TIBCO ActiveMatrix BusinessWorks™ Plug-in for EJB User's Guide

Software Release 6.0
November 2014

Two-Second Advantage®

Important Information

Contents

TIBCO Documentation and Support Services

All TIBCO documentation is available on the TIBCO Documentation site, which can be found here:

https://docs.tibco.com

Product-Specific Documentation

Documentation for TIBCO products is not bundled with the software. Instead, it is available on the TIBCO Documentation site. To directly access documentation for this product, double-click the following file:

TIBCO_HOME/release_notes/TIB_bwpluginejb_*version*_docinfo.html

The following documents for this product can be found on the TIBCO Documentation site:

- *TIBCO ActiveMatrix BusinessWorks Plug-in for EJB Installation*
- *TIBCO ActiveMatrix BusinessWorks Plug-in for EJB User's Guide*
- *TIBCO ActiveMatrix BusinessWorks Plug-in for EJB Release Notes*

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support as follows:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to:

http://www.tibcommunity.com

Plug-in Overview

You can use TIBCO ActiveMatrix BusinessWorks Plug-in for EJB to connect to J2EE-compliant application servers and invoke Enterprise JavaBeans (EJB) components, or enterprise beans on the servers.

TIBCO ActiveMatrix BusinessWorks is an easy-to-use integration product suitable for enterprise, web, and mobile applications. It uses TIBCO Business Studio, which is an Eclipse graphical user interface (GUI), to define business processes, and it uses the process engine to execute the business processes.

TIBCO ActiveMatrix BusinessWorks Plug-in for EJB plugs into TIBCO ActiveMatrix BusinessWorks, which connects TIBCO ActiveMatrix BusinessWorks with EJB containers.

TIBCO ActiveMatrix BusinessWorks supports plug-ins to extend the palette functionality. After installing the plug-in, an EJB Configuration shared resource and an EJB palette become available in the TIBCO Business Studio. You can add the plug-in activities to the business process you are designing, and integrate them with the BusinessWorks process.

At runtime, the plug-in activities are executed as part of the BusinessWorks process execution. Each plug-in consists of activities which share common functionality and properties.

The following three activities are included in the EJB Palette:

- Use the EJB2Home activity to retrieve EJB home object and create EJB remote object for EJB 2.x.
- Use the EJB2Remote activity to invoke the EJB remote method which is deployed on the EJB server for EJB 2.x.
- Use the EJB3Remote activity to get EJB remote object and invoke remote method which is deployed on the EJB server for EJB 3.x.

Integrating with JMS

To handle the message-driven beans, you must integrate EJB palette of this plug-in with the JMS palette of TIBCO ActiveMatrix BusinessWorks. Designing the BusinessWorks processes with this plug-in, only session beans and entity beans are supported.

Sending or Receiving Java Objects

Occasionally input parameters or return values for enterprise beans are Java objects. To send or receive a Java object to or from an EJB, use the Java palette to create the object within the process definition. For more information on working with Java objects in process definitions, see *TIBCO ActiveMatrix BusinessWorks Bindings and Palettes Reference*.

Examples:

- To call a remote method that requires a Java object as an input parameter, create an object with the Java Invoke activity.
- To use a Java object received as a return value from an EJB, parse an object to the Java Invoke activity.

EJB Overview

An EJB component is a software component that encapsulates the business logic of an application.

Enterprise JavaBeans (EJB) technology is the server-side component architecture for Java Platform, Enterprise Edition (Java EE). EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.

Use EJB Palette of TIBCO ActiveMatrix BusinessWorks Plug-in for EJB to connect to J2EE-compliant application servers and invoke Enterprise JavaBeans (EJB) components or enterprise beans on the servers.

The EJB specification includes support for Java Transaction API (JTA) UserTransactions. TIBCO ActiveMatrix BusinessWorks provides support for these transactions and you can call enterprise beans within a client-managed transaction. See *TIBCO ActiveMatrix BusinessWorks Application Development* for more information about transactions.

Accessing an EJB

The EJB standard defines mechanisms for a client machine to access an EJB 2.x or an EJB 3.x entity:

- To access an EJB 2.x, you need to obtain a reference to a JNDI server, and perform a JNDI lookup operation to obtain a reference to a home object and also a reference to a remote object from the home object. Then you can invoke methods on the remote object.
- To access an EJB 3.x, you need to obtain a reference to a JNDI server, and perform a JNDI lookup operation to obtain a reference to a remote object. Then you can invoke methods on the remote object.

Getting Started

A typical workflow for using the plug-in to create a process, test a process in the debugger, and deploy the application.

Prerequisites

Ensure your EJB application server is running and EJBs are deployed on the EJB application server before using the plug-in.

TIBCO ActiveMatrix BusinessWorks uses the Eclipse graphical user interface (GUI) provided by TIBCO Business Studio to define business processes and generate Enterprise Archives (EAR) files. The EAR files are deployed and run in the ActiveMatrix BusinessWorks runtime, and also are managed by TIBCO Enterprise Administrator (TEA).

The typical workflow for using the plug-in is:

1. Creating a Project
2. Connecting to an EJB Server
3. Creating an EJB Configuration Shared Resource
4. Configuring EJB Client JAR Files
5. Configuring a Process
6. Debugging and Running a Process
7. Deploying Applications

Creating a Project

Projects are BusinessWorks application modules that are created in TIBCO Business Studio. A project contains various resources.

Procedure

1. Start TIBCO Business Studio.
2. Click **File > New > BusinessWorks Resources**.
3. Click **BusinessWorks Application Module** in the **BusinessWorks Resource** dialog. Click **Next**.

