TIBCO EBX®
Installation Guide

Version 6.1.5
June 2025

TIBCO®

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 on computers only; it is not supported on mobile devices:

Microsoft Edge Chromium	As Microsoft Edge Chromium is updated frequently and it is not possible to deactivate automatic updates, Cloud Software Group, Inc. only tests and makes the best effort to support the latest version available.
Mozilla Firefox ESR	As Mozilla Firefox ESR is updated frequently and it is not possible to deactivate automatic updates, Cloud Software Group, Inc. only tests and makes the best effort to support the latest version available.
Google Chrome	As Google Chrome is updated frequently and it is not possible to deactivate automatic updates, Cloud Software Group, 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 cause issues. It is thus imperative to use the action buttons and links offered by EBX® instead of refreshing the page.

'Previous' and 'Next' buttons

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

Zoom troubleshooting

Zooming in or out may cause some minor display issues (for example extra scrollbar or misalignment). Those issues can be fixed by refreshing the screen using the provided navigation links.

Browser configuration

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

- JavaScript
- Ajax
- Pop-ups
- Cookies

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 17 or 21 LTS.

  > **Note**
  >
  > JRE 11 is no longer supported. We recommend upgrading your JRE to a recent LTS version to take advantage of its performance improvements.

  > **Note**
  >
  > The Unicode CLDR has changed with JRE versions and impacts locale-sensitive formats such as integer or decimal representations. Configure the search order of locale sensitive services using the `java.locale.providers` system property. You can use the value `COMPAT` to ensure compatibility with JRE 8. See java.util.spi.LocaleServiceProvider for more details.

- Any Servlet/JSP container that complies with Servlet 3.1 up to, but not including, 5.0. For example, Tomcat 9.0 up to, but not including, 10.0. Any Java Application Server that supports Java SE 17 or 21 LTS and does not use Jakarta EE 9 or greater. For example, WebSphere Liberty 23 or higher for Java EE 8, WebLogic Application Server 14c 14.1.2 (2024) or higher, JBoss EAP 7.4 Update 8+ or higher. See Java EE deployment overview [p 19].
- The application server must support the JSON Processing 1.1 (JSR 374), or allow the uses of the implementation embedded in the `ebx.jar` library. For example, Tomcat does not provide any library to support this specification (only the embedded one can be used), WebLogic Application Server 14c or higher supports this specification, WebSphere Application Server Liberty and JBoss EAP allow including or excluding 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 instruct 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.

1.3 **Supported databases**

The EBX® repository supports the relational database management systems listed below, with the 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 19c, 23c or higher.	The distinction of `null` values bears 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 13 or higher.	The user with which EBX® connects to the database needs the CONNECT privilege on the database hosting the EBX® repository. Other than this, the default privileges on the public schema of this database are suitable. Also, see this *limitation* regarding the evolution of datamodels in mapped modes.
Amazon Aurora PostgreSQL (compatible with PostgreSQL 13 or higher).	The comments in the above section for PostgreSQL apply.
Google Cloud SQL for PostgreSQL (compatible with PostgreSQL 13 or higher).	The comments in the above section for PostgreSQL apply.
Microsoft Azure Database for PostgreSQL (compatible with PostgreSQL 16 or higher).	The comments in the above section for PostgreSQL apply.
Microsoft SQL Server 2014 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 history tables. The default database collation can be specified when the database is created. Otherwise, the collation of the database server is used. To avoid naming conflicts or unexpected

	behaviors, a case- and accent-sensitive collation must be used as the default database collation (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 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 must be used as the default database collation (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 2.2.224 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 17]*

Configuring the EBX® repository *[p 48]*

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 4]

Refer to the *Security Best Practices* for information on navigating secure deployments.

2.2 Software components

EBX® uses the following components:

- Library `ebx.jar`
- Embedded [p 12] and required [p 12] third-party Java libraries
- EBX® built-in web applications [p 15] and optional custom web applications [p 15]
- EBX® main configuration file [p 45]
- *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 4]*

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
- Apache Log4j 2 API
- ArcGIS API for JavaScript
- Javax Activation
- Javax Annotations
- Javax JSON Binding
- Javax SAAJ API
- Javax WS RS
- Javax XML Bind
- LZ4 compression for Java
- MicroProfile OpenAPI
- Simple Logging Facade for Java (SLF4J) API

For more information regarding the versions or the details of the Third-Party Library, please refer to the: `TIB_ebx_6.1.5_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 19].

