TIBCO EBX®
Installation Guide

Version 6.0.0
March 2021

TIBCO®

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ANY SOFTWARE ITEM IDENTIFIED AS THIRD PARTY LIBRARY IS AVAILABLE UNDER SEPARATE SOFTWARE LICENSE TERMS AND IS NOT PART OF A TIBCO PRODUCT. AS SUCH, THESE SOFTWARE ITEMS ARE NOT COVERED BY THE TERMS OF YOUR AGREEMENT WITH TIBCO, INCLUDING ANY TERMS CONCERNING SUPPORT, MAINTENANCE, WARRANTIES, AND INDEMNITIES. DOWNLOAD AND USE OF THESE ITEMS IS SOLELY AT YOUR OWN DISCRETION AND SUBJECT TO THE LICENSE TERMS APPLICABLE TO THEM. BY PROCEEDING TO DOWNLOAD, INSTALL OR USE ANY OF THESE ITEMS, YOU ACKNOWLEDGE THE FOREGOING DISTINCTIONS BETWEEN THESE ITEMS AND TIBCO PRODUCTS.

TIBCO and TIBCO EBX are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. Please see the readme.txt file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (*https://www.tibco.com/patents*) for details.

Table of contents

CHAPTER 1

Supported environments

This chapter contains the following topics:

1.1 Browsing environment

Supported web browsers

The TIBCO EBX® web interface supports the following browsers:

Microsoft Edge chromium	As Microsoft Edge chromium is updated frequently and it is not possible to deactivate automatic updates, TIBCO Software Inc. only tests and makes the best effort to support the latest version available.
Microsoft Edge	Minimum supported version is 44 Compatibility mode is not supported.
Microsoft Internet Explorer 10, 11	Compatibility mode is not supported. **Performance limitations:** page loading with IE10 and IE11 is two times slower. This issue is observed when forms have many input components, and particularly many multi-occurrence groups. **Graphical layout:** graphical rendering in IE10 and IE11 can slightly differ from other browsers (for example, the alignment of some labels, icons and other components can be off by a few pixels).
Mozilla Firefox ESR 68 (see details)	As Mozilla Firefox is updated frequently, TIBCO Software Inc. only fully supports version ESR 68. See Mozilla Firefox ESR for more details.
Google Chrome	As Google Chrome is updated frequently and it is not possible to deactivate automatic updates, TIBCO Software Inc. only tests and makes the best effort to support the latest version available.

Screen resolution

The minimum screen resolution for EBX® is 1024x768.

Refreshing pages

Browser page refresh is not supported by EBX®. When a page refresh is performed, the last user action is re-executed, and therefore could result in potential issues. It is thus imperative to use the action buttons and links offered by EBX® instead of refreshing the page.

"Previous" and "Next" buttons

Browser's previous and next page buttons are not supported by EBX®. When navigating through page history, an obsolete user action is re-executed, and therefore could result in potential issues. It is thus imperative to use the action buttons and links offered by EBX® rather than browser's buttons.

Browser configuration

The following features must be activated in the browser configuration for the user interface to work properly:

- JavaScript
- Ajax
- Pop-ups

Attention

Avoid using any browser extensions or plug-ins, as they could interfere with the proper functioning of EBX®.

Limitations

Some browsers may have a limitation on the number of iframes that can be embedded. If this is the case, it limits to the number of items that can be pushed in the breadcrumb. Please check the browser documentation for more details.

1.2 Supported application servers

EBX® supports the following configurations:

- Java Runtime Environment: JRE 8 or 11, which necessarily includes the limitations specified by the Java Virtual Machine implementation vendor. For example, for JRE and JDK 8, Oracle states that they are "not updated with the latest security patches and are not recommended for use in production". See Oracle Java Archive site .
- Any Java application server that complies with the Servlet 3.0 up to 5.0 except, for example Tomcat 7.0 up to 10.0 except, WebSphere Application Server 8.5.5 or higher, WebLogic Application Server 14c or higher, JBoss EAP 6.0 or higher. See Java EE deployment overview [p 21].
- The application server must support the JSON Processing 1.1 (JSR 374) or allows the uses of the one embedded in the `ebx.jar` library. For example Tomcat does not provide any libraries to support this specification (only the embedded one can be use), WebSphere Application Server allows to reverse the classloading system (making the embedded one a priority), WebLogic Application Server 14c or higher supports this specification, JBoss EAP allows to include or exclude the available libraries.
- The application server must use UTF-8 encoding for HTTP query strings from EBX®. This can be set at the application server level.

  For example, on Tomcat, you can set the server to always use the UTF-8 encoding, by setting `URIEncoding` to 'UTF-8' on the `<Connector>` in the `server.xml` configuration file. Alternatively,

you can set the server to use the encoding of the request body by setting the parameter `useBodyEncodingForURI` to 'true' in `server.xml`.

Attention

- Limitations apply regarding clustering and hot deployment/undeployment:

  Clustering: EBX® does not include a cache synchronization mechanism, thus it cannot be deployed into a cluster of active instances. See *Technical architecture* for more information.

  Hot deployment/undeployment: EBX® does not support hot deployment/undeployment of web applications registered as EBX® modules, or of EBX® built-in web applications.
- WebSphere Application Server's Java SDKs under version 8.0.4.10 are incompatible with the embedded Apache Calcite 1.26.0 third-party library. It is highly recommended to use the latest Java SDK available and compatible with the application server.

1.3 Supported databases

The EBX® repository supports the Relational Database Management Systems listed below; with suitable JDBC drivers. It is important to follow the database vendor recommendations and update policies regarding the database itself, as well as the JDBC driver.

Oracle Database 12c or higher.	The distinction of `null` values encounters certain limitations. On simple `xs:string` elements, Oracle does not support the distinction between empty strings and `null` values. See *Empty string management* for more information. The user with which EBX® connects to the database requires the following privileges: • CREATE SESSION, • CREATE TABLE, • ALTER SESSION, • CREATE SEQUENCE, • A non-null quota on its default tablespace.
PostgreSQL 9.6 or higher.	When using PostgreSQL as the underlying database, a request fetch size must be set, otherwise the JDBC driver will fetch the whole result set at once. This could lead to an `OutOfMemoryError` when retrieving large amounts of data. See `Request.setFetchSize`[API]. The user with which EBX® connects to the database needs the CONNECT privilege on the database hosting the EBX® repository. Beyond this, the default privileges on the public schema of this database are suitable.
Amazon Aurora PostgreSQL 2.3 (compatible with PostgreSQL 10.7) or higher.	The comments in the above section for PostgreSQL apply.
SAP HANA Database 2.0 or Higher.	When using SAP HANA Database as the underlying database, certain schema evolutions are not supported. It is, for example, impossible to reduce the length of a column; this is a limitation of HANA, as mentioned in the SQL reference guide: "For row table, only increasing the size of VARCHAR and NVARCHAR type column is allowed."
Microsoft SQL Server 2012 SP4 or higher.	When used with Microsoft SQL Server, EBX® uses the default database collation to compare and sort strings stored in the database. This applies to strings used in the data model definition, as well as data stored in relational and history tables. The default database collation can be specified when

	the database is created. Otherwise, the database engine server collation is used. To avoid naming conflicts or unexpected behaviors, a case- and accent-sensitive collation as the default database collation must be used (the collation name is suffixed by "CS_AS" or the collation is binary). The default setting to enforce transaction isolation on SQL Server follows a pessimistic model. Rows are locked to prevent any read/write concurrent accesses. This may cause liveliness issues for mapped tables (history or relational). To avoid such issues, it is recommended to activate snapshot isolation on your SQL Server database. The user with which EBX® connects to the database requires the following privileges: • CONNECT, SELECT and CREATE TABLE on the database hosting the EBX® repository, • ALTER, CONTROL, UPDATE, INSERT, DELETE on its default schema.
Microsoft Azure SQL Database	EBX® has been qualified on Microsoft Azure SQL Database v12 (12.00.700), and is regularly tested to verify compatibility with the current version of the Azure database service. When used with Microsoft Azure SQL, EBX® uses the default database collation to compare and sort strings stored in the database. This applies to strings used in the data model definition, as well as data stored in relational and history tables. The default database collation can be specified when the database is created. Otherwise, the database engine server collation is used. To avoid naming conflicts or unexpected behaviors, a case- and accent-sensitive collation as the default database collation must be used (the collation name is suffixed by "CS_AS" or the collation is binary). The user with which EBX® connects to the database requires the following privileges: • CONNECT, SELECT and CREATE TABLE on the database hosting the EBX® repository, • ALTER, CONTROL, UPDATE, INSERT, DELETE on its default schema.
H2 v1.4.196 or higher.	H2 is not supported for production environments. The default H2 database settings do not allow consistent reads when records are modified.

For other relational databases, please contact the Support team at https://support.tibco.com.

Attention

In order to guarantee the integrity of the EBX® repository, it is strictly forbidden to perform direct modifications to the database (for example, using direct SQL writes).

See also

Repository administration

Data source of the EBX® repository [p 19]

Configuring the EBX® repository [p 45]

CHAPTER 2

Java EE deployment

This chapter contains the following topics:

2.1 Introduction

This chapter details deployment specifications for TIBCO EBX® on a Java application server. For specific information regarding supported application servers and inherent limitations, see Supported environments. [p 6]

2.2 Software components

EBX® uses the following components:

- Library `ebx.jar`
- Embedded [p 14] and required [p 14] third-party Java libraries
- EBX® built-in web applications [p 17] and optional custom web applications [p 17]
- EBX® main configuration file [p 43]
- *EBX® repository*
- *Default user and roles directory*, integrated within the EBX® repository, or a third-party system (LDAP, RDBMS) for the user authentication

See also *Supported environments [p 6]*

2.3 Embedded third-party libraries

To increase EBX® independence and interoperability, it embeds its own third-party libraries. Even if some of them have been modified, preventing conflicts, others must remain unchanged since they are official Java APIs.

The ones that can produce conflicts are:

- Apache Geronimo JSON
- Javax Activation
- Javax Annotations
- Javax JSON Bind
- Javax SAAJ API
- Javax WS RS
- Javax XML Bind

For more information regarding the versions or the details of the Third-Party Library, please refer to the: `TIB_ebx_6.0.0_license.pdf`.

Since those libraries are already integrated, custom web applications should not include them anew, otherwise linkage errors can occur. Furthermore, they should not be deployed aside from the `ebx.jar` library for the same reasons.

2.4 Required third-party libraries

EBX® requires several third-party Java libraries. These libraries must be deployed and be accessible from the class-loader of `ebx.jar`. Depending on the application server and the Java runtime environment being used, these libraries may already be present or may need to be added manually.

Data compression library

The library named `ebx-lz4.jar` must be deployed separately from `ebx.jar`. It contains several compression implementations: JNI dedicated architecture libraries and Java fallbacks. It is possible to ensure optimal compression and decompression performance for EBX® repository by following prerequisites. If prerequisites can not be validated, EBX® will function in Java fallbacks safe or unsafe, but its performance will be degraded. The default location for `ebx-lz4.jar` library is beside `ebx.jar`.

To verify the compression implementation actually used by the EBX® repository, please check the value of 'Compression' in 'Administration > System Information', section 'Repository information'. It should be '`JNI - validated`' for optimal performance. Otherwise, it will be '`Java[Safe|Unsafe] - validated`' for Java fallbacks.

Performance prerequisites

The JNI access is allowed to the following operating system architectures: `i386`, `x86`, `amd64`, `x86_64`, `aarch64` or `ppc64le`. To verify this value, please check the value of 'Operating system architecture' in 'Administration > System Information', section 'System information'.

To enable JNI access for `ebx-lz4.jar`, the library should be loaded by the **system class loader** (also known as the application class loader). The deployment may be done by following the specific instructions for your application server [p 21].

Database drivers

The EBX® repository requires a database. Generally, the required driver is configured along with a data source, if one is used. Depending on the database defined in the main configuration file, one of the following drivers is required. Keep in mind that, whichever database you use, the version of the JDBC client driver must be equal to or higher than the version of the database server.