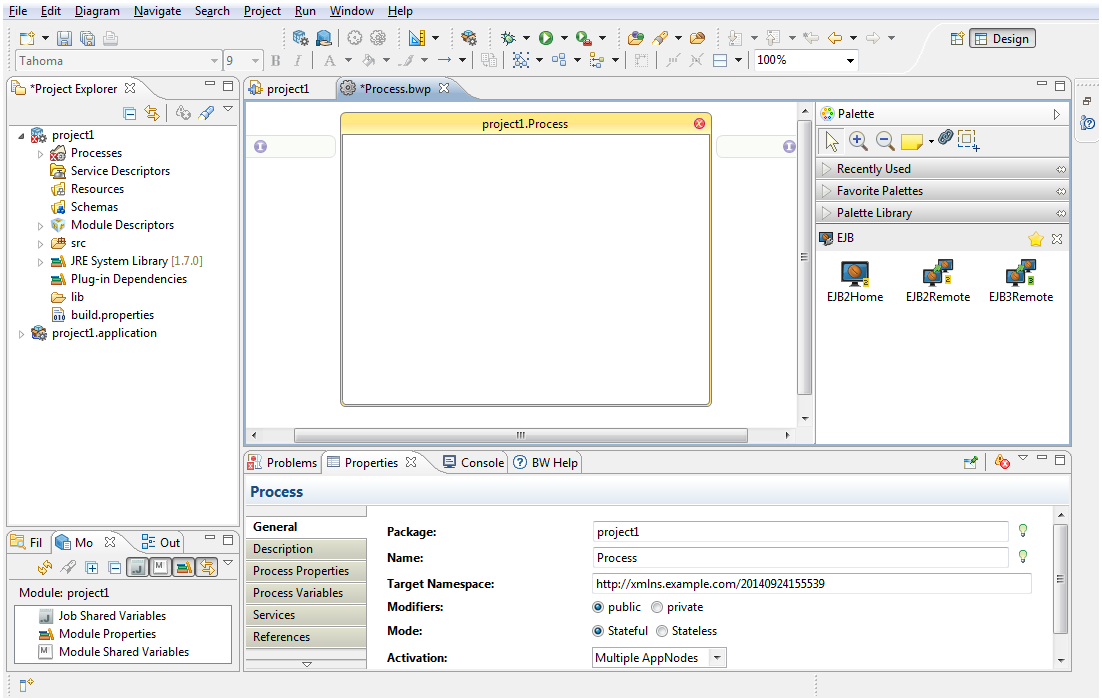
   

   There are several ways to open the **New BusinessWorks Application Module** dialog and create a new project in TIBCO Business Studio. See the TIBCO ActiveMatrix BusinessWorks documentation for more information.
4. Type a name for the project that you are creating in the **Project name** field.
5. Select the **Use default location**, **Create empty process**, **Create Application** and **Use Java configuration** check boxes. Click **Finish**.
6. In the Project Explorer view, click **Module Descriptors > Overview**, click the **MANIFEST.MF** tab, add the following content in the editor:

   ```
   Eclipse-RegisterBuddy: com.tibco.bw.palette.ejb.runtime
   ```

Result

A project and an application are created and displayed in the Project Explorer view. The Process editor is displayed automatically.



Connecting to an EJB Server

Connect to an EJB server before invoking any EJB using TIBCO ActiveMatrix BusinessWorks Plug-in for EJB.
TIBCO ActiveMatrix BusinessWorks Plug-in for EJB supports the following application server vendors:

- JBoss Application Server version 6.x
- JBoss Application Server version 7.1.x and WildFly 8.x
- Oracle WebLogic Server version 12.x
- IBM WebSphere Application Server version 8.x

Depending on your application server vendor, perform the following tasks to connect to the EJB server:

- Connecting to JBoss Application Server
- Connecting to Oracle WebLogic Server
- Connecting to IBM WebSphere Application Server

Connecting to JBoss Application Server

Prerequisites

Ensure you have created a project in TIBCO ActiveMatrix BusinessWorks.

Procedure

1. Copy all `.jar` files into a local directory.

   - The `.jar` files are located in the *`JBoss_Home`*`/client` directory, where *`JBoss_Home`* is the directory your JBoss Application Server version 6.x installs into.
   - The `.jar` files are located in the *`JBoss_Home`*`/bin/client` directory, where *`JBoss_Home`* is the directory your JBoss Application Server version 7.1.x and WildFly 8.x install into.

   For more information about the server JAR files, see related documents from your EJB server vendor.

2. In this local directory, create `.properties` files and add contents to each `.properties` file. Create `.properties` files based on the version of your JBoss Application Server.

   - JBoss Application Server version 6.x

     Create a `.properties` file with file name `jndi.properties` and add the following content to the file:

     ```
     java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
     java.naming.provider.url=jnp://<JBoss6_IP_Address>:<Port>
     java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
     ```

   - JBoss Application Server version 7.1.x and WildFly 8.x

     Create a `.properties` file with file name `jndi.properties` and add the following content to the file:

     ```
     jboss.naming.client.ejb.context=true
     java.naming.security.principal=remote://<JBoss7_IP_Address>:<Port>
     java.naming.factory.initial=org.jboss.naming.remote.client.
     InitialContextFactory
     java.naming.security.principal=<username>
     java.naming.security.credentials=<password>
     ```

     Create another `.properties` file with file name `jboss-ejb-client.properties` and add the following content to the file:

     ```
     remote.connectionprovider.create.options.org.xnio.Options.SS
     L_ENABLED=false
     remote.connections=default
     remote.connection.default.host=<JBoss7_IP_Address>
     remote.connection.default.port=<Port>
     remote.connection.default.connect.options.org.xnio.Options.S
     ASL_POLICY_NOANONYMOUS=false
     remote.connection.default.username=<username>
     remote.connection.default.password=<password>
     ```

3. Expand the created project in the Project Explorer view, drag the local directory to the **lib** folder.
4. Click **Link to files and folders** in the prompted **File and Folder Operation** dialog and press Enter.

   If you want to deploy the application to TIBCO Enterprise Administrator, click **Copy files and folders**.

5. Click **Window > Open Perspective > Others > Java** and press Enter.
6. In the Package Explorer view, expand **META-INF**, right-click **MANIFEST.MF**; then click **Open With > Plug-in Manifest Editor**.
7. In the **Runtime** tab, click **Add** in the Classpath panel.
8. In the **Jar Selection** window, click **lib > the local directory name** and press Enter.

Connecting to Oracle WebLogic Server

Prerequisites

Ensure you have created a project in TIBCO ActiveMatrix BusinessWorks.

Procedure

1. Copy all `.jar` files into a local directory.

   The `.jar` files are located in the *`WebLogic_Home`*`/server/lib` directory, where *`WebLogic_Home`* is the directory your Oracle WebLogic Server installs into.

   For more information about the server JAR files, see related documents from your EJB server vendor.
2. Expand the created project in the Project Explorer view, drag the local directory to the **lib** folder.
3. Click **Link to files and folders** in the prompted **File and Folder Operation** dialog and press Enter.

If you want to deploy the application to TIBCO Enterprise Administrator, click **Copy files and folders**.

Connecting to IBM WebSphere Application Server

Prerequisites

Ensure you have created a project in TIBCO ActiveMatrix BusinessWorks.

Procedure

1. Copy all `.jar` files into a local directory.

   The `.jar` files are located in the *`WebSphere_Home`*`/AppServer/runtimes` directory, where *`WebSphere_Home`* is the directory your IBM WebSphere Application Server installs into.

   For more information about the server JAR files, see related documents from your EJB server vendor.
2. Copy the following files from the IBM WebSphere Application Server to the local directory:
   - *`WebSphere_Home`*`/AppServer/profiles/AppSrv01/properties/ssl.client.props`
   - *`WebSphere_Home`*`/AppServer/profiles/AppSrv01/properties/sas.client.props`
3. At a command prompt, navigate to the *`TIBCO_HOME`*`/tibcojre/`*`version_number`*`/bin` directory, and then type the following command: **`keytool.exe`**.

   For example:

```
keytool.exe -genkey -v alias test -keystore D:/key.jks
-storepass password
```

   The key store files, `key.jks` and `trust.jks`, are created after this step.