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 2.2.224 validated. Note that H2 is not supported in production environments. https://www.h2database.com/ If the H2 repository has been built with version 1.0 driver, this migration to v2 process must be applied.
Oracle JDBC	Oracle database 23ai 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 `ojdbc11.jar` certified library with JDK 11. Oracle database JDBC drivers download.
SQL Server JDBC	SQL Server 2014 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-12.6.1.jre11.jar` library with JDK 11. Download Microsoft JDBC Driver 12.6.1 for SQL Server (zip).
PostgreSQL	PostgreSQL Java 8 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 17]*

Configuring the EBX® repository *[p 48]*

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 1.5.6 of this API. See Activating and configuring SMTP and emails [p 55] 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)

SSL features activation, configuration, and management are tightly linked to the web application server used. Consequently, you should refer to the appropriate web application server documentation. For example, Tomcat SSL/TLS Configuration How-To.

See also *TIBCO EBX® main configuration file* *[p 45]*

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 45]*

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.

Attention

Web application deployment on `/` path context is no more supported. The path context must not be empty nor equals to `/`. Moreover, web applications deployment on paths of different depth is deprecated. Every web application path context must be set on the same path depth.

For more information, see the note on repackaging the EBX® EAR [p 20] 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 16] 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-authentication`	EBX® authentication 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 65]

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 45]

Supported application servers [p 5]

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"/>
```

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 48] 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 48]

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 55] 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 56] 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 18] 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 56] 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 56] 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 56] 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 7 to Jakarta EE 8 application server that supports Servlet 3.1, up to but not including version 5.0. The following documentation on Java deployment and installation notes are available:

- Installation note for Tomcat 9.x [p 33]
- Installation note for JBoss EAP 7.4.x [p 21]
- Installation note for WebSphere Application Server Liberty 23.x [p 27]

- Installation note for WebLogic 14c [p 39]

Attention

- The EBX® installation notes on Java EE 7 to Jakarta EE 8 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 custom or add-on 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 7.4.x

This chapter contains the following topics:

3.1 Overview

Attention

- This chapter describes a *quick installation example* of TIBCO EBX® on JBoss 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.
- The complete description of the components needed by EBX® is given in chapter Java EE deployment [p 11].
- 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.
- Refer to the *Security Best Practices* for information on navigating secure deployments.

3.2 Requirements

- Java SE 17 LTS
- JBoss Application Server EAP 7.4 (update 8 and above)
- Database and JDBC driver
- EBX® CD
- EBX® license key
- No CDI features in EBX®'s additional modules (since CDI will be automatically disable)

See also *Supported environments* *[p 4]*

3.3 JBoss Application Server installation

This quick installation example is performed for an unix operating system.

1. Download JBoss EAP 7.4.0 `Installer` jar version from:
   https://developers.redhat.com/products/eap/download/
2. Run the `Installer` using *java -jar* command line.
   For further installation details, 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 the 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' in the JBoss EAP 7.4 installation root directory,
   12. Click 'done'.
4. Download and apply the last update for JBoss EAP 7.4.0, this link requires an authentication:
   `https://access.redhat.com/jbossnetwork/restricted/listSoftware.html?downloadType=patches&product=appplatform&version=7.4`

3.4 EBX® home directory 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 file:

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

3. Create an `ebx-license.properties` file in *EBX_HOME* as shown below:

   `/opt/ebx/home/ebx-license.properties`

   The content of the file:

```
################################################################
## EBX® License Key
##    Ensure the file is writable by EBX®
##    to enable updates using the UI license key wizard process.
################################################################
ebx.license=<your_license_key>
```

4. If needed, edit the `ebx.properties` file to override the default database. By default the standalone H2 database is defined. The property key `ebx.persistence.factory` must be uncommented for other supported databases and the `h2.standalone` one must be commented.

3.5 JBoss Application Server and Java Virtual Machine configuration

1. 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).
2. Add 'ebx.properties' and 'ebx.home' properties to the 'JAVA_OPTS' environment variable respectively set with `ebx.properties` file's path and *EBX_HOME* directory's path.
3. Set the 'JBOSS_MODULES_SYSTEM_PKGS' environment variable like the following:

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

4. Copy from the *EBX® CD*, the `ebx.software/lib/ebx-lz4.jar` [p 12] Data compression library to a dedicated directory (for example `<JBOSS_HOME>/compress`).
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 like the following:

```
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
```

3.6 **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"/>
```

   Caution: 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"/>
```

3.7 **Data source and JDBC provider configuration**

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=javaee.api,sun.jdk,ibm.jdk,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, ensure 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.xa.PGXADataSource\
    )
```

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

```
data-source add \
  --name=jdbc/EBX_REPOSITORY \
  --jndi-name=java:/EBX_REPOSITORY \
  --driver-name=postgresql \
  --connection-url=jdbc:postgresql://<SERVER_NAME>:<PORT>/<DATABASE_NAME> \
  --user-name=<PERSISTENCE_USER> \
  --password=<PERSISTENCE_PASSWORD>
```

3.8 EBX.ear application update

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 and additional web modules.