H2	Version 1.3.170 validated. Any 1.3.X version should be suitable. Note that H2 is not supported in production environments. http://www.h2database.com/
Oracle JDBC	Oracle database 12cR2 is validated on their latest patch set update. Determine the driver that should be used according to the database server version and the Java runtime environment version. Download the `ojdbc8.jar` certified library with JDK 8. Oracle database JDBC drivers download.
SQL Server JDBC	SQL Server 2012 SP4 and greater, with all corrective and maintenance patches applied, are validated. Remember to use an up-to-date JDBC driver, as some difficulties have been encountered with older versions. Include the `mssql-jdbc-8.4.1.jre8.jar` or `mssql-jdbc-8.4.1.jre11.jar` library, depending on the Java runtime environment version you use. Download Microsoft JDBC Driver 8.4.1 for SQL Server (zip).
PostgreSQL	PostgreSQL 9.6 and above validated Include the latest JDBC driver version 4.2 released for your database server and Java runtime environment. PostgreSQL JDBC drivers download.

See also

Data source of the EBX® repository [p 19]

Configuring the EBX® repository [p 45]

SMTP and emails

According to the web application server being used, the library `JavaMail API` for email management may already be provided, or must be added manually.

EBX® requires a library that is compatible with version 1.2 of this API. See Activating and configuring SMTP and emails [p 50] for more information on the configuration.

To facilitate manual installation, the `javax.mail-1.5.6.jar` has been provided and placed under the `ebx.software/lib/lib-mail` directory.

See also *JavaMail*

Secure Socket Layer (SSL)

These libraries are required if your web applications use SSL features.

- `jsse.jar`: https://www.oracle.com/java/technologies/jsse-v103-for-cdc-v102.html
- `ibmjsse.jar`: https://www.ibm.com/developerworks/java/jdk/security/

See also *TIBCO EBX® main configuration file [p 43]*

Java Message Service (JMS)

When using JMS, version 1.1 or higher is required.

Depending on whether a Java EE application server or a Servlet/Java Server Pages (JSP) implementation is being used, the library required is as follows:

- For an application server based on Java EE (Java Platform Enterprise Edition), the required JMS provider library is available by default. See https://www.oracle.com/java/technologies/java-ee-glance.html for more information.
- For a Servlet/Java Server Pages (JSP) implementation using Java SE (Java Platform Standard Edition), for example Apache Tomcat, a JMS provider library such as Apache ActiveMQ may need to be added. See https://www.oracle.com/java/technologies/java-se-glance.html for more information.

> **Note**
>
> In EBX®, the supported JMS model is exclusively Point-to-Point (PTP). PTP systems allow working with queues of messages.

See also *TIBCO EBX® main configuration file [p 43]*

XML Catalog API

A library holding the XML Catalog API, introduces by the JAVA SE 9, is required if your web applications are running over a Java Runtime Environment 8 or below, except when a WebLogic 14c application server is used. To ease the installation steps, the following library has been bundled aside from `ebx.jar`, in the *EBX® CD*.

- `xml-apis-1.4.01.jar`, version 1.4.01, from August 20, 2011

See Installation notes [p 21] for more information.

2.5 Web applications

EBX® provides pre-packaged EARs that can be deployed directly if your company has no custom EBX® module web applications to add. If deploying custom web applications as EBX® modules, it is recommended to rebuild an EAR containing the custom modules packaged at the same level as the built-in web applications.

For more information, see the note on repackaging the EBX® EAR [p 22] at the end of this chapter.

EBX® built-in web applications

EBX® includes the following built-in web applications.

Web application name	Description	Required
`ebx`	EBX® entry point, which handles the initialization on start up. See Deployment details [p 18] for more information.	Yes
`ebx-root-1.0`	EBX® root web application. Any application that uses EBX® requires the root web application to be deployed.	Yes
`ebx-ui`	EBX® user interface web application.	Yes
`ebx-manager`	EBX® user interface web application.	Yes
`ebx-dma`	EBX® data model assistant, which helps with the creation of data models through the user interface. **Note:** The data model assistant requires the `ebx-manager` user interface web application to be deployed.	Yes
`ebx-dataservices`	EBX® data services web application. Data services allow external interactions with the EBX® repository using the *SOAP operations* and Web Services Description Language *WSDL generation* standards or using the *Built-in RESTful services*. **Note:** The EBX® web service generator requires the deployment of the `ebx-manager` user interface web application.	Yes

Custom web applications

It is possible to extend and customize the behavior of EBX® by deploying custom web applications which conform to the EBX® module requirements.

See also

Packaging TIBCO EBX® modules

Declaring modules as undeployed [p 57]

2.6 Deployment details

Introduction

This section describes the various options available to deploy the 'ebx' web application. These options are available in its deployment descriptor (`WEB-INF/web.xml`) and are complemented by the properties defined in the main configuration file.

Attention

For JBoss application servers, any unused resources must be removed from the `WEB-INF/web.xml` deployment descriptor.

See also

TIBCO EBX® main configuration file [p 43]

Supported application servers [p 8]

User interface and web access

The web application 'ebx' (packaged as `ebx.war`) contains the servlet `FrontServlet`, which handles the initialization and serves as the sole user interface entry point for the EBX® web tools.

Configuring the deployment descriptor for 'FrontServlet'

In the file `WEB-INF/web.xml` of the web application 'ebx', the following elements must be configured for `FrontServlet`:

`/web-app/servlet/load-on-startup`	To ensure that `FrontServlet` initializes upon EBX® start up, the `web.xml` deployment descriptor must specify the element `<load-on-startup>1</load-on-startup>`.
`/web-app/servlet-mapping/url-pattern`	`FrontServlet` must be mapped to the path '`/`'.

Configuring the application server for 'FrontServlet'

- `FrontServlet` must be authorized to access other contexts, such as `ServletContext`.

  For example, on Tomcat, this configuration is performed using the attribute `crossContext` in the configuration file `server.xml`, as follows:

  ```
  <Context path="/ebx" docBase="(...)" crossContext="true"/>
  ```

- When several EBX® Web Components are to be displayed on the same HTML page, for instance using iFrames, it may be required to disable the management of cookies due to limitations present in some Internet browsers.

  For example, on Tomcat, this configuration is provided by the attribute `cookies` in the configuration file `server.xml`, as follows:

  ```
  <Context path="/ebx" docBase="(...)" cookies="false"/>
  ```

Data source of the EBX® repository

> **Note**
>
> If the EBX® main configuration specifies the property `ebx.persistence.url`, then the environment entry below will be ignored by EBX® runtime. This option is only provided for convenience; it is always recommended to use a fully-configurable datasource. In particular, the size of the connection pool must be set according to the number of concurrent users. See Configuring the EBX® repository [p 45] for more information on this property.

The JDBC datasource for EBX® is specified in the deployment descriptor `WEB-INF/web.xml` of the '`ebx`' web application as follows:

Reserved resource name	Default JNDI name	Description
`jdbc/EBX_REPOSITORY`	Weblogic: `EBX_REPOSITORY` JBoss: `java:/EBX_REPOSITORY`	JDBC data source for EBX® Repository. Java type: `javax.sql.DataSource`

See also

Configuring the EBX® repository [p 45]

Rules for the database access and user privileges

Mail sessions

> **Note**
>
> If the EBX® main configuration does not set `ebx.mail.activate` to 'true', or if it specifies the property `ebx.mail.smtp.host`, then the environment entry below will be ignored by EBX® runtime. See SMTP [p 50] in the EBX® main configuration properties for more information on these properties.

SMTP and email is declared in the deployment descriptor `WEB-INF/web.xml` of the '`ebx`' web application as follows:

Reserved resource name	Default JNDI name	Description
`mail/EBX_MAIL_SESSION`	Weblogic: `EBX_MAIL_SESSION` JBoss: `java:/EBX_MAIL_SESSION`	Java Mail session used to send emails from EBX®. Java type: `javax.mail.Session`

JMS connection factory

> **Note**
>
> If the EBX® main configuration does not activate JMS through the property `ebx.jms.activate`, the environment entry below will be ignored by the EBX® runtime. See JMS [p 51] in the EBX® main configuration properties for more information on this property.

The JMS connection factory is declared in the deployment descriptor `WEB-INF/web.xml` of the '`ebx`' web application as follows:

Reserved resource name	Default JNDI name	Description	Required
`jms/EBX_JMSConnectionFactory`	Weblogic: `EBX_JMSConnectionFactory` JBoss: `java:/EBX_JMSConnectionFactory`	JMS connection factory used by EBX® to create connections with the JMS provider configured in the operational environment of the application server. Java type: `javax.jms.ConnectionFactory`	Yes

> **Note**
>
> For deployment on WildFly, JBoss and WebLogic application servers with JNDI capabilities, you must update `EBX.ear` or `EBXForWebLogic.ear` for additional mappings of all required resource names to JNDI names.

JMS for data services

To configure data services to use JMS instead of the default HTTP, you must configure the JMS connection factory [p 20] and the following queues, declared in the `WEB-INF/web.xml` deployment descriptor of the '`ebx`' web application. This is the only method for configuring JMS for data services.

When a SOAP request is received, the SOAP response is optionally returned if the header field `JMSReplyTo` is defined. If so, the fields `JMSCorrelationID` and `JMSType` are retained.

See JMS [p 51] for more information on the associated EBX® main configuration properties.

> **Note**
>
> If the EBX® main configuration does not activate JMS through the property `ebx.jms.activate`, then the environment entries below will be ignored by EBX® runtime. See JMS [p 51] in the EBX® main configuration properties for more information on this property.

Reserved resource name	Default JNDI name	Description	Required
`jms/EBX_QueueIn`	Weblogic: `EBX_QueueIn` JBoss: `java:/jms/EBX_QueueIn`	JMS queue for incoming SOAP requests sent to EBX® by other applications. Java type: `javax.jms.Queue`	No
`jms/EBX_QueueFailure`	Weblogic: `EBX_QueueFailure` JBoss: `java:/jms/EBX_QueueFailure`	JMS queue for failures. It contains incoming SOAP requests for which an error has occurred. This allows replaying these messages if necessary. Java type: `javax.jms.Queue` **Note:** For this property to be read, the main configuration must also activate the queue for failures through the property `ebx.jms.activate.queueFailure`. See JMS [p 51] in the EBX® main configuration properties for more information on these properties.	No

JAR files scanner

To speed up the web applications server startup, the JAR files scanner configuration should be modified to exclude, at least, the `ebx.jar` and `ebx-addons.jar` libraries.

For example, on Tomcat, this should be performed in the `tomcat.util.scan.DefaultJarScanner.jarsToSkip` property from the `catalina.properties` file.

2.7 Installation notes

EBX® can be deployed on any Java EE application server that supports Servlet 3.0 up to 5.0 except. The following documentation on Java EE deployment and installation notes are available:

- Installation note for JBoss EAP 7.1.x [p 23]
- Installation note for Tomcat 9.x [p 29]
- Installation note for WebSphere AS 9 [p 33]

- Installation note for WebLogic 14c [p 39]

Attention

- The EBX® installation notes on Java EE application servers do not replace the native documentation for each application server.
- These are *not* general installation recommendations, as the installation process is determined by architectural decisions, such as the technical environment, application mutualization, delivery process, and organizational decisions.
- In these examples, no additional EBX® modules are deployed. To deploy additional modules, the best practice is to rebuild an EAR with the module as a web application at the same level as the other EBX® modules. The web application must declare its class path dependency as specified by the Java™ 2 Platform Enterprise Edition Specification, v1.4:

  J2EE.8.2 Optional Package Support

  (...)

  A JAR format file (such as a JAR file, WAR file, or RAR file) can reference a JAR file by naming the referenced JAR file in a Class-Path header in the Manifest file of the referencing JAR file. The referenced JAR file is named using a URL relative to the URL of the referencing JAR file. The Manifest file is named META-INF/MANIFEST.MF in the JAR file. The Class-Path entry in the Manifest file is of the form:

  `Class-Path: list-of-jar-files-separated-by-spaces`

  In an "industrialized" process, it is strongly recommended to develop a script that automatically builds the EAR, with the custom EBX® modules, the EBX® web applications, as well as all the required shared libraries.
- In order to avoid unpredictable behavior, the guideline to follow is to avoid any duplicates of `ebx.jar` or other libraries in the class-loading system.
- In case of deployment on Oracle WebLogic server, please refer to the *Module structure* section.