4. Modify the `ssl.client.props` file to customize your environment. You can obtain the file from the WebSphere Application Server installation.

```
com.ibm.ssl.protocol=SSL
com.ibm.ssl.trustManager=SunX509
com.ibm.ssl.keyManager=SunX509
com.ibm.ssl.contextProvider=SunJSSE
com.ibm.ssl.keyStoreType=JKS
com.ibm.ssl.keyStoreProvider=SUN
com.ibm.ssl.keyStore=/home/user1/etc/key.jks
com.ibm.ssl.trustStoreType=JKS
com.ibm.ssl.trustStoreProvider=SUN
com.ibm.ssl.trustStore=/home/user1/etc/trust.jks
```

   For more information, see http://www-01.ibm.com/support/knowledgecenter/SS7JFU_8.0.0/com.ibm.websphere.express.doc/info/exp/ae/tcli_ejbthinclient.html.

5. Create a `.properties` file with file name `jndi.properties` in the local directory and add the following JVM parameters:

   - `com.ibm.SSL.ConfigURL` references a file URL that points to the `ssl.client.props` file.
   - `com.ibm.CORBA.ConfigURL` references a file URL that points to the `sas.client.props` file.
   - `com.ibm.CORBA.Debug.Output` with a value of `NUL`.

   For example:

```
-com.ibm.SSL.ConfigURL="file:///home/user1/ssl.client.props"
-com.ibm.CORBA.ConfigURL="file:///home/user1/sas.client.props"
-com.ibm.CORBA.Debug.Output=NUL
```

6. Expand the created project in the Project Explorer view, drag the local directory to the **lib** folder.
7. Click **Link to files and folders** in the prompted **File and Folder Operation** dialog and press Enter.

   If you want to deploy the application to TIBCO Enterprise Administrator, click **Copy files and folders**.

8. Before running a process, from the **Run** menu, click **Run > Run Configurations**. Select the **Arguments** tab and add the following JVM parameters in the **VM arguments**.

   - `Dcom.ibm.SSL.ConfigURL` references a file URL that points to the `ssl.client.props` file.
   - `Dcom.ibm.CORBA.ConfigURL` references a file URL that points to the `sas.client.props` file.
   - `Dcom.ibm.CORBA.Debug.Output` with a value of `NUL`.

   For example:

```
-Dcom.ibm.SSL.ConfigURL="file:///home/user1/ssl.client.props"
-Dcom.ibm.CORBA.ConfigURL="file:///home/user1/sas.client.props"
-Dcom.ibm.CORBA.Debug.Output=NUL
```

   If you want to deploy the application to TIBCO Enterprise Administrator, navigate to the *TIBCO_HOME*`/bw/6.2/bin` directory, and add the following content in the `bwcommon.tra` file:

```
java.extended.properties=-Dcom.ibm.SSL.ConfigURL="file:///home/user1/
ssl.client.props" -Dcom.ibm.CORBA.ConfigURL="file:///home/user1/
sas.client.props" -Dcom.ibm.CORBA.Debug.Output=NUL
```

Creating an EJB Configuration Shared Resource

An EJB Configuration shared resource is necessary to specify the configuration for the JNDI server before running any EJB activity.

Prerequisites

The EJB Configuration shared resource is available at the Resources level. Before creating an EJB configuration, you must create a new project.

Ensure that you already connected to an EJB server.

Procedure

1. Expand the created project in the Project Explorer view.
2. Right-click the **Resources** folder and click **New > EJB Configuration**.
3. Type a name in the **Resource Name** field in the **EJB** dialog. Click **Finish**.
4. Configure the EJB Configuration shared resource in the displayed editor, and click **Test Connection** to verify the configuration, as described in EJB Configuration Shared Resource

Configuring EJB Client JAR Files

The EJB client JAR files are necessary for the plug-in to be used as the client view of the EJB.

Prerequisites

Before creating an EJB configuration, you must create a new project.

Ensure that you already connected to an EJB server.

Procedure

1. Copy the EJB client JAR files to a local directory.
2. Expand the created project in the Project Explorer view, drag the local directory to the **lib** folder.
3. Click **Link to files and folders** in the prompted **File and Folder Operation** dialog and press Enter.

If you want to deploy the application to TIBCO Enterprise Administrator, click **Copy files and folders**.

Configuring a Process

Processes define the business logic. Once a project is created, you must configure a process by adding activities, conditions, and services.

Prerequisites

Ensure you already finished the following tasks before configuring a process:

- Creating a Project
- Connecting to an EJB Server
- Creating an EJB Configuration Shared Resource
- Configuring EJB Client JAR Files

Procedure

1. Select an activity from the Palette view and drop it in the Process editor.
   For example, click and drop the **EJB2Home** activity from the EJB palette.
2. Click  to create links between the activities and configure the condition types.
3. Configure the added activities, as described in EJB Palette.

   An EJB Configuration shared resource is required when configuring the activities. See Creating an EJB Configuration Shared Resource for more details on how to create the EJB Configuration shared resource.
4. Click **File > Save** to save the project.

Debugging and Running a Process

Debug the application you configured to ensure that the application configuration is correct.

Procedure

1. Open the process you configured in TIBCO Business Studio.
2. On the toolbar, click **Run > Debug Configurations**.
3. Click **BusinessWorks Application > BWApplication** in the left panel.
4. Ensure that only the application you want to debug and run is selected in the **Applications** tab in the right panel.
5. Click the **Advanced** tab and click **Browse** to locate the `logback` file.

   By default, the log file is located in the *`TIBCO_HOME`*`/bw/`*`version_number`*`/config/design/logback` directory and error logs are captured. See Managing Logs for more details.
6. Click **Debug**.
   TIBCO Business Studio changes to the Debug perspective. Logs are displayed in the Console view.

Checking Output of an Activity

After debugging an application, you can check the output of activities.

Procedure

1. In the Debug perspective, expand **BWApplication** and click the activity in the upper left panel.
2. In the upper right panel, click the **Job Data** view and click **Output**.

You can also check the activity output in the plug-in logs. See Managing Logs for more information.

Deploying Applications

After deploying applications, you can manage BusinessWorks applications by using TIBCO Enterprise Administrator.

Prerequisites

The following tasks are required before deploying applications:

- Creating a Project
- Generating an EAR File

A typical workflow of deployment includes:

1. Uploading an EAR file.
2. Deploying an application.
3. Configuring an application.
4. Starting an application.

To deploy an application EAR file, use the command-line mode with the **`bwadmin`** utility. See *TIBCO ActiveMatrix BusinessWorks Administration* for more details about how to deploy an application.

Generating an EAR File

Application archives are enterprise archive (EAR) files that are created in TIBCO Business Studio. An EAR file is required when deploying an application.

Prerequisites

An application project was created, as described in Creating a Project.

There are many ways to generate an EAR file, the following is one of the methods. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information.

Procedure

1. Go to the File Explorer view and click the **Open Directory to Browse** icon.
2. Select the folder where you want to generate the EAR file and click **OK**.
   The new folder is displayed in the File Explorer view.
3. Drag the application from the Project Explorer view to the new folder in the File Explorer view.

   The EAR file is generated with the name `<name>.<application>_<version>.ear`.