   Mail: see SMTP and emails [p 14] for more information.

   SSL: see Secure Socket Layer (SSL) [p 14] for more information.

   JMS: see Java Message Service (JMS) [p 14] for more information.
3. Update the `/META-INF/application.xml` and `/META-INF/jboss-deployment-structure.xml` files according to the added additional web modules.
4. Compress anew the ear archive.

3.9 EBX.ear application deployment

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

3.10 EBX® application start

1. After the launch of the JBoss Application Server, with the `<JBOSS_HOME>/bin/standalone.sh -c standalone-full.xml` command line or through the service command, run the EBX® web application by entering the following URL in the browser: http://localhost:8080/ebx/.
2. At first launch, EBX® Wizard [p 71] helps to configure the default properties of the initial repository.

CHAPTER 4

Installation note for WebSphere Application Server Liberty 23.x

This chapter contains the following topics:

4.1 Overview

Attention

- This chapter describes a *quick installation example* of TIBCO EBX® on WebSphere Application Server Liberty.
- 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 chapter Java EE deployment [p 11].
- 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 <WLP_HOME> to refer to the WebSphere Application Server Liberty installation directory, and from which relative paths are resolved.
- Refer to the *Security Best Practices* for information on navigating secure deployments.

4.2 Requirements

- Java SE 17 or 21 LTS. We recommended using IBM Semeru, or Eclipse Adoptium Java distributions.
- WebSphere Application Server Liberty 23.0.0.10 and above for Java EE 8
- Database and JDBC driver
- EBX® CD
- EBX® license key
- No CDI features in EBX®'s additional modules (since CDI will be automatically disabled)

See also *Supported environments* *[p 4]*

4.3 WebSphere Application Server Liberty installation

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

1. Download WebSphere Application Server Liberty 23.x latest version (zip distribution) from:
   https://www.ibm.com/support/pages/23001-websphere-application-server-liberty-23001 .
   The following example uses *Liberty with Java EE 8 Web Profile* zip distribution.
2. Extract the content of this package to a directory on your local machine.
3. Create a new application server (example: 'ebxServer') by running the command:
   ```
   <WLP_HOME>/wlp/bin/server create ebxServer
   ```
4. Edit `<WLP_HOME>/usr/servers/<SERVER_NAME>/server.xml` file. In our example, <SERVER_NAME> is 'ebxServer'.

- Replace the `<feature>webProfile-8.0</feature>` tag with the following lines:

```
<feature>javaMail-1.5</feature>
<feature>jdbc-4.2</feature>
<feature>jndi-1.0</feature>
<feature>servlet-4.0</feature>
```

- Add the following line into the `<server>` tag:

```
<webContainer deferServletLoad="false"  allowQueryParamWithNoEqual="true"/>
```

> **Note**
>
> The `deferServletLoad` attribute must be set to `false` since EBX® requires servlets to be initialized on application server startup. By default, WebSphere Application Server Liberty defers servlets loading until a request is received for the associated web application. See Specifying when servlets are loaded and initialized for more information.

> **Note**
>
> The `allowQueryParamWithNoEqual` attribute must be set to `true` since, according to the Servlet API specification, a value of a request parameter must be returned as a `String`, or `null` if the parameter does not exist. Thus, EBX® expects a non `null` value for an existing request parameter.

4.4 EBX® home directory 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 file:

   `/opt/ebx/home/ebx.properties`.
3. Create an `ebx-license.properties` file in *EBX_HOME* as shown below:

   `/opt/ebx/home/ebx-license.properties`

   The content of the file:

```
######################################################################
## EBX® License Key
##    Ensure the file is writable by EBX®
##    to enable updates using the UI license key wizard process.
######################################################################
ebx.license=<your_license_key>
```

4. If needed, edit the `ebx.properties` file to override the default database. By default the standalone H2 database is defined. The property key `ebx.persistence.factory` must be uncommented for other supported databases and the `h2.standalone` one must be commented.