CHAPTER 3

Installation note for JBoss EAP 7.1.x

This chapter contains the following topics:

3.1 Overview

Attention

- This chapter describes a quick installation example of TIBCO EBX® on the JBoss Application Server.
- It does not replace the documentation of this application server.
- These are *not* general installation recommendations, as the installation process is determined by architectural decisions such as the technical environment, application mutualization, delivery process, and organizational decisions.
- The complete description of the components required by EBX® is given in the following chapter: Java EE deployment [p 13].
- In order to avoid unpredictable behavior, the guideline to follow is to avoid any duplicates of `ebx.jar` or other libraries in the class-loading system.

- JBoss Application Server installation
- `EBX_HOME` directory configuration
- Java Virtual Machine configuration
- JNDI entries configuration
- Data source and JDBC provider creation

- EBX.ear application update
- EBX.ear application deployment
- EBX® application start

3.2 Requirements

- JBoss Application Server EAP 7.1
- Database and JDBC driver
- EBX® CD
- No CDI features in EBX®'s additional modules (since CDI will be automatically disable)

See also *Supported environments* *[p 6]*

3.3 Installation

This quick installation example is performed for a Linux operating system.

1. To download JBoss EAP 7.1, please first download `Installer` jar version 7.1.x from: https://developers.redhat.com/products/eap/download/
2. Run the `Installer` using *java -jar* command line.
   For further installation details, please refer to the documentation .
3. Perform a standard installation:
   1. Select the language and click 'OK',
   2. Accept the License and click 'Next',
   3. Choose the installation path and click 'Next',
   4. Keep the 'Component Selection' as it is and click 'Next',
   5. Enter 'Admin username', 'Admin password' and click 'Next',
   6. On 'Installation Overview' click 'Next',
   7. On 'Component Installation' click 'Next',
   8. On 'Configure Runtime Environment' leave selection as it is and click 'Next',
   9. When 'Processing finished' appear, click 'Next',
   10. Uncheck 'Create shortcuts in the start menu' and click 'Next',
   11. Generate 'installation script and properties file' at JBoss EAP 7.1 installation root path,
   12. Click on 'done'.

3.4 Configuration for EBX®

EBX® home directory creation and configuration

1. Create the *EBX_HOME* directory, for example `/opt/ebx/home`.

2. Copy from the *EBX® CD*, the `ebx.software/files/ebx.properties` file to *EBX_HOME*. In our example, we will have the following text file:

   `/opt/ebx/home/ebx.properties`.

3. Edit the `ebx.properties` file to override the default database if needed. By default, the standalone H2 database is defined. The property key `ebx.persistence.factory` must be uncommented for other supported database and it is required to comment the `h2.standalone` one.

Java Virtual Machine configuration

1. Copy from the *EBX® CD*, the `ebx.software/lib/ebx-lz4.jar` [p 14] Data compression library to a dedicated directory (for example `<JBOSS_HOME>/compress`).
2. Open the `standalone.conf` configuration file, placed in `<JBOSS_HOME>/bin` (or `jboss-eap.conf` file placed in `<JBOSS_HOME>/bin/init.d` for running the server as a service).
3. Add 'ebx.properties' and 'ebx.home' properties to 'JAVA_OPTS' environment variable respectively set with `ebx.properties` file's path and *EBX_HOME* directory's path.
4. Set the 'JBOSS_MODULES_SYSTEM_PKGS' environment variable like the following:

```
JBOSS_MODULES_SYSTEM_PKGS="org.jboss.byteman,net.jpountz"
```

5. Open the `standalone.sh` script file, placed in `<JBOSS_HOME>/bin`.
6. Create a 'CLASSPATH' environment variable like the following:

```
CLASSPATH="<path_to_the_data_compression_library>:${JBOSS_HOME}/jboss-modules.jar:${CLASSPATH}"

# For our example
# CLASSPATH="${JBOSS_HOME}/compress/ebx-lz4.jar:${JBOSS_HOME}/jboss-modules.jar:${CLASSPATH}"
```

7. Replace the launch command options for foreground and background executions:

```
if [ "x$LAUNCH_JBOSS_IN_BACKGROUND" = "x" ]; then
   # Execute the JVM in the foreground
   eval \"$JAVA\" -D\"[Standalone]\" $JAVA_OPTS \
      -cp "$CLASSPATH" \
      \"-Dorg.jboss.boot.log.file="$JBOSS_LOG_DIR"/server.log\" \
      \"-Dlogging.configuration=file:"$JBOSS_CONFIG_DIR"/logging.properties\" \
      org.jboss.modules.Main \
      $MODULE_OPTS \
      -mp \""${JBOSS_MODULEPATH}"\" \
      org.jboss.as.standalone \
      -Djboss.home.dir=\""$JBOSS_HOME"\" \
      -Djboss.server.base.dir=\""$JBOSS_BASE_DIR"\" \
      "$SERVER_OPTS"
      JBOSS_STATUS=$?

else
   # Execute the JVM in the background
   eval \"$JAVA\" -D\"[Standalone]\" $JAVA_OPTS \
      -cp "$CLASSPATH" \
      \"-Dorg.jboss.boot.log.file="$JBOSS_LOG_DIR"/server.log\" \
      \"-Dlogging.configuration=file:"$JBOSS_CONFIG_DIR"/logging.properties\" \
      org.jboss.modules.Main \
      $MODULE_OPTS \
      -mp \""${JBOSS_MODULEPATH}"\" \
      org.jboss.as.standalone \
      -Djboss.home.dir=\""$JBOSS_HOME"\" \
      -Djboss.server.base.dir=\""$JBOSS_BASE_DIR"\" \
      "$SERVER_OPTS" "&"
...
fi
```

JNDI entries configuration

1. Open the `standalone-full.xml` file placed in `<JBOSS_HOME>/standalone/configuration`.

2. Add, at least, the following lines to the `server` tag in `messaging-activemq` subsystem:

```
<connection-factory
    name="jms/EBX_JMSConnectionFactory"
    entries="java:/EBX_JMSConnectionFactory"
    connectors="To Be Defined"/>
<jms-queue
    name="jms/EBX_D3ReplyQueue"
    entries="java:/jms/EBX_D3ReplyQueue"
    durable="true"/>
<jms-queue
    name="jms/EBX_QueueIn"
    entries="java:/jms/EBX_QueueIn"
    durable="true"/>
<jms-queue
    name="jms/EBX_QueueFailure"
    entries="java:/jms/EBX_QueueFailure"
    durable="true"/>
<jms-queue
    name="jms/EBX_D3MasterQueue"
    entries="java:/jms/EBX_D3MasterQueue"
    durable="true"/>
<jms-queue
    name="jms/EBX_D3ArchiveQueue"
    entries="java:/jms/EBX_D3ArchiveQueue"
    durable="true"/>
<jms-queue
    name="jms/EBX_D3CommunicationQueue"
    entries="java:/jms/EBX_D3CommunicationQueue"
    durable="true"/>
```

   Warning: the `connectors` attribute value, from the `connection-factory` element, has to be defined. Since the kind of connectors is strongly reliant on the environment infrastructure, a default configuration can not be provided.

   See configuring messaging for more information.

3. Add, at least, the following line to `mail` subsystem:

```
<mail-session name="mail" debug="false" jndi-name="java:/EBX_MAIL_SESSION"/>
```

Data source and JDBC provider creation

1. After the launch of the JBoss Server, run the management CLI without the use of '--connect' or '-c' argument.
2. Use the 'module add' management CLI command to add the new core module. Sample for PostgreSQL configuration:

```
module add \
  --name=org.postgresql \
  --resources=<PATH_TO_JDBC_JAR> \
  --dependencies=javax.api,javax.transaction.api
```

3. Use the 'connect' management CLI command to connect to the running instance.
4. Register the JDBC driver. When running in a managed domain, be sure to precede the command with '/profile=<PROFILE_NAME>'. Sample for PostgreSQL configuration:

```
/subsystem=\
  datasources/jdbc-driver=\
    postgresql:add(\
      driver-name=postgresql,\
      driver-module-name=org.postgresql,\
      driver-xa-datasource-class-name=org.postgresql.ds.PGConnectionPoolDataSource,\
      driver-class-name=org.postgresql.Driver\
    )
```

5. Define the datasource using the 'xa-data-source add' command, specifying the appropriate argument values. Sample for PostgreSQL configuration:

```
xa-data-source add \
  --name=jdbc/EBX_REPOSITORY \
  --jndi-name=java:/EBX_REPOSITORY \
```

```
--driver-name=postgresql \
--xa-datasource-class=org.postgresql.ds.PGConnectionPoolDataSource \
--xa-datasource-properties={"URL"=>"<CONNECTION_URL>"}
```

3.5 Updating EBX® Enterprise Application aRchive

1. Copy from the *EBX® CD*, the `ebx.software/webapps/ear-packaging/EBX.ear` file to your working directory.
2. Uncompress the ear archive to add the application's specific required third-party libraries.

   Mail: see SMTP and emails [p 16] for more information.

   SSL: see Secure Socket Layer (SSL) [p 16] for more information.

   JMS: see Java Message Service (JMS) [p 16] for more information.

   XML Catalog API: see XML Catalog API [p 16] for more information.
3. Compress anew the ear archive.

3.6 Deploying EBX®

1. Copy EBX.ear into *`JBOSS_HOME`*`/standalone/deployments` folder.

3.7 Start EBX®

1. After the launch of the JBoss Server, with the `<JBOSS_HOME>/bin/standalone.sh -c standalone-full.xml` command line or through the service command, run the EBX® web application: http://localhost:8080/ebx/.
2. At first launch, EBX® Wizard [p 61] helps you to configure the default properties of your initial repository.

CHAPTER 4

Installation note for Tomcat 9.x

This chapter contains the following topics:

4.1 Overview

Attention

- This chapter describes a *quick installation example* of TIBCO EBX® on Tomcat Application Server.
- It does not replace the documentation of this application server.
- They are *not* general installation recommendations, as the installation process is determined by architectural decisions, such as the technical environment, application mutualization, delivery process, and organizational decisions.
- Tomcat 10.x is not supported.
- The complete description of the components needed by EBX® is given in chapter Java EE deployment [p 13].
- In order to avoid unpredictable behavior, the guideline to follow is to avoid any duplicates of `ebx.jar`, `ebx-lz4.jar` or other libraries in the class-loading system.
- The description below uses the variable name `$CATALINA_HOME` to refer to the directory into which you have installed Tomcat, and from which most relative paths are resolved. However, if you have configured Tomcat for multiple instances by setting a `$CATALINA_BASE` directory, you should use `$CATALINA_BASE` instead of `$CATALINA_HOME` for each of these references.

- Create `EBX_HOME` directory: copy `ebx.properties`

- Configure JVM arguments (Java system properties): create the `JAVA_OPTS` environment variables and configure the `CLASSPATH`
- Copy EBX® and third-party libraries: add libraries (*jar files*) to Tomcat appropriate directories
- Deploy EBX® application: copy all war files to Tomcat `webapps` directory

4.2 Requirements

- Java SE 8 or 11
- Apache Tomcat 9.x
- Database and JDBC driver
- EBX® CD

See also *Supported environments [p 6]*

4.3 Installation

1. To download Tomcat 9.x, choose a core binary distributions from https://tomcat.apache.org/download-90.cgi
2. Run the installer or extract the archive and perform standard installation with default options

4.4 Configuration for EBX®

1. Create *EBX_HOME* directory, for example `C:\EBX\home`, or `/home/ebx`
2. Copy from *EBX® CD* the `ebx.software\files\ebx.properties` file to *EBX_HOME*. In our example, we will then have the following file:

   `C:\EBX\home\ebx.properties`, or `/home/ebx/ebx.properties`
3. Edit the `ebx.properties` file to override the default database if needed, by default the standalone H2 database is defined. The property key `ebx.persistence.factory` must be uncommented for other supported databases and it is required to comment the `h2.standalone` one.
4. Copy third-party library files from the *EBX® CD* to `$CATALINA_HOME\lib\` (or `$CATALINA_BASE\lib\`) directory. In our example, we will have:

   `$CATALINA_HOME\lib\javax.mail-1.5.6.jar` coming from `ebx.software\lib\lib-mail` directory.

   `$CATALINA_HOME\lib\h2-1.4.196.jar` (default persistence factory) coming from `ebx.software\lib\lib-h2` directory.