EJB Configuration Shared Resource

Use an EJB Configuration shared resource to specify the connection configuration for the JNDI server.

General

Field	Module Property?	Description
Package	No	Name of the package where the new shared resource is added.
Name	No	The name to be displayed as the label for the shared resource.
Description	No	Short description of the shared resource.

Advance Configuration

Field	Module Property?	Description
Use Shared JNDI Configuration	No	When this check box is selected, the **JNDI Configuration** field is displayed. You can use it to choose a JNDI Configuration resource. When this check box is clear, the **JNDI Context Factory**, **JNDI Context URL**, **JNDI User Name**, and **JNDI Password** fields are displayed.
JNDI Configuration	No	Specifies a JNDI Configuration resource.  This field is displayed only when the **Use Shared JNDI Configuration** check box is selected.
JNDI Context Factory	Yes	Select an initial context factory supplied by each supported application server vendor from the list. Different initial context factories other than the default options are available in this list: • **org.jnp.interfaces.NamingContextFactory** JBoss Application Server version 6.x • **org.jboss.naming.remote.client.InitialContextFactory** JBoss Application Server version 7.1.x and WildFly 8.x • **weblogic.jndi.WLInitialContextFactory** Oracle WebLogic Server 12.x • **com.ibm.websphere.naming.WsnInitialContextFactory** IBM WebSphere Application Server 8.x • **Others (Unsupported)** For other application servers  This field is displayed only when the **Use Shared JNDI Configuration** check box is clear.

Field	Module Property?	Description
JNDI Context URL	Yes	Specify the URL of the server. See related JNDI provider documentation for the syntax of the URL.  This field is displayed only when the **Use Shared JNDI Configuration** check box is clear.
JNDI User Name	Yes	Specify the user name to use when logging into the JNDI server. If the JNDI provider does not require access control, this field can be empty.  This field is displayed only when the **Use Shared JNDI Configuration** check box is clear.
JNDI Password	Yes	Specify the password to use when logging into the JNDI server. If the JNDI provider does not require access control, this field can be empty.  This field is displayed only when the **Use Shared JNDI Configuration** check box is clear.

EJB Configuration

Field	Module Property?	Description
Max Connections	Yes	Maximum number of naming contexts that are created and cached in the connection pool. See Pooling and Caching for more information.  If this field is set to zero, the naming context is not cached, and a new context is created for each lookup operation.
Connection Retries	Yes	Maximum number of attempts can be made to connect to the application server, or to create the naming context.  If this field is set to zero, only one attempt can be made to establish a connection.
Retry Interval (ms)	Yes	The time interval, in milliseconds, of making each connection attempt to the application server, or to create the naming context.  If this field is set to zero, the new connection attempt is started immediately.
Test Connection	No	Press the button to test the connection configuration for the JNDI server.

Pooling and Caching

Creating an InitialContext class requires a large amount of overhead. Therefore, contexts can be cached and placed in a pool to improve performance over time. Define the **Max Connections** field to specify the maximum number of InitialContext that the plug-in creates at any given time.

The InitialContext is created when you start a process during initializing the EJB2Home activity or EJB3Remote activity, and all created contexts are placed into the pool. When you want to use a context, fetch an existing context from the pool to obtain a reference to the EJB 2.x home object or EJB 3.x remote object. Once the EJB 2.x home object or EJB 3.x remote object is obtained, the context is released back into the pool. If all contexts are being used, TIBCO ActiveMatrix BusinessWorks Plug-in for EJB blocks any new requests until a context is freed from the pool.

All contexts are cached for reuse by subsequent process instances. If you specify zero in the **Max Connections** field, contexts are not cached, and each request creates a new InitalContext.

If a context becomes stale (for example, the server is restarted), TIBCO ActiveMatrix BusinessWorks Plug-in for EJB attempts to create a new context to replace the stale context in the pool. Use the **Connection Retries** field to define the maximum number of attempts to reestablish the context.

EJB Palette

The EJB palette contains activities that can be added to your business processes.

The EJB palette consists of three activities:

- EJB2Home

  This activity connects to an EJB server for EJB 2.x (specified by an EJB Configuration shared resource), performs a JNDI lookup operation to obtain a reference to a home object, and obtains a reference to a remote object from the home object. Use this activity to invoke any method defined by the home object.

- EJB2Remote

  This activity invokes remote methods on the remote object obtained by the EJB2Home activity from an EJB server for EJB 2.x. This activity must be placed after an EJB2Home activity in a process definition.

- EJB3Remote

  This activity connects to an EJB server for EJB 3.x (specified by an EJB Configuration shared resource), performs a JNDI lookup operation to obtain a reference to a remote object, and invokes remote methods.

EJB2Home

The EJB2Home activity connects to an EJB server for EJB 2.x, performs a JNDI lookup operation to obtain a reference to a home object, and obtains a reference to a remote object from the home object. Use this activity to invoke any method defined by the home object.

You can also use this activity to invoke a method on the EJB home. Once this activity is executed, use the EJB2Remote activity to invoke methods on the remote object without performing additional remote lookups.

It is optional to cache the home object and the stateless remote objects to reuse them across process instances. However, this improves the performance because further remote lookups are not necessary.

General

The **General** tab contains the following fields:

Field	Module Property?	Description
Name	No	Name to be displayed as the label for the activity in the process.
EJB Connection	Yes	An EJB Configuration shared resource defines a set of relationships and their participating entities. Click  to select an EJB Configuration shared resource. See Creating an EJB Connection Shared Resource for detailed instructions.
JNDI Name	Yes	The name registered with the JNDI server for the EJB.

Field	Module Property?	Description
Home Interface Class	No	Name of the home interface for the EJB (the interface which extends javax.ejb.EJBHome). Click  to display the available classes for enterprise beans.
Home Interface Method	No	Method of the home interface to call. The list provides the methods contained in the selected interface class in the **Home Interface Class** field.

Description

The **Description** tab provides a short description for the activity.

Advanced

The **Advanced** tab contains the following fields:

Field	Module Property?	Description
Cache Home Object	No	Specifies whether the home object is to be cached for use by process instances. If unchecked, each process instance performs a JNDI lookup to obtain the home object reference. If checked, process instances reuse the cached home object reference. Choosing to cache the home or remote object improves performance over time. However, the home or remote object might become "stale". That is, the object changes on the EJB server, and the cached object no longer matches the object on the server. If the EJB2Home activity encounters a stale home object in the cache, the activity attempts to recreate the home object. The Connection Retries field of the EJB Configuration shared resource specifies the number of retries that the EJB2Home activity will attempt before failing to recreate the home object. If the EJB2Home activity is successful in recreating the home object, the activity succeeds and a new home object is placed in the cache. If the activity cannot recreate the home object after the maximum number of retries, the activity fails and takes the error transition.
Cache Stateless Remote Object	No	Specifies whether the remote object is to be cached for use by process instances. The remote object is only cached for stateless session enterprise beans. If unchecked, each process instance obtains a new remote object reference. If checked, process instances reuse the cached remote object reference. If the process instance encounters a stale cached remote object, the object is removed from the cache, an error is returned, and the activity fails. However, subsequent process instances will attempt to recreate the remote object.

