ibm.pvc.ejb	EJB APIs
com.ibm.pvc.jta	JTA APIs
com.ibm.pvc.txncontainer.common	Common classes shared between runtime and tools
com.ibm.pvc.txncontainer.j2se	JSR 169 JDBC interface (for device)
com.ibm.pvc.txncontainer.jclmax	Fully compliant J2SE JDBC interface
com.ibm.pvc.txncontainer	Transaction Container Implementation

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



The Embedded Transaction Container incorporates logging and tracing for serviceability. JSR 47 logging is used to log all messages. Runtime logging and tracing levels can be modified by updating rcpinstall.properties. Details can be found in the guide *Developing applications for Lotus Expeditor*.



What is included in EJB packaging? One or more beans can exist in each bundle, definitions for JNDI binding generated by tools. The bundle containing the bean is started when bean lookup occurs.

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 Lotus

EJB, J2SE, Java, JDBC, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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 or changes in the products or programs 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 (for example, 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.

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.