   `$CATALINA_HOME\lib\xml-apis-1.4.01.jar` coming from `ebx.software\lib\lib-xml-apis` directory.

   `$CATALINA_HOME\compress\ebx-lz4.jar` (compression library) coming from `ebx.software\lib` directory.

   The exact description of these components is given in chapter Software components [p 13]. Obviously, if those components are already deployed on the class-loading system, they do not have to be duplicated.

5. Modify `$CATALINA_HOME\conf\server.xml` (or `$CATALINA_BASE\conf\server.xml`) file. Add the following line to the <Host> element.

   ```
   <Context path="/ebx" crossContext="true" docBase="ebx.war"/>
   ```

   After this modification, we will have:

   ```
   <Host name=...>
   ... ...
   <Context path="/ebx" crossContext="true" docBase="ebx.war"/>
   ... ...
   </Host>
   ```

6. Modify the `$CATALINA_HOME\conf\catalina.properties` (or `$CATALINA_BASE\conf\catalina.properties`) file. Add the following lines to the `tomcat.util.scan.DefaultJarScanner.jarsToSkip` property:

   ```
   ebx.jar,\
   ebx-addons.jar,\
   ebx-lz4.jar,\
   ```

7. Configure the launch properties

   - For Windows' Command Prompt launch

     Set the environment variables by creating a `setenv.bat` file either into `$CATALINA_HOME\bin` or `$CATALINA_BASE\bin`. This file will hold, at least, the following lines:

     ```
     set EBX_HOME="<path_to_the_directory_ebx_home>"
     set EBX_OPTS="-Debx.home=%EBX_HOME% -Debx.properties=%EBX_HOME%\ebx.properties"
     set JAVA_OPTS="%EBX_OPTS% %JAVA_OPTS%"
     set CLASSPATH="<$CATALINA_HOME_or_$CATALINA_BASE>\compress\ebx-lz4.jar;%CLASSPATH%"
     ```

     Where <$CATALINA_HOME_or_$CATALINA_BASE> must be replaced by `%CATALINA_HOME%` or `%CATALINA_BASE%` if you have configured them. Otherwise you must replace this piece of text with the directory into which you have installed Tomcat.

   - For Windows users that have installed Tomcat as a service

     Set Java options through the Tomcat service manager GUI (Java tab).

     Be sure to set options on separate lines in the *Java Options* field of the GUI:

     ```
     -Debx.home=<path_to_the_directory_ebx_home>
     -Debx.properties=<path_to_the_directory_ebx_home>\ebx.properties
     ```

     Update the service using the //US// parameter to set the proper classpath value.

     ```
     C:\> tomcat9 //US//Tomcat9 --Classpath=<$CATALINA_HOME_or_$CATALINA_BASE>\compress\ebx-lz4.jar;
     %CLASSPATH%
     ```

     Where <$CATALINA_HOME_or_$CATALINA_BASE> must be replaced by `%CATALINA_HOME%` or `%CATALINA_BASE%` if you have configured them. Otherwise you must replace this piece of text with the directory into which you have installed Tomcat.

   - For Unix shell launch

     Set the environment variables by creating a `setenv.sh` file either into `$CATALINA_HOME\bin` or `$CATALINA_BASE\bin`. This file will hold, at least, the following lines:

     ```
     EBX_HOME="<path_to_the_directory_ebx_home>"
     EBX_OPTS="-Debx.home=${EBX_HOME} -Debx.properties=${EBX_HOME}/ebx.properties"
     export JAVA_OPTS="${EBX_OPTS} ${JAVA_OPTS}"
     export CLASSPATH="<$CATALINA_HOME_or_$CATALINA_BASE>/compress/ebx-lz4.jar:${CLASSPATH}"
     ```

Where <$CATALINA_HOME_or_$CATALINA_BASE> must be replaced by `${CATALINA_HOME}` or `${CATALINA_BASE}` if you have configured them. Otherwise you must replace this piece of text with the directory into which you have installed Tomcat.

(!) Accounts used to launch EBX® must have create/update/delete rights on *EBX_HOME* directory.

> **Note**
>
> <path_to_the_directory_ebx_home> is the directory where we copied `ebx.properties`. In our example, it is `C:\EBX\home`, or `/home/ebx`.

4.5 Deploying EBX®

1. Copy from *EBX® CD* the `ebx.software\lib\ebx.jar` file to `$CATALINA_HOME\lib\` (or `$CATALINA_BASE\lib\`) directory. In our example, we will have:

   `$CATALINA_HOME\lib\ebx.jar`

2. Copy from the *EBX® CD* the war files in `ebx.software\webapps\wars-packaging` to the `$CATALINA_HOME\webapps\` (or `$CATALINA_BASE\webapps\`) directory. In our example, we will have:

   `$CATALINA_HOME\webapps\ebx.war`: Initialization servlet for EBX® applications

   `$CATALINA_HOME\webapps\ebx-root-1.0.war`: Provides a common default module for data models

   `$CATALINA_HOME\webapps\ebx-manager.war`: Master Data Management web application

   `$CATALINA_HOME\webapps\ebx-dataservices.war`: Data Services web application

   `$CATALINA_HOME\webapps\ebx-dma.war`: Data Model Assistant web application

   `$CATALINA_HOME\webapps\ebx-ui.war`: User Interface web application

4.6 Start EBX®

1. After Tomcat launch, run EBX® web application: http://localhost:8080/ebx/
2. At first launch, EBX® Wizard [p 61] helps you to configure the default properties of your initial repository.

CHAPTER 5

Installation note for WebSphere AS 9

This chapter contains the following topics:

5.1 Overview

Attention

- This chapter describes a quick installation example of TIBCO EBX® on the WebSphere Application Server.
- It does not replace the documentation of the application server.
- These are *not* general installation recommendations, as the installation process is determined by architectural decisions such as the technical environment, application mutualization, delivery process, and organizational decisions.
- The complete description of the components required by EBX® is given in the following chapter: Java EE deployment [p 13].
- In order to avoid unpredictable behavior, the guideline to follow is to avoid any duplicates of `ebx.jar` or other libraries in the class-loading system.

- Install the WebSphere Application Server
- Create the `<EBX_HOME>` and `<EBX_LIB>` directories
- Create a data source and JDBC provider
- Configure the Java Virtual Machine

- Install the EBX.ear application
- Start the EBX® application

5.2 Requirements

- WebSphere Application Server 9
- Database and JDBC driver
- EBX® CD
- No CDI features in EBX®'s additional modules (since CDI will be automatically disabled).

See also *Supported environments [p 6]*

5.3 Install the WebSphere Application Server

This quick installation example is performed for a Linux operating system.

1. To download WebSphere AS 9, please first download 'Installation Manager' latest version from: https://www.ibm.com/support/pages/node/609575
2. Run the 'Installation Manager' and add the following repositories:
   - WebSphere Application Server V9.0:

     `http://www.ibm.com/software/repositorymanager/V9WASBase`
   - WebSphere Application Server Network Deployment V9.0:

     `http://www.ibm.com/software/repositorymanager/V9WASND`
3. Install the 'WebSphere Application Server Network Deployment'

   For further installation details, please refer to the documentation.
4. Run the 'WebSphere Customization Toolbox' and perform a standard installation with default options:
   1. Create profile: click 'Create' then select 'Application Server', and click 'Next'
   2. Profile Creation Options: select 'Advanced profile creation' and click 'Next'
   3. Optional Application Deployment: select those options:
      - Deploy the 'Administrative Console'
      - Deploy the 'Installation Verification Tool' application

      Click 'Next'
   4. Profile Name and Location: enter a profile name (example: 'EbxAppSrvProfile') and a directory. In our example, we will get:

      `/opt/IBM/WebSphere/AppServer/profiles/EbxAppSrvProfile`

      further correspond to `<PROFILE_HOME>` and click 'Next'
   5. Node and Host Names: enter the node name (example: 'Node1'), the server name (example: 'EbxServer'), the host name (example: 'localhost'), and click 'Next'
   6. Administrative Security: check 'Enable administrative security' option, enter the user name, the password, and click 'Next'

7. Security Certificate (part 1): select 'Create a new default personal certificate' and 'Create a new root signing certificate', and click 'Next'
8. Security Certificate (part 2): keep as default and click 'Next'
9. Port Value Assignment: keep as default and click 'Next'
10. Linux Service Definition: check 'Run the application server process as a Linux service' option, enter the user name (example: 'ebx'), and click 'Next'
11. Web Server Definition: keep as default and click 'Next'
12. Profile Creation Summary: keep as default and click 'Create'
13. Profile Creation Complete: uncheck 'Launch the First steps console' option, and click 'Finish'

5.4 Create the <EBX_HOME> and <EBX_LIB> directories

1. Create the `<EBX_HOME>` directory, for example `/opt/ebx/home`
2. Copy from the *EBX® CD*, the `ebx.software\files\ebx.properties` file to `<EBX_HOME>`. In our example, we will get the following text file:

   `/opt/ebx/home/ebx.properties`
3. Edit the `ebx.properties` file to override the default database if needed. By default, the standalone H2 database is defined. The property key `ebx.persistence.factory` must be uncommented for other supported database and it is required to comment the `h2.standalone` one.
4. Create the `<EBX_LIB>` directory, for example `/opt/ebx/lib`
5. Copy third-party library files, from the *EBX® CD* or from other sources, to the `<EBX_LIB>` directory. In our example, for a PostgreSQL database, we will get:

   `postgresql-X.X.X-driver.jar` (coming from another source than the *EBX® CD*).

   `xml-apis-1.4.01.jar` (coming from the `ebx.software/lib/lib-xml-apis/` directory of the *EBX® CD*).

   `ebx-lz4.jar` (coming from the `ebx.software/lib/` directory of the *EBX® CD*).

   The exact description of these components is given in the chapter Components [p 13]. If those components are already deployed on the class-loading system, they do not have to be duplicated (ex: javax.mail-1.5.6.jar is already present on the WebSphere Application Server, and the database driver is configured at the data source level).

5.5 Create a data source and JDBC provider

1. Start the server with the following command line:

   sudo <PROFILE_HOME>/bin/startServer.sh <serverName>

   where:

   <PROFILE_HOME> corresponds to the previously created profile home directory. In our example, we will get: `/opt/IBM/WebSphere/AppServer/profiles/EbxAppSrvProfile`.

   <serverName> corresponds to the server to start. In our example, we will get: `EbxServer`.
2. Connect into the 'WebSphere Integrated Solutions Console', using the user name and password typed during the profile creation (Administrative Security step), by entering the following url in the browser:

https://localhost:9043/ibm/console

3. On the left menu, go to 'Resources > JDBC > Data Sources', to configure your database access. Choose the jdbc 'Scope' (for example use 'Cell'), and click 'New'
4. Enter basic data source information:
   - Data source name: `EBX_REPOSITORY`
   - JNDI name: `jdbc/EBX_REPOSITORY`

   Click 'Next'
5. Select 'Create new JDBC provider', and click 'Next'
6. Create a new JDBC provider: (example with a PostgreSQL database)
   - Database type: `User-defined`
   - Implementation class name: `org.postgresql.ds.PGConnectionPoolDataSource`
   - Name: `PostgreSQL`

   Click 'Next'
7. Enter database class path information: (example with a PostgreSQL database)
   - Class path: `<EBX_LIB>/postgresql-X.X.X-driver.jar`

     In our example, <EBX_LIB> corresponds to `/opt/ebx/lib`.

   Click 'Next'
8. Enter database specific properties for the data source: keep as default and click 'Next'
9. Setup security aliases: keep as default and click 'Next'
10. Summary: click 'Finish'
11. Save the master configuration
12. Configure the data source: `jdbc/EBX_REPOSITORY`
    1. Click on 'Data Sources > `EBX_REPOSITORY`'
    2. On the right in the 'Configure additional properties' section, click on 'Additional Properties' and define the database account access:
       - Define `user` value to the according user
       - Define `password` value to the according password
    3. Save the master configuration
    4. Test the connection

5.6 Configure the Java Virtual Machine

1. Click on 'Application Servers'
2. Click on the server name (for example: 'EbxServer')
3. Click on 'Process definition' under 'Server infrastructure > Java Process Management'
4. Click on 'Java Virtual Machine' under 'Additional Properties'
5. Add 'ebx.properties' and 'ebx.home' properties, in the 'Generic JVM arguments' section, respectively set to `ebx.properties` file's path and `<EBX_HOME>` directory's path.