Input

The following is the input for the activity:

Input Item	Data Type	Description
MethodParameters	complex	An object containing the parameters required by the home interface method (shown only if the method is selected in the **General** tab requires parameters).

Output

The following is the output for the activity:

Output Item	Data Type	Description
MethodReturnValue	complex	The value returned by the home interface method. Shown only if the home interface method selected in the **General** tab returns a value.

Fault

The **Fault** tab lists exceptions that are thrown by this activity:

Error Schema Element	Data Type	Description
errorCode	string	The error code returned by the plug-in.
errorMessage	string	The error message returned by the plug-in.
errorStackTrace	string	The complete stack trace that causes the exception.
exceptionClassName	string	The class name of the root exception that causes the exception.

EJB2Remote

The EJB2Remote activity invokes remote methods on the remote object from an EJB server for EJB 2.x. This activity must be placed after an EJB2Home activity in a process definition.

General

The **General** tab contains the following fields:

Field	Module Property?	Description
Name	No	Name to be displayed as the label for the activity in the process.

Field	Module Property?	Description
Home Activity	No	The EJB2Home activity in the process definition that obtained the reference to the remote object. The list only supplies EJB2Home activities that obtain references to remote objects. EJB2Home activities that do not create remote objects are not listed.
Remote Interface Class	No	Name of the remote interface (the interface which extends javax.ejb.EJBObject) obtained by the EJB2Home activity. This field is populated automatically based on the selected EJB2Home activity.
Remote Interface Method	No	Method of the remote interface to call. The list provides a list of public methods contained in the remote interface class.

Description

The **Description** tab provides a short description for the activity.

Advanced

The **Advanced** tab contains the following fields:

Field	Module Property?	Description
Release Remote Object	No	If this field is selected, the remote object is released after this activity is completed. If this field is clear, the remote object is used by subsequent EJB2Remote activities within this process definition. This field is only meaningful for stateful session and entity beans. Cached remote stateless session beans are never released when this field is checked. This field is useful if you want to release the memory used by the stateful session and entity bean before continuing with the remainder of the process definition.  If you are invoking a method on a stateful session or entity bean, only check this field when no subsequent activities in the process definition invoke the remote object. If the object is released and a subsequent invocation is made, an error is returned.

Input

The following is the input for the activity:

Input Item	Data Type	Description
MethodParameters	complex	An object containing the parameters required by the remote interface method (shown only if the method is selected in the **General** tab requires parameters).

Output

The following is the output for the activity:

Output Item	Data Type	Description
MethodReturnValue	complex	The value returned by the remote interface method. Shown only if the remote interface method selected in the **General** tab returns a value.

Fault

The **Fault** tab lists exceptions that are thrown by this activity:

Error Schema Element	Data Type	Description
errorCode	string	The error code returned by the plug-in.
errorMessage	string	The error message returned by the plug-in.
errorStackTrace	string	The complete stack trace that causes the exception.
exceptionClassName	string	The class name of the root exception that causes the exception.

EJB3Remote

The EJB3Remote activity connects to an EJB server for EJB 3.x, performs a JNDI lookup operation to obtain a reference to the remote object, and invokes remote methods.

General

The **General** tab contains the following fields:

Field	Module Property?	Description
Name	No	Name to be displayed as the label for the activity in the process.
Use Cached Remote Object	No	When selected, the **Cache From** field is displayed, then you can choose an existing remote object. When cleared, the **EJB Configuration** and **JNDI Name** fields are displayed.

Field	Module Property?	Description
Cache From	No	Specifies which remote object you want to reuse.  This field is displayed only when the **Use Cached Remote Object** check box is selected.
EJB Configuration	Yes	An EJB Configuration shared resource defines a set of relationships and their participating entities. Click  to select an EJB Configuration shared resource.  This field is displayed only when the **Use Cached Remote Object** check box is clear.
JNDI Name	Yes	The name registered with the JNDI server for the EJB.  This field is displayed only when the **Use Cached Remote Object** check box is clear.
Remote Interface Class	No	Name of the Remote interface for EJB 3.x, the interface name is with the annotation @Remote (javax.ejb.Remote) Click  to display the available classes for enterprise beans.
Remote Interface Method	No	The remote interface method to call. The list provides the methods contained in the selected interface class in the **Remote Interface Class** field.

Description

The **Description** tab provides a short description for the activity.

Advanced

The **Advanced** tab contains the following fields:

Name	Module Property?	Description
Release Remote	No	When selected, the remote object is released after this activity is completed. When clear, the remote object is put in cache pool after this activity is completed, therefore you can reuse this remote object in the other EJB3Remote activities.

Input

The following is the input for the activity:

Input Item	Data Type	Description
MethodParameters	complex	An object containing the parameters required by the remote interface method (shown only if the method is selected in the **General** tab requires parameters).

Output

The following is the output for the activity:

Output Item	Data Type	Description
MethodReturnValue	complex	The value returned by the remote interface method. Shown only if the remote interface method selected in the **General** tab returns a value.

Fault

The **Fault** tab lists exceptions that are thrown by this activity:

Error Schema Element	Data Type	Description
errorCode	string	The error code returned by the plug-in.
errorMessage	string	The error message returned by the plug-in.
errorStackTrace	string	The complete stack trace that causes the exception.
exceptionClassName	string	The class name of the root exception that causes the exception.

Working with Sample Project

Working through the sample projects helps you understand how TIBCO ActiveMatrix BusinessWorks Plug-in for EJB works.

TIBCO ActiveMatrix BusinessWorks Plug-in for EJB packages one sample project within the installer. The sample project is located in the *TIBCO_HOME*/bw/palettes/ejb/*version_number*/Samples/ directory.

The Examples Project, with its six process, shows the basic usage of activities of TIBCO ActiveMatrix BusinessWorks Plug-in for EJB. This project uses JBoss Application Server version 7 as the EJB container.

Importing Sample Project

Before running the project, you must import the sample project to TIBCO Business Studio.

Prerequisites

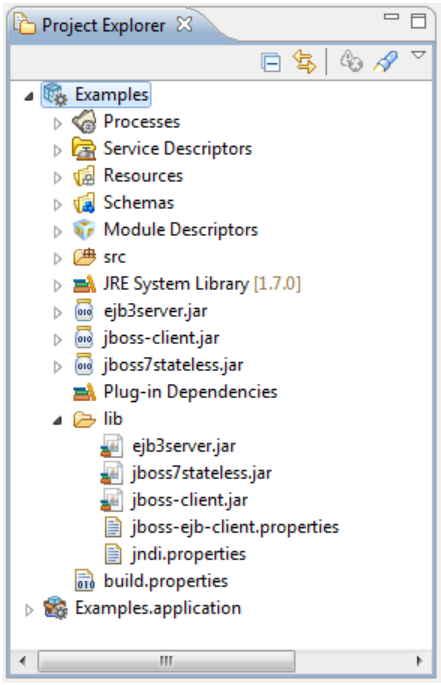
- Ensure you have started a JBoss Application Server version 7.1.x.
- Ensure that the `.jar` files, `ejb3server.jar` and `jboss7stateless.jar`, in the *TIBCO_HOME*/bw/palettes/ejb/*version_number*/Samples/Examples/Examples/lib are deployed to your JBoss Application Server.