6. Add, in the 'Classpath' section, the paths to the third-party library files placed in the `<EBX_LIB>` directory. In our example, we will get:

```
/opt/ebx/lib/xml-apis-1.4.01.jar
/opt/ebx/lib/ebx-lz4.jar
```

> **Note**
>
> Every library's path declaration must be on a separate line.

7. Click 'Ok'
8. Save the master configuration

5.7 Install the EBX application

1. Copy from the *EBX® CD*, the `ebx.software/webapps/ear-packaging/EBX.ear` to the `<EBX_HOME>/ear` directory. In our example, we will get:

   `/opt/ebx/home/ear/EBX.ear`

2. Connect into the 'WebSphere Integrated Solutions Console', using the user name and password typed during the profile creation (Administrative Security step), by entering the following url in the browser:

   https://localhost:9043/ibm/console

3. Click on 'WebSphere enterprise applications' under 'Applications > Application Types'
4. Install the EBX®.ear
   1. Enterprise Applications: click on 'Install'
   2. Preparing for the application installation: Browse to your EBX.ear file. In our example, it is located under the `/opt/ebx/home/ear` directory.

      Click 'Next'
   3. How do you want to install the application?: Select 'Fast Path...', then click 'Next'
   4. Select installation options: keep as default, then click 'Next'
   5. Map modules to servers: select all modules, then click 'Next'
   6. Map resource references to resources: copy the 'Resource Reference' value and paste it in the 'Target Resource JNDI Name' field, for every modules, then click 'Next'
   7. Warnings will appear related to `JNDI:mail/EBX_MAIL_SESSION` and `JNDI:jms/EBX_JMSConnectorFactory`. This behavior is normal since we had not configured these resources.

      Click 'Continue'
   8. Map resource environment references to resources: Copy the 'Resource Reference' value and paste it to the 'Target Resource JNDI Name' value, for every modules, then click 'Next'
   9. Warnings will appear related to unavailable resources. This behavior is normal since we had not configured these resources.

      Click 'Continue'
   10. Map virtual hosts for Web modules: select all modules and click 'Next'

11. Summary: keep as default, click 'Finish'
12. If installation succeeds, it logs 'Application EBX® installed successfully'.

    Click 'Save'

5. On the left menu, go to 'Applications > Enterprise Applications'
6. Change EBX® application's class loader policy
   1. Click on EBX® resource’s name
   2. On the 'configuration' pane, under 'Detail Properties', click on 'Class loading and update detection'
   3. Under 'General Properties', change 'Class loader order' to 'Classes loaded with local class loader first (parent last)' and click 'OK'
   4. Save the master configuration
7. On the left menu, go to 'Applications > Enterprise Applications', select EBX®, then click 'Start'

   The EBX® 'Application status' will changed to a green arrow.

5.8 **Start EBX®**

1. After the launch of the WebSphere application Server, run the EBX® web application by entering the following url in the browser:

   http://localhost:9080/ebx/
2. At first launch, EBX® Wizard [p 61] helps you to configure the default properties of your initial repository.

CHAPTER 6

Installation note for WebLogic 14c

This chapter contains the following topics:

6.1 Overview

Attention

- This chapter describes a quick installation example of TIBCO EBX® on the WebLogic Server.
- It does not replace the documentation of this application server.
- They are *not* general installation recommendations, as the installation process is determined by architectural decisions, such as the technical environment, application mutualization, delivery process, and organizational decisions.
- The complete description of the components needed by EBX® is given in the following chapter: Java EE deployment [p 13].
- In order to avoid unpredictable behavior, the guideline to follow is to avoid any duplicates of `ebx.jar` or other libraries in the class-loading system.

- Install the Java Virtual Machine: Create the JAVA_HOME environment variable
- Install the WebLogic Server
- Create the `EBX_HOME` directory: copy `ebx.properties`
- Create a new domain by using the 'Configuration Wizard'
- Configure the domain: Declare EBX® `JAVA_OPTIONS` and copy the database JDBC driver
- Install and configure the JDBC driver
- Deploy the EBX® application: install dedicated ear file

6.2 Requirements

- Certified Oracle Java SE 8 or 11
- WebLogic Server 14c
- Database and JDBC driver
- EBX® CD

See also *Supported environments* *[p 6]*

6.3 Installation

1. To download WebLogic 14c, please refer to https://www.oracle.com/middleware/technologies/fusionmiddleware-downloads.html
2. Run the 'Oracle Fusion Middleware Weblogic installation' wizard using a certified Oracle JDK and the *java -jar* command line
3. Perform a standard installation with default options and choose the appropriate installation directory
4. Leave the 'Automatically launch the Configuration Wizard' option activated to perform the next steps:
    1. Create Domain: choose 'Create a new domain' and specify the domain home folder, then click 'Next'
    2. Templates: keep as default and click 'Next'
    3. Administrator Account: enter a domain administrator username and password and click 'Next'
    4. Domain Mode and JDK: choose the production mode and your jdk installation home and click 'Next'
    5. Advanced configuration: check 'Administration server' and 'Topology'. That way, we create two independent domain nodes: an administration one and an application one, and click 'Next'
    6. Administration Server: enter your administration node name (for example 'AdminServer') and listen port (by default `7001`), then click 'Next'
    7. Managed Servers: add the application node name (for example 'EbxServer') and listen port (for example `7003`), then click 'Next'
    8. Clusters: keep as default and click 'Next'
    9. Server Templates: keep as default and click 'Next'
    10. Machines: keep as default and click 'Next'
    11. Configuration Summary: click 'Create'
    12. Configuration Process: click 'Next'
    13. End Of Configuration: click 'Finish'

6.4 Configuration for EBX®

1. Create the *EBX_HOME* directory, for example `C:\EBX\home`, or `/home/ebx`

2. Copy from the *EBX® CD* the `ebx.software\files\ebx.properties` file to *EBX_HOME*. In our example, we will then have the following file:

   `C:\EBX\home\ebx.properties`, or `/home/ebx/ebx.properties`

3. Edit the `ebx.properties` file to override the default database if needed, by default the standalone H2 database is defined. The property key `ebx.persistence.factory` must be uncommented for other supported databases and it is required to comment the `h2.standalone` one.

4. Configure the launch properties for the *Managed Server* (for example 'EbxServer')

   Edit the `<DOMAIN_HOME>/bin/startManagedWebLogic.sh` script file by adding the following lines:

   ```
   EBX_HOME="<path_to_the_directory_ebx_home>"
   EBX_OPTIONS="-Debx.home=${EBX_HOME} -Debx.properties=${EBX_HOME}/ebx.properties"
   export JAVA_OPTIONS="${EBX_OPTIONS} ${JAVA_OPTIONS}"
   ```

5. 1. Copy third-party library files from the *EBX® CD* to the `<DOMAIN_HOME>/lib` directory except for the Data compression library [p 14]. In our example, for an H2 standalone data base, we will have:

      `h2-1.4.196.jar` (default persistence factory)

   2. Create a directory dedicated to the Data compression library [p 14] (for example `<DOMAIN_HOME>/compress`) and copy it there.

   3. Edit the `<DOMAIN_HOME>/bin/setDomainEnv.sh` script file by adding the following line:

      ```
      PRE_CLASSPATH="<path_to_the_data_compression_library>"

      # For our example
      # PRE_CLASSPATH="${DOMAIN_HOME}/compress/ebx-lz4.jar"
      ```

   The exact description of these components is given in chapter Components [p 13]. Obviously, if those components are already deployed on the class-loading system, they do not have to be duplicated (ex: javax.mail-1.5.6.jar and xml-apis-1.4.01.jar are already present in the WebLogic Server).

6. Start the 'Administration server' (for example 'AdminServer'), `<DOMAIN_HOME>/bin/startWebLogic.sh`

7. Launch the 'WebLogic Server Administration Console' by entering the following url in the browser:

   http://localhost:7001/console.

   Log in with the domain administrator username and password

8. Click on 'Services > Data sources' in the 'Domain Structure' panel, then click on the 'New > Generic Data Source' button:

   1. Type Name: `EBX_REPOSITORY`, JNDI Name: `EBX_REPOSITORY` Database Type: *Your database type* and click 'Next'
   2. Choose your database driver type, and click 'Next'
   3. Uncheck 'Supports Global Transactions', and click 'Next'
   4. Setup your database 'Connection Properties' and click 'Next'
   5. Click 'Test Configuration' and then 'Finish'
   6. Switch on the 'Targets' tab and select all Servers, then click 'Save'
   7. Restart the Administration server (for example 'AdminServer')

      `<DOMAIN_HOME>/bin/stopWebLogic.sh`

```
<DOMAIN_HOME>/bin/startWebLogic.sh
```

6.5 Deploying EBX®

1. Copy from the *EBX® CD* the `ebx.software/webapps/ear-packaging/EBXForWebLogic.ear` to the *EBX_HOME* directory. In our example, we will have:

   `C:\EBX\home\EBXForWebLogic.ear`, or `/home/ebx/EBXForWebLogic.ear`

2. Launch the 'WebLogic Server Administration Console' by entering the following url in the browser:

   http://localhost:7001/console

3. Click on 'Lock and Edit' in the 'Change Center' panel
4. Click on 'Deployments' in the 'Domain Structure' panel, and click 'Install':
   1. Install Application Assistant: Enter in 'Path' the application full path to EBXForWebLogic.ear file, located on `C:\EBX\home\`, or `/home/ebx/` folder and click 'Next'
   2. Choose the installation type and scope: Click on 'Install this deployment as an application' and 'Global' default scope and click 'Next'
   3. Select the deployment targets: Select a node name (for example 'EbxServer') from the 'Servers' list and click 'Next'
   4. Optional Settings: keep as default and click 'Finish'
5. Click on 'Activate Changes', on the top left corner. The deployment status will change to 'prepared'
6. Switch to 'Control' tab, select the 'EBXForWebLogic' enterprise application, then click on 'Start' > 'Servicing all requests'
7. Start the application node name (for example 'EbxServer'):

   ```
   <DOMAIN_HOME>/bin/startManagedWebLogic.sh EbxServer http://localhost:7001
   ```

6.6 Start EBX®

1. After WebLogic Server launch, run the EBX® web application:

   http://localhost:7003/ebx/

2. At first launch, EBX® Wizard [p 61] helps you to configure the default properties of your initial repository

CHAPTER 7

TIBCO EBX® main configuration file

This chapter contains the following topics:

7.1 Overview

The EBX® main configuration file, by default named `ebx.properties`, contains most of the basic parameters for running EBX®. It is a Java properties file that uses the standard simple line-oriented format.

The main configuration file complements the Java EE deployment descriptor [p 18]. Administrators can also perform further configuration through the user interface, which is then stored in the EBX® repository.

See also

Deployment details *[p 18]*

UI administration

Location of the file

The access path to the main configuration file can be specified in several ways. In order of descending priority:

1. By defining the Java system property 'ebx.properties'. For example, this property can be set by adding the option `-Debx.properties=<filePath>` to the `java` command-line command. See Java documentation.
2. By defining the servlet initialization parameter 'ebx.properties'.

   This standard Java EE setting must be specified in the `web.xml` file of the web application '`ebx`'. EBX® accesses this parameter by calling the method `ServletConfig.getInitParameter("ebx.properties")` in the servlet `FrontServlet`.

   See getInitParameter in the Oracle `ServletConfig` documentation.
3. By default, if nothing is specified, the main configuration file is located at `WEB-INF/ebx.properties` of the web application '`ebx`'.

> **Note**
>
> In addition to specifying properties in the main configuration file, it is also possible to set the values of properties directly in the system properties. For example, using the `-D` argument of the `java` command-line command.

Custom properties and variable substitution

The value of any property can include one or more variables that use the syntax `${`*`propertyKey`*`}`, where *`propertyKey`* is either a system property, or a property defined in the main configuration file.

For example, the default configuration file provided with EBX® uses the custom property `ebx.home` to set a default common directory, which is then included in other properties.

7.2 Setting automatic installation on first launch

Repository can be automatically installed on first startup.

```
######################################################################
## Installation on first launch.
## All values are ignored if the repository is  already installed.
######################################################################
## Enables repository installation on first startup (default is false).
ebx.install.enabled=true

## Following properties configure the repository. Values are optional and defaults are automatically generated.
ebx.install.repository.id=00275930BB88
ebx.install.repository.label=A Test

## Following properties specify the EBX administrator. These are ignored if a custom directory is defined.
ebx.install.admin.login=admin
ebx.install.admin.firstName=admin
ebx.install.admin.lastName=admin
ebx.install.admin.email=adamin@example.com

## Following property specifies the none encrypted password used for the EBX administrator.
## It is ignored if a custom directory is defined. It cannot be set if property
 ebx.install.admin.password.encrypted is set.
#ebx.install.admin.password=admin
```

```
## Following property specifies the encrypted password used for the EBX administrator.
## It is ignored if a custom directory is defined. It cannot be set if property ebx.install.admin.password is
 set.
## Password can be encrypted by using command:
##  java -cp ebx.jar com.orchestranetworks.service.directory.EncryptPassword password_to_encrypt
ebx.install.admin.password.encrypted=8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918
```

7.3 Setting the EBX® root directory

The EBX® root directory contains archives, the XML audit trail and, when the repository is persisted on H2 standalone mode, the H2 database files.

```
################################################
## Path for EBX® XML repository
################################################
ebx.repository.directory=${ebx.home}/ebxRepository
```

7.4 Configuring the EBX® repository

Before configuring the persistence properties of the EBX® repository, carefully read the section *Technical architecture* in the chapter 'Repository administration'.

The required library (driver) for each supported database is described in the chapter Database drivers [p 15].

See also

Repository administration

Rules for the database access and user privileges

Supported databases [p 10]

Data source of the EBX® repository [p 19]

Database drivers [p 15]

```
#####################################################################
## The maximum time to set up the database connection,
## in milliseconds.
#####################################################################
ebx.persistence.timeout=10000
#####################################################################
## The prefix to add to all table names of persistence system.
## This may be useful for supporting multiple repositories in the relational database.
## Default value is 'EBX_'.
#####################################################################
ebx.persistence.table.prefix=

#####################################################################
## Case EBX® persistence system is H2 'standalone'.
#####################################################################
ebx.persistence.factory=h2.standalone
ebx.persistence.user=sa
ebx.persistence.password=

#####################################################################
## Case EBX® persistence system is H2 'server mode',
#####################################################################
#ebx.persistence.factory=h2.server

## Specific properties to be set only only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url=jdbc:h2:tcp://127.0.0.1/ebxdb
#ebx.persistence.user=xxxxxxxxx
#ebx.persistence.password=yyyyyyyy


#####################################################################
## Case EBX® persistence system is Oracle database.
#####################################################################
```

```
#ebx.persistence.factory=oracle

## Specific properties to be set only only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url=jdbc:oracle:thin:@127.0.0.1:1521:ebxDatabase
#ebx.persistence.driver=oracle.jdbc.OracleDriver
#ebx.persistence.user=xxxxxxxxx
#ebx.persistence.password=yyyyyyyy


## Activate to use VARCHAR2 instead of NVARCHAR2 on Oracle; never modify on an existing repository.
#ebx.persistence.oracle.useVARCHAR2=false

#####################################################################
## Case EBX® persistence system is SAP Hana
#####################################################################
#ebx.persistence.factory=hana

## Specific properties to be set only only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url=jdbc:sap://127.0.0.1:39041
#ebx.persistence.driver=com.sap.db.jdbc.Driver
#ebx.persistence.user=xxxxxxxxx
#ebx.persistence.password=yyyyyyyy


#####################################################################
## Case EBX® persistence system is Microsoft SQL Server.
#####################################################################
#ebx.persistence.factory=sqlserver

## Specific properties to be set only only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url= \
#jdbc:sqlserver://127.0.0.1:1036;databasename=ebxDatabase
#ebx.persistence.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
#ebx.persistence.user=xxxxxxxxx
#ebx.persistence.password=yyyyyyyy


#####################################################################
## Case EBX® persistence system is Microsoft Azure SQL database.
#####################################################################
#ebx.persistence.factory=azure.sql

## Specific properties to be set only only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url= \
#jdbc:sqlserver://myhost.database.windows.net:1433;database=ebxDatabase;encrypt=true;\
#trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;
#ebx.persistence.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
#ebx.persistence.user=xxxxxxxxx
#ebx.persistence.password=yyyyyyyy


#####################################################################
## Case EBX® persistence system is PostgreSQL.
#####################################################################
#ebx.persistence.factory=postgresql

## Specific properties to be set only only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url=jdbc:postgresql://127.0.0.1:5432/ebxDatabase
#ebx.persistence.driver=org.postgresql.Driver
#ebx.persistence.user=xxxxxxxxx
#ebx.persistence.password=yyyyyyyy
```

7.5 **Configuring the user and roles directory**

This parameter specifies the Java directory factory class name. It must only be defined if not using the default EBX® directory.

See also

Users and roles directory

`DirectoryFactory`[API]

```
################################################
## Specifies the Java directory factory class name.
## Value must be the fully qualified name of the Java class.
## The class must extend com.orchestranetworks.service.directory.DirectoryFactory.
################################################
#ebx.directory.factory=xxx.yyy.DirectoryFactoryImpl
```

It is also possible to disable the built-in role "ADMINISTRATOR".

```
################################################
## Specifies whether the built-in role ADMINISTRATOR is disabled.
## Default value is false.
################################################
#ebx.directory.disableBuiltInAdministrator=true
```

7.6 Configuring EBX® localization

This parameter is used to configure the locales used at runtime. This list must contain all the locales that are exposed to the end-user. EBX® will not be able to display labels and messages in a language that is not declared in this list.

The default locale must be the first one in the list.

```
#####################################################################
## Available locales, separated by a comma.
## The first element in the list is considered as the default locale.
## If not set, available locales are 'en-US, fr-FR'.
##
#####################################################################
#ebx.locales.available=en-US, fr-FR
```

See also *Extending TIBCO EBX® internationalization*

7.7 Setting temporary files directories

Temporary files are stored as follows:

```
################################################
## Directories for temporary resources.
################################################
# The property ebx.temp.directory allows to specify a directory for temporary files.
# Default value is java.io.tmpdir
#
ebx.temp.directory = ${java.io.tmpdir}
#ebx.temp.directory = /tmp/java

# The property ebx.temp.cache.directory allows to specify the directory containing temporary files for cache.
# Default value is ${ebx.temp.directory}/ebx.platform.
#ebx.temp.cache.directory = ${ebx.temp.directory}/ebx.platform

# The property ebx.temp.import.directory allows to specify the directory containing temporary files for import.
# Default value is ${ebx.temp.directory}/ebx.platform.
#ebx.temp.import.directory = ${ebx.temp.directory}/ebx.platform
```

7.8 Activating the XML audit trail

By default, the XML audit trail is deactivated. It can be activated using the following variable:

```
#####################################################################
# The XML history has been replaced by an SQL history.
# This old XML history can be activated using the following variable.
# Default is false.
```

```
#####################################################################
ebx.history.xmlaudittrail.activated = false
```

See also *Audit trail*

7.9 Configuring the EBX® logs

The most important logging categories are:

ebx.log4j.category.log.kernel	Logs for EBX® main features, processes, exceptions and compilation results of modules and data models.
ebx.log4j.category.log.workflow	Logs for main features, warnings and exceptions about workflow.
ebx.log4j.category.log.persistence	Logs related to communication with the underlying database.
ebx.log4j.category.log.setup	Logs for the compilation results of all EBX® objects, except for modules and data models.
ebx.log4j.category.log.validation	Logs for datasets validation results.
ebx.log4j.category.log.mail	Logs for the activity related to the emails sent by the server (see Activating and configuring SMTP and emails [p 50]). **Note:** This category must not use the Custom SMTP appender [p 50] in order to prevent infinite loops.
ebx.log4j.category.log.d3	Logs for D3 events on EBX®.
ebx.log4j.category.log.dataservices	Logs for data service events in EBX®.
ebx.log4j.category.log.monitoring	Raw logs for *monitoring*.
ebx.log4j.category.log.request	Logs all `Request`[API] and `Query`[API] issued in the EBX® repository having a duration exceeding `ebx.logs.request.durationThreshold` milliseconds. All queries are logged regardless of their duration, if log level is set to `DEBUG`.
ebx.log4j.category.log.restServices	Logs for REST services events in EBX®, including those from the *REST Toolkit*.

Some of these categories can also be written to through custom code using the `LoggingCategory`[API] interface.

```
 ################################################
## Log4J properties:
##
## We have some specific syntax extensions:
## - Appender ebxFile:<aFileName>
## Defines a file appender with default settings (threshold=DEBUG)
##
## - property log.defaultConversionPattern is set by Java
################################################
#ebx.log4j.debug=true
#ebx.log4j.disableOverride=
#ebx.log4j.disable=
ebx.log4j.rootCategory= INFO
ebx.log4j.category.log.kernel= INFO, Console, ebxFile:kernel, kernelMail
ebx.log4j.category.log.workflow= INFO, ebxFile:workflow
ebx.log4j.category.log.persistence= INFO, ebxFile:persistence
ebx.log4j.category.log.setup= INFO, Console, ebxFile:kernel
ebx.log4j.category.log.mail= INFO, Console, ebxFile:mail
ebx.log4j.category.log.frontEnd= INFO, Console, ebxFile:kernel
ebx.log4j.category.log.frontEnd.incomingRequest= INFO
ebx.log4j.category.log.frontEnd.requestHistory= INFO
ebx.log4j.category.log.frontEnd.UIComponentInput= INFO
ebx.log4j.category.log.fsm= INFO, Console, ebxFile:fsm
ebx.log4j.category.log.fsm.dispatch= INFO
ebx.log4j.category.log.fsm.pageHistory= INFO
ebx.log4j.category.log.wbp= FATAL, Console
#-------------------------------------------------
ebx.log4j.appender.Console.Threshold = INFO
ebx.log4j.appender.Console=com.onwbp.org.apache.log4j.ConsoleAppender
ebx.log4j.appender.Console.layout=com.onwbp.org.apache.log4j.PatternLayout
ebx.log4j.appender.Console.layout.ConversionPattern=${log.defaultConversionPattern}
#-------------------------------------------------
ebx.log4j.appender.kernelMail.Threshold = ERROR
ebx.log4j.appender.kernelMail = com.onwbp.org.apache.log4j.net.SMTPAppender
ebx.log4j.appender.kernelMail.To = admin@domain.com
ebx.log4j.appender.kernelMail.From = admin${ebx.site.name}
ebx.log4j.appender.kernelMail.Subject = EBX® Error on Site ${ebx.site.name} (VM ${ebx.vm.id})
ebx.log4j.appender.kernelMail.layout.ConversionPattern=**Site ${ebx.site.name} (VM${ebx.vm.id})**%n
${log.defaultConversionPattern}
ebx.log4j.appender.kernelMail.layout = com.onwbp.org.apache.log4j.PatternLayout

#-------------------------------------------------
ebx.log4j.category.log.monitoring= INFO, ebxFile:monitoring
ebx.log4j.category.log.dataServices= INFO, ebxFile:dataServices
ebx.log4j.category.log.d3= INFO, ebxFile:d3
ebx.log4j.category.log.request= INFO, ebxFile:request
ebx.log4j.category.log.restServices= INFO, ebxFile:dataServices
```

Custom 'ebxFile' appender

The token `ebxFile:` can be used as a shortcut to define a daily rolling file appender with default settings. It must be followed by a file name. It then activates an appender that writes to a file located in the directory `ebx.logs.directory`, with a threshold set to `DEBUG`.

The property `ebx.log4j.appender.ebxFile.backup.Threshold` allows defining the maximum number of backup files for daily rollover.

```
################################################
## Directory of log files 'ebxFile:'
## This property is used by special appender prefixed
## by 'ebxFile:' (see log section below)
################################################
ebx.logs.directory=${ebx.home}/ebxLog

#####################################################################
# Daily rollover threshold of log files 'ebxFile:'
# Specifies the maximum number of backup files for daily rollover of 'ebxFile:' appenders.
# When set to a negative value, backup log files are never purged.
# Default value is -1.
#####################################################################
ebx.log4j.appender.ebxFile.backup.Threshold=-1
```

Custom SMTP appender

The appender `com.onwbp.org.apache.log4j.net.SMTPAppender` provides an asynchronous email sender.