Procedure

1. Start TIBCO Business Studio.
2. Click **File > Import**.
3. In the **Import** dialog, expand the **General** folder and select the **Existing Studio Projects into Workspace** item. Click **Next.**
4. Click **Browse** next to the **Select root directory** field to locate the samples. Click **Finish.**
   The sample project is located in the *TIBCO_HOME*/bw/palettes/ejb/*version_number*/Samples/ directory.
5. In the Project Explorer view, expand **lib**. Double-click **jboss-ejb-client.properties** and **jndi.properties** and update the content according to your JBoss Application Server.
6. In the Project Explorer view, click **Examples > Resources > test**, double-click **NewEJBResource.ejbResource**. Configure the EJB Configuration shared resource in the displayed editor according to your JBoss Application Server, and click **Test Connection** to verify the configuration.
   The expected message is `Test Connection Successfully`.

Result

The sample project is imported to TIBCO Business Studio.



Working with the Examples Project

This project uses the activities of TIBCO ActiveMatrix BusinessWorks Plug-in for EJB to show six basic process examples to finish various tasks.

Prerequisites

Ensure that you have imported the sample project, as described in Importing Sample Projects.

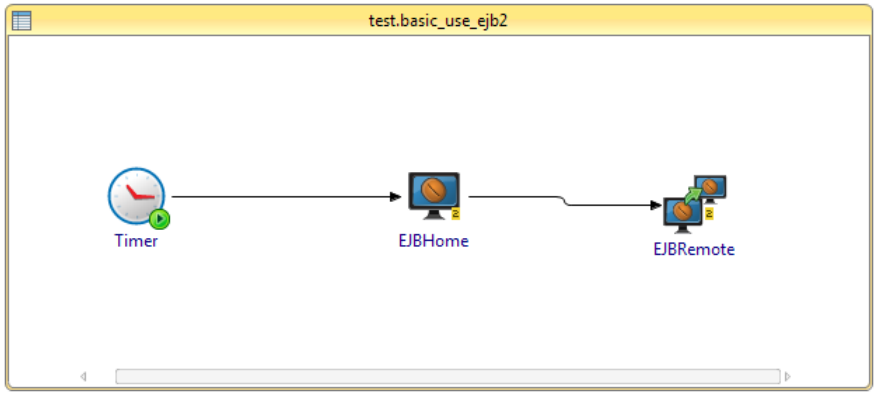
In the Examples project, there is one package **test**, in which six processes are located.

Procedure

1. Expand the Examples project in the Project Explorer view.
2. Click **Processes > test**.
3. Click **Run > Run Configurations**.
4. Expand **BusinessWorks Application** in the **Run Configurations** dialog and click **BWApplication**.
5. Ensure that only **Examples.application** is selected in the **Applications** tab.
6. Click **Run**.
7. Click the **Terminate** icon to stop the process.

Configurations for basic_use_ejb2

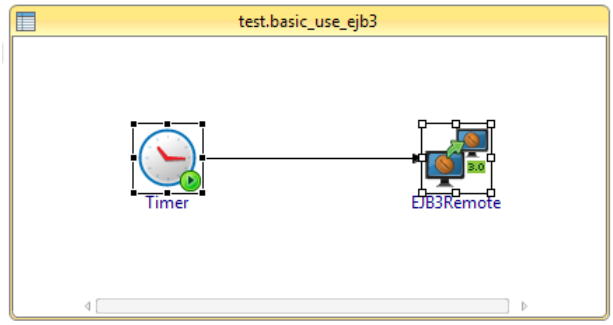
The basic_use_ejb2 process shows how to perform a lookup of an EJB 2.x home object, create a remote object and invoke a remote method.



Activity	Description
Timer	This activity starts the process at a specific time.
EJBHome	This activity performs a JNDI lookup in the **test.NewEJBResource** EJB Configuration shared resource and creates an EJB 2.x remote object.
EJBRemote	This activity invokes a remote method on the remote object that is created by the EJBHome activity.

Configurations for basic_use_ejb3

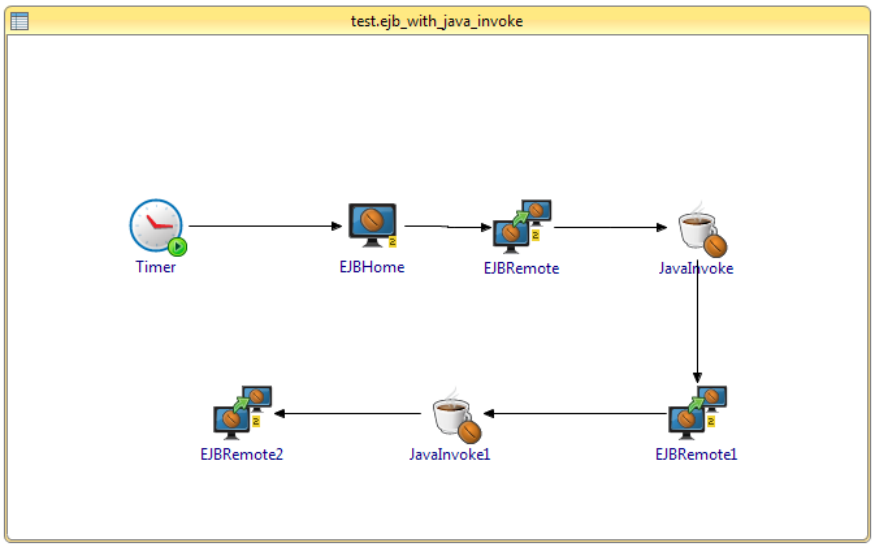
The basic_use_ejb3 process shows how to perform a lookup of an EJB 3.x remote object and invoke a remote method.



Activity	Description
Timer	This activity starts the process at a specific time.
EJB3Remote	This activity performs a JNDI lookup in the **test.NewEJBResource** EJB Configuration shared resource and creates an EJB 3.x remote object, and then invokes a remote method.

Configurations for ejb_with_java_invoke

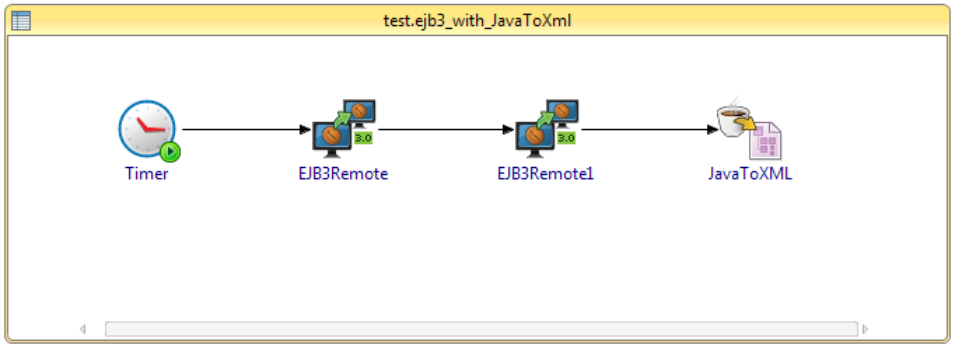
The ejb_with_java_invoke process shows how to use the EJB activities with the Java Invoke activities, to add, update and delete data in a catalogue.