See also *Activating and configuring SMTP and emails* *[p 50]*

Custom module log threshold

By default, the log level threshold of the logging category associated with a custom module is set to `INFO`.

This threshold can be customized by setting the property `ebx.log4j.category.log.wbp.xxxxxx` for the custom module `xxxxxx`.

Example: `ebx.log4j.category.log.wbp.mycompany-module=DEBUG`.

See also *`ModuleContextOnRepositoryStartup.getLoggingCategory`*[API]

Add-on module log threshold

By default, the log level threshold of any add-on module is set to `INFO`.

The log level threshold can be customized by setting the property `ebx.log4j.category.log.addon.xxxxxx` for the add-on module `ebx-addon-xxxxxx`.

Example: `ebx.log4j.category.log.addon.daqa=DEBUG`

7.10 Activating and configuring SMTP and emails

The internal mail manager sends emails asynchronously. It is used by the workflow engine and the custom SMTP appender `com.onwbp.org.apache.log4j.net.SMTPAppender`.

See also *Mail sessions* *[p 19]*

```
################################################
## SMTP and emails
################################################

## Activate emails (true or false, default is false).
## If activated, the deployer must ensure that the entry 'mail/EBX_MAIL_SESSION' is bound
## in the operational environment of the application server (except if a specific email
## configuration is used by setting the property ebx.mail.smtp.host below).
#ebx.mail.activate=false

## Polling interval is in seconds (default is 10).
#ebx.mail.polling.interval=10

## Specific properties to be set only only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.mail.smtp.host = smtp.domain.com
## SMTP port default is 25.
#ebx.mail.smtp.port= 25
#ebx.mail.smtp.login=
#ebx.mail.smtp.password=
## Activate SSL (true or false, default is false).
## If SSL is activated, a SSL factory and a SSL provider are required.
#ebx.mail.smtp.ssl.activate=true
#ebx.mail.smtp.ssl.provider=com.sun.net.ssl.internal.ssl.Provider
#ebx.mail.smtp.ssl.factory=javax.net.ssl.SSLSocketFactory
```

7.11 **Configuring data services**

```
######################################################################
## Data services
######################################################################

# Specifies the default value of the data services parameter
# 'disableRedirectionToLastBroadcast'.
# Default is false.
#ebx.dataservices.disableRedirectionToLastBroadcast.default=false

# Specifies the default value for deletion at the end of close and
# merge operations.
# If the parameter is set in the request operation, it overrides
# this default setting.
# If unspecified, default is false.
#ebx.dataservices.dataDeletionOnCloseOrMerge.default=false
#ebx.dataservices.historyDeletionOnCloseOrMerge.default=false

# Specifies the default maximum pagination size value for the select
# operations. This configuration is used by SOAP and REST connectors.
# Default value is 10000, maximum recommended value is 100000
#ebx.dataservices.pagination.maxSize.default= 10000

# Upon WSDL generation, specifies if the target namespace value
# corresponds to the content before 5.5.0 'ebx-services'
# or 'urn:ebx:ebx-services' in conformity with the URI syntax.
# If the parameter is set to true, there is no check of the target
# namespace as URI at the WSDL generation.
# If unspecified, default is false.
#ebx.dataservices.wsdlTargetNamespace.disabledCheck=false

######################################################################
## REST configuration
######################################################################

# If activated, the HTTP request header 'Accept' is used to specify
# the accepted content type. If none is supported, an error is
# returned to the client with the HTTP code 406 'Not acceptable'.
# If deactivated, the header is ignored therefore the best content
# type is used.
# Default is false.
#ebx.dataservices.rest.request.checkAccept=false

# If activated, when a REST data service authentication negotiate fails,
# EBX response includes fallback to 'Basic' authentication method by setting
# the HTTP header 'WWW-Authenticate' to 'Basic'.
# Note: This property only activate/deactivate
# the authentication fallback.
# Default is false.
#ebx.dataservices.rest.auth.tryBasicAuthentication=false

# Authorization token timeout is seconds.
# Default value is 1800 seconds (30 minutes)
# This value is ignored if 'Token Authentication Scheme' is not activated.
#ebx.dataservices.rest.auth.token.timeout=1800
```

7.12 **Activating and configuring JMS**

See also *JMS for data services* *[p 20]*

```
######################################################################
## JMS configuration for Data Services
######################################################################

## Activates JMS (true or false, default is false).
## If activated, the deployer must ensure that the entry 'jms/EBX_JMSConnectionFactory'
## are bound in the operational environment of the application server.
## The entry 'jms/EBX_QueueIn' should also be bound to enable handling Data Services
## request using JMS.
#ebx.jms.activate=false

## Activates JMS queue for failures (true or false, default is false).
## If activated, the deployer must ensure that the entry 'jms/EBX_QueueFailure' is bound
## in the operational environment of the application server.
#ebx.jms.activate.queueFailure=false
```

```
## Number of concurrent listener(s)
## Default is 3.
## Property is used if ebx.jms.activate is set to true.
#ebx.jms.listeners.count=3
```

7.13 Configuring distributed data delivery (D3)

See *Configuring D3 nodes* for the main configuration file properties pertaining to D3.

See also

JMS for distributed data delivery (D3)

Introduction to D3

7.14 Configuring REST toolkit services

```
################################################################
## REST configuration
################################################################

# Defines the maximum number of bytes that will be extracted
# from the REST request body to build some DEBUG log messages.
# Default value is 8192 bytes.
# This value is ignored if DEBUG level is not activated on the restServices logger.
#ebx.restservices.log.body.content.extract.size=8192
```

7.15 Configuring Web access from end-user browsers

HTTP Authorization header policy

EBX® natively offers three policies to send and receive credentials using HTTP headers:

`standard`	It corresponds to the authentication scheme, using the HTTP Authorization header, described in the RFC 2617.
`ebx`	To prevent HTTP Authorization header override issues, this policy acts the same as the `standard` but the credentials are stored in an EBX® specific HTTP header.
`both`	It is the combination of the two previously described policies.

```
################################################
## EBX® authorization header policy for HTTP requests
##
## Possible values are: standard, ebx, both.
##  standard:
##    the standard HTTP Authorization header holds the credentials
##  ebx:
##    an EBX® specific HTTP header holds the credentials
##  both:
##    both (standard and specific) HTTP headers hold the credentials
##
## Default value is: both.
################################################
#ebx.http.authorization.header.policy=both
```

URLs computing

By default, EBX® runs in "standalone" mode, where external resources (images, JavaScript, etc.) are provided by the application server.

Also by default, URL-related parameters in the main configuration file do not have to be set.

In this case, the server name and the port are obtained from the initial request sent to EBX®.

```
#####################################################################
## EBX® FrontServlet: default properties for computing servlet address
##
## {useLocalUrl}:
## If set to true, servlet address is a "local absolute" URL.
## (that is, a relative URL consisting of an absolute path: "/path")
## See RFC 2396, http://www.ietf.org/rfc/rfc2396.txt).
## This property is defined once for HTTP and HTTPS.
## Default value is false.
##
## {host}:
## If neither defined nor adapted, retrieves initial request host
## {port}:
## If neither defined nor adapted, retrieves initial request host
## {path}:
## Mandatory, may be empty
## {ui.path}:
## If not defined, defaults to ebx-ui/
##
## Resulting address will be:
## EBX®: protocol://{host}:{port}/{path}
## UI: protocol://{host}:{port}/{ui.path}
##
## Each property for HTTP (except {port}) may be inherited from HTTPS property,
## and reciprocally.
#####################################################################

#ebx.servlet.useLocalUrl=true
#ebx.servlet.http.host=
#ebx.servlet.http.port=
ebx.servlet.http.path=ebx/
#ebx.servlet.http.ui.path=ebx-ui/
#ebx.servlet.https.host=
#ebx.servlet.https.port=
#ebx.servlet.https.path=
#ebx.servlet.https.ui.path=ebx-ui/

#####################################################################
## External resources: default properties for computing external resources address
##
## The same rules apply as EBX® FrontServlet properties (see comments).
##
## Each property may be inherited from EBX® FrontServlet.
#####################################################################

#ebx.externalResources.useLocalUrl=true
#ebx.externalResources.http.host=
#ebx.externalResources.http.port=
#ebx.externalResources.http.path=
#ebx.externalResources.https.host=
#ebx.externalResources.https.port=
#ebx.externalResources.https.path=
```

Proxy mode

Proxy mode allows using a front-end HTTP server to provide static resources (images, CSS, JavaScript, etc.). This architecture reduces the load on the application server for static HTTP requests. This configuration also allows using SSL security on the front-end server.

The web server sends requests to the application server according to a path in the URL. The `servletAlias` and `uiServletAlias` paths are specified in the main configuration file.

The web server provides all external resources. These resources are stored in a dedicated directory, accessible using the `resourcesAlias` path.

EBX® must also be able to access external resources from the file system. To do so, the property `ebx.webapps.directory.externalResources` must be specified.

The main configuration file is configured as follows:

```
################################################
## Path for external resources if they are not
## delivered within web applications
## This field is mandatory if in proxy mode.
################################################
ebx.webapps.directory.externalResources=
D:/http/resourcesFolder

################################################

#ebx.servlet.useLocalUrl=true
#ebx.servlet.http.host=
#ebx.servlet.http.port=
ebx.servlet.http.path= servletAlias
ebx.servlet.http.ui.path= uiServletAlias
#ebx.servlet.https.host=
#ebx.servlet.https.port=
ebx.servlet.https.path= servletAlias
ebx.servlet.https.ui.path= uiServletAlias

################################################

#ebx.externalResources.useLocalUrl=true
#ebx.externalResources.http.host=
#ebx.externalResources.http.port=
ebx.externalResources.http.path= resourcesAlias
#ebx.externalResources.https.host=
#ebx.externalResources.https.port=
ebx.externalResources.https.path= resourcesAlias
```

Reverse proxy mode

URLs generated by EBX® for requests and external resources must contain a server name, a port number and a specific path prefix.

The properties are configured as follows:

```
################################################
ebx.servlet.http.host= reverseDomain
#ebx.servlet.http.port=
ebx.servlet.http.path=ebx/
#ebx.servlet.http.ui.path=ebx-ui/
ebx.servlet.https.host= reverseDomain
#ebx.servlet.https.port=
ebx.servlet.https.path=ebx/
#ebx.servlet.https.ui.path=ebx-ui/
################################################
## Web parameters (for external resources)
## if nothing is set, values are taken from servlet.
################################################
ebx.externalResources.http.host= reverseDomain
#ebx.externalResources.http.port=
#ebx.externalResources.http.path=ebx/
ebx.externalResources.https.host= reverseDomain
#ebx.externalResources.https.port=
ebx.externalResources.https.path=ebx/
```

7.16 **Configuring failover**

These parameters are used to configure the failover mode and activation key, as well as heartbeat logging in DEBUG mode.

See also *Failover with hot-standby*

```
################################################
## Mode used to qualify the way in which a server accesses the repository.
## Possible values are: unique, failovermain, failoverstandby.
## Default value is: unique.
```

```
################################################
#ebx.repository.ownership.mode=unique

## Activation key used in case of failover. The backup server must include this
## key in the HTTP request used to transfer exclusive ownership of the repository.
## The activation key must be an alphanumeric ASCII string longer than 8 characters.
#ebx.repository.ownership.activationkey=

## Specifies whether to hide heartbeat logging in DEBUG mode.
## Default value is true.
#ebx.repository.ownership.hideHeartBeatLogForDebug=true
```

7.17 Tuning the EBX® repository

Some options can be set so as to optimize memory usage.

The properties are configured as follows:

```
################################################################
## Technical parameters for memory and performance tuning
################################################################
# Import commit threshold allows to specify the commit threshold
# exclusively for the archive import launched directly from Manager.
#
# For more details about the commit threshold,
# see the JavaDoc ProcedureContext.setCommitThreshold().
# Default value is 0.
#
ebx.manager.import.commit.threshold=100
```

See also *Validation report page* *[p 60]*

7.18 Miscellaneous

Activating data workflows

This parameter specifies whether data workflows are activated. This parameter is not taken into account on the fly. The server must be restarted whenever the value changes.

```
################################################
## Workflow activation.
## Default is false.
################################################
ebx.workflow.activation = true
```

Disabling user task legacy mode

This parameter specifies whether the creation service of a user task in legacy mode should be offered in the workflow modeling. The default value is `true`.

See `UserTask.UserTaskMode.LEGACY_MODE`[API] for more information.

```
## Disables legacy work item mode(default is true)
## Specify if the creation service of user task in legacy mode must be offered
## in workflow modeling.
#ebx.manager.workflow.legacy.userTaskMode=false
```

Disabling hierarchy plan view

This parameter specifies whether the hierarchy plan view is hidden. The default value is `true`.

```
## Activate or deactivate Workflow hierarchy plan view
ebx.manager.workflow.hierarchyPlanView.hidden=false
```

Log procedure starts

This parameter specifies whether starts of the procedure execution are logged.

```
################################################
## Specifies whether transaction starts are logged. Default is false.
################################################
ebx.logs.logTransactionStart = true
```

Log validation starts

This parameter specifies whether starts of datasets validation are logged.

```
################################################
## Specifies whether validation starts are logged. Default is false.
################################################
ebx.logs.logValidationStart = true
```

Request duration threshold for logs

This parameter specifies in milliseconds the threshold of duration of `Request`[API] and `Query`[API] to be logged. Logs are generated if logging category `ebx.log4j.category.log.request` level is not higher than `INFO`. If the level is `DEBUG`, all `Request`[API] and `Query`[API] are logged.

```
################################################
## Specifies in milliseconds the threshold of duration of Requests and Queries
## to be logged
## Default value is 1000 ms.
## If unset, the default value is used.
################################################
#ebx.logs.request.durationThreshold=1000
```

Request duration threshold for logs

This parameter specifies in milliseconds the delay between 2 logs for `Request`[API] and `Query`[API] that goes beyond the threshold of duration. If this value is greater than 0, and the query duration goes beyond the threshold of duration, it will be logged again repeatedly with at least this delay between each log. As log messages include duration, this is useful to track long queries duration.

```
################################################
## Specifies in milliseconds the delay between 2 logs for Requests and Queries that goes
## beyond the threshold of duration. If this value is greater than 0, and the query duration
## goes beyond the threshold of duration, it will be logged again repeatedly with at least
## this delay between each log.
## Default value is 30000 ms.
## If unset, the default value is used.
################################################
#ebx.logs.request.logAgainEvery=30000
```

Deployment site identification

This parameter allows specifying the email address to which technical log emails are sent.

```
################################################
## Unique Site Name
## --> used by monitoring emails and by the repository
################################################
ebx.site.name= name@domain.com
```

Dynamically reloading the main configuration

Some parameters can be dynamically reloaded, without restarting EBX®. The parameter `thisfile.checks.intervalInSeconds` indicates how frequently the main configuration file is checked.

```
################################################
### Checks if this file has been updated
### If value <= 0, no more checks will be done
################################################
thisfile.checks.intervalInSeconds=1
```

In development mode, this parameter can be set to as low as one second. On production systems, where changes are expected to be less frequent, the value can be greater, or set to '0' to disable hot reloading entirely.

This property is not always supported when the module is deployed as a WAR, as it would then depend on the application server.

Declaring modules as undeployed

On application server startup, the initialization of deployed web applications / EBX® modules and the initialization of the EBX® repository are performed asynchronously. In order to properly initialize the EBX® repository, it is necessary to compile all the data models used by at least a dataset, hence EBX® will wait endlessly for referenced modules to be registered.

If a module is referenced by a data model but is not deployed (or no longer deployed), it is necessary to declare this module as undeployed to unlock the wait and continue the startup process.

> **Note**
>
> The `kernel` logging category indicates which modules are awaited.

> **Note**
>
> A module declared as undeployed cannot be registered into EBX® until it is removed from the property `ebx.module.undeployedModules`.

> **Note**
>
> Any data model based on an unregistered module will have an "undeployed module" compilation error.

See also

Module registration

Dynamically reloading the main configuration [p 57]

```
################################################
## Comma-separated list of EBX® modules declared
## as undeployed.
## If a module is expected by the EBX® repository but is
## not deployed, it must be declared in this property.
## Caution:
## if the "thisfile.checks.intervalInSeconds" property is deactivated,
## a restart is mandatory, otherwise it will be hot-reloaded.
################################################
ebx.module.undeployedModules=
```

Module public path prefix

EBX® modules' public paths are declared in the 'module.xml' file of each module. A context prefix can be declared for all modules, without having to modify the 'module.xml' content, by specifying the property that follows.

This prefix will apply to any EBX® module, including core, add-on and specific modules. It does not apply to 'ebx.war' nor to 'ebx-dataservices.war' web applications, since they do not define an EBX® module.

```
######################################################################
## EBX® Module context path prefix
##
## If defined, applies to all EBX® modules public paths declared in
## any module.xml file (core, add-on and specific).
######################################################################
#ebx.module.publicPath.prefix=aPrefix/
```

EBX® run mode

This property defines how EBX® runs. Three run modes are available: *development,integration* and *production*.

When running in *development* mode, the *development tools* are activated in EBX®, some features thus become fully accessible and more technical information is displayed.

> **Note**
>
> The administrator can always access this information regardless of the mode used.

The additional features accessible when running in *development* mode include the following (non-exhaustive list):

Documentation pane	In the case of a computed value, the Java class name is displayed. A button is displayed giving access to the path to a node.
Compilation information	Module and schema compilation information is displayed in the dataset validation report.
Java bindings	The generation of Java bindings is available if the schema of the dataset mentions at least one binding.
Web component link generator	The Web component link generator is available on datasets and dataspaces.
Data model assistant	Data model configuration and additional options, such as Services, Business Objects and Rules, Java Bindings, Toolbars and some advanced properties.
Workflow modeling	Declare specific script tasks.
Log	The logs include additional technical information intended for the developer. For example, a warning is written to logs if a drop-down list is defined on a node which is not an enumeration in a UI Bean.
Product documentation	The product documentation is always the most complete one (i.e "advanced"), including administration and development chapters.

```
################################################
## Server Mode
## Value must be one of: development, integration, production
## Default is production.
################################################
backend.mode=integration
```

> **Note**
>
> There is no difference between the *integration* and *production* modes.

Resource filtering

This property allows the filtering of certain files and directories in the resource directory contents (resource type node, with an associated facet that indicates the directory that contains usable resources).

```
################################################
## list (separated by comma) of regexps excluding resource
## the regexp must be of type "m:[pattern]:[options]".
## the list can be void
################################################
```

```
ebx.resource.exclude=m:CVS/*:
```

Validation report page

The validation report page can display a finite number of items for each severity. This number can be tuned with this property.

```
##################################################
## Defines the maximum item displayed for each severity in the validation report page.
## Default value is 100.
##################################################
ebx.validation.report.maxItemDisplayed=200
```

See also *Tuning the EBX® repository [p 55]*

Validation report logs

This property allows to specify the number of validation messages to display in the logs when validating a dataset or a table.

```
##################################################
## Defines the maximum number of messages displayed in the logs.
## Default value is 100.
## When set to 0 or a negative value, the limit is not considered.
##################################################
ebx.validation.report.maxItemDisplayedInLogs=500
```

See also *Tuning the EBX® repository [p 55]*

CHAPTER 8

Initialization and first-launch assistant

Deliverables can be found on TIBCO eDelivery(an account is mandatory in order to access eDelivery, please contact the support team to request one).

The TIBCO EBX® Configuration Assistant helps with the initial configuration of the EBX® repository. If EBX® does not have a repository installed upon startup and if the automatic installation [p 44] is not enabled, the configuration assistant is launched automatically.

Before starting the configuration of the repository, make sure that EBX® is correctly deployed on the application server. See Java EE deployment [p 13].

> **Note**
>
> The EBX® main configuration file must also be properly configured. See TIBCO EBX® main configuration file [p 43].

This chapter contains the following topics:

8.1 Configuration steps

The EBX® configuration assistant guides you through the following steps:

1. Validating the license agreement.
2. Configuring the repository.
3. Defining users in the default user and roles directory (if a custom directory is not defined).
4. Validating the information entered.
5. Installing the EBX® repository.

Validating the license agreement

In order to proceed with the configuration, you must read and accept the product license agreement.

Configuring the repository

This page displays some of the properties defined in the EBX® main configuration file. You also define several basic properties of the repository in this step.

Id of the repository (`repositoryId`)	Must uniquely identify the repository (in the scope of the enterprise). The identifier is 48 bits (6 bytes) long and is usually represented as 12 hexadecimal digits. This information is used for generating the Universally Unique Identifiers (UUIDs) of entities created in the repository, and also of transactions logged in the history. This identifier acts as the "UUID node", as specified by RFC 4122.
Repository label	Defines a user-friendly label that indicates the purpose and context of the repository.

See also *TIBCO EBX® main configuration file* *[p 43]*

Defining users in the default directory

If a custom user and roles directory is not defined in the EBX® main configuration file, the configuration assistant allows to define default users for the default user and roles directory.

An administrator user must be defined. You may optionally create a second user.

See also *Users and roles directory*

Validating the information entered

Before proceeding with the installation of the repository, you can review the configuration of the repository and the information entered on the 'Configuration Summary' page. If you need to modify information, you can return to the previous pages using the configuration assistant **< Back** button.

Once you have verified the configuration, click the button **Install the repository >** to proceed with the installation.

Installing the EBX® repository

The repository installation is performed using the provided information. When the installation is complete, you are redirected to the repository login page.

CHAPTER 9

Deploying and registering TIBCO EBX® add-ons

> **Note**
>
> Refer to the documentation of each add-on for additional installation and configuration information in conjunction with this documentation.

This chapter contains the following topics:

9.1 Deploying an add-on module

> **Note**
>
> **Each add-on bundle version is intended to run with a specific EBX® version and all its fix releases. Make sure that the EBX® and add-on bundle versions are compatible, otherwise the add-on registration will abort.**

The web application deployment descriptor for the add-on module must specify that class definitions and resources from the web application are to be loaded in preference to classes from the parent and server classloaders.

For example, on WebSphere Application Server, this can be done by setting `<context-priority-classloader>true</context-priority-classloader>` in the `web-app` element of the deployment descriptor.

On WebLogic, include `<prefer-web-inf-classes>true</prefer-web-inf-classes>` in `weblogic.xml`.

See the documentation on class loading of your application server for more information.

The EBX® add-on common JAR file, named `lib/ebx-addons.jar`, must be copied in the library directory shared by all web applications.

> **Note**
>
> The add-on log level can be managed in the main configuration file [p 50].

9.2 Registering an add-on module

Registering an add-on makes its configuration available in the admin section. Add-on features are only available to end-users when the add-on is also activated [p 64].

To register a new EBX® add-on in the repository:

1. Navigate to the 'Administration' area.
2. Click the down-arrow in the navigation pane and select **Technical configuration** > **Add-ons registration**.
3. On the **Registered add-ons** page, click the + button to create a new entry.
4. Select the add-on you are registering.
5. Click on **Save**.

> **Note**
>
> Unregistering an add-on will not delete any existing configuration, but will make it available in the UI until the add-on is registered again.

9.3 Activating an add-on module

Activating an add-on makes its features available to the end-users. Only registered add-ons can be activated.

To activate an EBX® add-on in the repository:

1. Navigate to the 'Administration' area.
2. Click the down-arrow in the navigation pane and select **Technical configuration** > **Add-ons registration**.
3. Select the registered add-on you are activating and enable the 'Activation' field.
4. Click on **Save**.

9.4 Deleting an add-on module

To delete an add-on module from the EBX® repository:

1. Navigate to the 'Administration' area.
2. Click the down-arrow in the navigation pane and select **Technical configuration** > **Add-ons registration**.
3. On the **Registered add-ons** page, tick the box corresponding to the add-on to be deleted.
4. In the 'Actions' menu, select 'Delete'.
5. Close and purge the Administration datasets related to the previously used add-on, as well as the including dataspaces.

When an add-on is no longer deployed, a dataspace corresponding to the Administration dataset will then appear in the list of Reference children under the dataspaces. When an add-on module is no longer deployed, it is thus necessary to close/delete and purge manually all data/dataspaces related to the add-on.