Activity	Description
Timer	This activity starts the process at a specific time.
EJBHome	This activity performs a JNDI lookup in the **test.NewEJBResource** EJB Configuration shared resource and create an EJB 2.x remote object.
EJBRemote	This activity invokes the addCatalogue() method on the remote object which is created by the EJBHome activity. As a result, a record is added in the catalogue.
JavaInvoke	This activity invokes a Java class method to check if the catalogue record is added after the EJBRemote activity.
EJBRemote1	This activity invokes the updateCataloguePrice() method on the remote object that is created by the EJBHome activity. As a result, a record is updated in the catalogue.
JavaInvoke1	This activity invokes a Java class method to check if the catalogue record is updated after the EJBRemote1 activity.
EJBRemote2	This activity invokes the deleteCataloguePrice() method on the remote object that is created by the EJBHome activity. As a result, a record is deleted in the catalogue.

Configurations for ejb3_with_JavaToXml

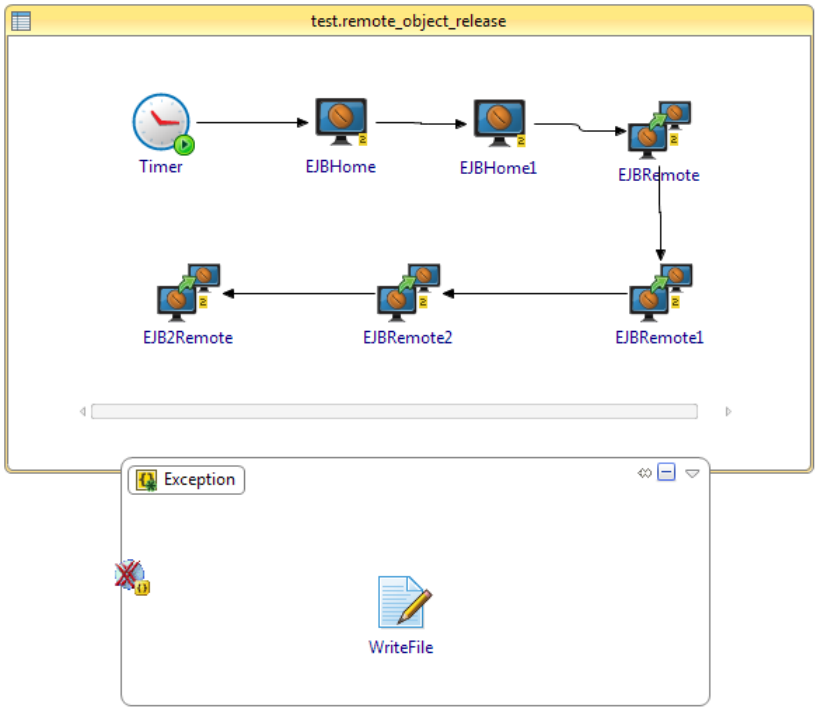
The ejb3_with_JavaToXml process shows how to use EJB activities with a JavaToXML activity to convert an EJB 3.x remote object to XML format.



Activity	Description
Timer	This activity starts the process at a specific time.
EJB3Remote	This activity performs a JNDI lookup in the **test.NewEJBResource** EJB Configuration shared resource and creates an EJB 3.x remote object, and then invokes a remote method.
EJB3Remote1	This activity receives the cached EJB 3.x object from the EJB3Remote activity and invokes another remote method.
Java To XML	This activity converts the data of the EJB 3.x object into an XML document.

Configurations for remote_object_release

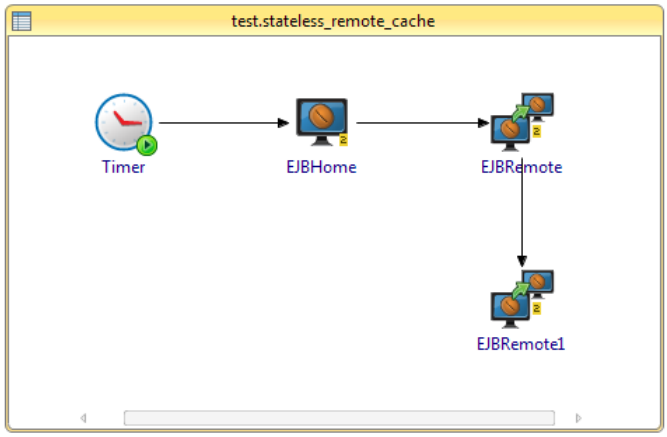
The remote_object_release process shows how to release EJB 2.x remote objects in one process instance.



Activity	Description
Timer	This activity starts the process at a specific time.
EJBHome	This activity performs a JNDI lookup in the **test.NewEJBResource** EJB Configuration shared resource and create an EJB 2.x remote object.
EJBHome1	This activity performs a JNDI lookup in the **test.NewEJBResource** EJB Configuration shared resource and create another EJB 2.x remote object.
EJBRemote	This activity invokes a remote method on the remote object which is created by the EJBHome activity. The **Release Remote Object** check box is selected in **Advance** tab of this activity.
EJBRemote1	is activity invokes a remote method on the remote object which is created by the EJBHome1 activity. The **Release Remote Object** check box is clear in **Advance** tab of this activity.
EJBRemote2	This activity invokes a remote method on the remote object which is created by the EJBHome activity. As the EJB 2.x object was already released after the EJBRemote activity is completed, a null pointer exception occurs.
EJB2Remote	This activity invokes a remote method on the remote object which is created by the EJBHome1 activity.
Write File	This activity writes the exception details to the specified file.

Configurations for stateless_remote_cache

The stateless_remote_cache process shows how to keep an EJB 2.x remote object in cache, when handling a stateless session bean.



Activity	Description
Timer	This activity starts the process at a specific time.

Activity	Description
EJBHome	This activity uses the **test.NewEJBResource** EJB Configuration shared resource and performs a JNDI lookup of an EJB 2.x remote object. The **Cache Stateless Remote Object** check box is selected in **Advance** tab of this activity.
EJBRemote	This activity invokes a remote method on the remote object which is created by the EJBHome activity. The **Release Remote Object** check box is selected in **Advance** tab of this activity.
EJBRemote1	This activity invokes a remote method on the remote object which is created by the EJBHome activity.

Managing Logs

Logs are used to trace and troubleshoot plug-in exceptions.

A `logback.xml` file is located in the *`TIBCO_HOME`*`/bw/`*`version_number`*`/config/design/logback` directory. Update this file to set up a log level and export logs to a file.

Log Levels

The plug-in captures logs at different levels.

Log Level	Description
Trace	Includes all the information regarding the running process.
Debug	Indicates a developer-defined tracing message.
Info	Indicates normal plug-in operations. No action is required. A tracing message tagged with Info indicates that a significant processing step was reached and logged for tracking or auditing purposes. Only info messages preceding a tracking identifier are considered as significant steps.
Warn	Indicates that an abnormal condition is found. Processing continues, but special attention from an administrator is recommended.
Error	Indicates that an unrecoverable error has occurred. Depending on the severity of the error, the plug-in continues with the next operation or stops altogether.

Setting Up Log Levels

By default, the log level is `Error` . You can use the plug-in to change the log level to trace different messages.

If neither the plug-in log nor the BusinessWorks log is configured in the `logback.xml` file, the error logs of the plug-in are displayed in the Console view by default.

If the plug-in log is not configured but the BusinessWorks log is configured in the `logback.xml` file, the configuration for the BusinessWorks log is implemented by the plug-in.

Procedure

1. Navigate to the *`TIBCO_HOME`*`/bw/`*`version_number`*`/config/design/logback` directory and open the `logback.xml` file.
2. Add the following node in the User loggers area to specify the log level for the plug-in.
   ```
   <logger name="com.tibco.bw.palette.bwpluginejb.runtime">
      <level value="DEBUG"/>
   </logger>
   ```
   The `level` tag defines the log level and the value is `Error` or `Debug`.

   When the `level` is set to `Debug`, the input and output for the plug-in activities are also displayed in the Console view. See Log Levels for more details regarding each log level.

3. Optional: Add the following node in User loggers area to specify the log level for an activity.

```
<logger name="com.tibco.bw.palette.bwpluginejb.runtime.ActivityNameActivity">
   <level value="DEBUG"/>
</logger>
```

   For example, if you want to set the log level of the EJB2Home activity to `Debug`, add the following node:

```
<logger name="com.tibco.bw.palette.bwpluginejb.runtime.EJBHomeActivity">
   <level value="DEBUG"/>
</logger>
```

   The activities that do not configure with specific log levels, still inherit log level configured for the plug-in or BusinessWorks.

4. Save the file.

Exporting Logs to a File

Modify the `logback.xml` file to export plug-in logs to a file.

Procedure

1. Navigate to the *TIBCO_HOME*`/bw/`*version_number*`/config/design/logback` directory and open the `logback.xml` file.

   When deploying an application in TIBCO Enterprise Administrator, navigate to the *TIBCO_HOME*`/bw/domains/mydomain/appnodes/myspace/mynode` directory to find the `logback.xml` file.

2. Add the following node to specify the file location.

```
<appender name="FILE" class="ch.qos.logback.core.FileAppender">
   <file>c:/bw6-bwpluginejb.log</file>
      <encoder>
        <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36}-%msg%n</pattern>
      </encoder>
</appender>
```

   The `file` tag defines the location to which the log is exported, and the value is the absolute path of the file.

   Add also the file name in the file path.

3. Add the following node to the `root` node at the bottom of the `logback.xml` file to enable exporting the logs to a file.

```
    <appender-ref ref="FILE" />
```

```
<root level="DEBUG">
   <appender-ref ref="STDOUT" />
   <appender-ref ref="FILE" />
</root>
```

4. Save the file.

Error Codes

The exceptions that are thrown by the plug-in are listed with corresponding descriptions and resolutions.

Error Code and Error Message	Role	Category	Description	Resolution
TIBCO-BW-PALETTE-EJB-500001 `{0}`	errorRole	BW_Plugin	The general information of ERROR messages.	No action.
TIBCO-BW-PALETTE-EJB-500002 `Class {0} can not be loaded in app class path.`	errorRole	BW_Plugin	Failed to load the specified class in app class path.	Check if the class jar file is added into the classpath of this project.
TIBCO-BW-PALETTE-EJB-500003 `Exception occurred while {0}.`	errorRole	BW_Plugin	An exception occurred when initializing JNDI server.	Check for the exception error message.
TIBCO-BW-PALETTE-EJB-500004 `Unknown exception occurred while {0}.`	errorRole	BW_Plugin	An unknown exception occurred.	Check for the exception error message.
TIBCO-BW-PALETTE-EJB-500005 `Method {0} can not be found in class {1}.`	errorRole	BW_Plugin	Failed to find the specified method in the class.	Check if the invoked method has been declared in the class file.
TIBCO-BW-PALETTE-EJB-500006 `Security exception while get method {0} from class {1}.`	errorRole	BW_Plugin	An security exception occurred while receiving the specified method from the class.	Check the permission for communication.
TIBCO-BW-PALETTE-EJB-500007 `Class cast exception occurred, the look up object can not be cast to EJBHome object.`	errorRole	BW_Plugin	A class cast exception occurred while casting the look up object to the EJB home object.	Check if the JNDI name is correct, the lookup result is not an EJB home reference.

Error Code and Error Message	Role	Category	Description	Resolution
TIBCO-BW-PALETTE-EJB-500008 `IO exception occurred while {0}.`	errorRole	BW_Plugin	An IO exception occurred.	Check the IO channel.
TIBCO-BW-PALETTE-EJB-500009 `{0} {1}, Invalid beantype. Beantype= {2}.`	errorRole	BW_Plugin	An error occurred when BeanType was invalid.	Check in the EJB container if the BeanType has been specified of the EJB.
TIBCO-BW-PALETTE-EJB-500010 `{0} {1}, remote exception occurred while {2}.`	errorRole	BW_Plugin	A remote exception occurred.	Check the error message on the application server.
TIBCO-BW-PALETTE-EJB-500011 `{0} {1}, illegal access exception while invoke method {2} in class {3}.`	errorRole	BW_Plugin	An illegal access exception occurred while invoking the method in the specified class.	Check the access permission.
TIBCO-BW-PALETTE-EJB-500012 `{0} {1}, illegal argument exception while invoke method {2} in class {3}.`	errorRole	BW_Plugin	An illegal argument exception occurred while invoking the method in the specified class.	Specify the input parameters to match the requirements of the invoked method.
TIBCO-BW-PALETTE-EJB-500013 `{0} {1}, invocation target exception while invoke method {2} in class {3}.`	errorRole	BW_Plugin	An invocation target exception occurred while invoking the method in the specified class.	Check the error message on the application server.
TIBCO-BW-PALETTE-EJB-500014 `{0} {1}, remote object can not be found in checked in object.`	errorRole	BW_Plugin	Failed to find the remote object in the checked-in object.	Check if the JNDI name is correct and perform the lookup again.

Error Code and Error Message	Role	Category	Description	Resolution
TIBCO-BW-PALETTE-EJB-500015 `{0}, checked in object can not be found in process job resource.`	errorRole	BW_Plugin	Failed to find the checked-in object in the process job resource.	Perform the lookup again because the object has been released from the process context.