4.5 EBX® and third-party libraries deployment

1. Create the *EBX_LIB* directory. For example, `/opt/ebx/home/lib`.
2. Copy third-party libraries from the *EBX® CD*, or from other sources, to the *EBX_LIB* directory. In our example and for a PostgreSQL database, we will get:

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

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

The complete description of these components is given in the chapter Java EE deployment [p 11]. If those components are already deployed on the class-loading system, they do not have to be duplicated (ex: javax.mail 1.5.x is already present since `javaMail-1.5` feature has been activated for this server).

4.6 JNDI entries configuration

1. For *JMS* configuration on WebSphere Application Server Liberty, see Configuring JMS connection factories.
2. For *JavaMail* configuration on WebSphere Application Server Liberty, see Administering JavaMail on Liberty.

4.7 Data source and JDBC provider configuration

1. To create the JDBC driver library, edit the `<WLP_HOME>/usr/servers/<SERVER_NAME>/server.xml` file by adding the following lines into the `<server>` tag:

```
<library id="jdbcDriver">
  <file name="${EBX_LIB}/To be completed"/>
</library>
```

Caution: the `name` attribute value, from the `file` tag, has to be completed. In our example, it should be `${EBX_LIB}/postgresql-X.X.X-driver.jar`.

2. To create the JDBC data source, edit the same file and same XML tag by adding the following lines:

```
<dataSource id="ebxRepository" jndiName="jdbc/EBX_REPOSITORY">
  <jdbcDriver libraryRef="jdbcDriver"/>
  <properties.toBeCompleted/>
</dataSource>
```

Caution: the `properties` tag, has to be completed. Since the kind of tag is strongly reliant on the JDBC driver used, a default configuration can not be provided. For a PostgreSQL driver, it may looks like:

```
<properties.postgresql databaseName="EBXDB" serverName="localhost" portNumber="5432" user="ebx"
 password="ebx"/>
```

> **Note**
>
> To hide the password for a JDBC connection in the `server.xml` file of WebSphere Application Server Liberty, refer to the documentation Configuring authentication aliases for Liberty.

4.8 Java Virtual Machine configuration

1. Set server's environment variables by adding the following lines into `<WLP_HOME>/usr/servers/<SERVER_NAME>/server.env` file:

```
EBX_HOME="<path_to_the_directory_ebx_home>"
EBX_LIB="<path_to_the_directory_ebx_lib>"
JAVA_HOME="<path_to_the_java_home>"
```

2. Set Java options by creating a new `jvm.options` file under the `<WLP_HOME>/usr/servers/<SERVER_NAME>/` directory. This file will hold, at least, the following lines:

```
-Debx.home="${EBX_HOME}"
-Debx.properties="${EBX_HOME}/ebx.properties"
```

4.9 **EBX® application deployment**

1. Create the `<EBX_HOME>/ear/` directory.
2. 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`
3. To create the EBX® library, edit the `<WLP_HOME>/usr/servers/<SERVER_NAME>/server.xml` file by adding the following lines into the `<server>` tag:

```
<library id="ebxLibLz4">
  <file name="${EBX_LIB}/ebx-lz4.jar"/>
</library>
```

> **Note**
>
> WebSphere Application Server Liberty doesn't support `LZ4 JNI implementation`.

4. To deploy EBX.ear, edit the `<WLP_HOME>/usr/servers/<SERVER_NAME>/server.xml` file by adding the following lines into the `<server>` tag:

```
<application id="ebxApp" location="${EBX_HOME}/ear/{ebx.ear.name}.ear" name="ebxApp" type="ear">
  <classloader commonLibraryRef="ebxLibLz4" />
</application>
```

4.10 **EBX® application start**

1. After the launch of the WebSphere Application Server Liberty, with the `<WLP_HOME>/wlp/bin/server start ebxServer` command line, run the EBX® web application by entering the following URL in the browser:

   http://localhost:9080/ebx/
2. At first launch, EBX® Wizard [p 71] helps to configure the default properties of the initial repository.

CHAPTER 5

Installation note for Tomcat 9.x

This chapter contains the following topics:

5.1 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 or greater is not supported.
- The complete description of the components needed by EBX® is given in chapter Java EE deployment [p 11].
- 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 `$CATALINA_HOME` to refer to the Tomcat installation directory, and from which most relative paths are resolved. However, if the `$CATALINA_BASE` directory has been set for a multiple instances configuration, it should be used for each of these references.
- Refer to the *Security Best Practices* for information on navigating secure deployments.

5.2 Requirements

- Java SE 17 or 21 LTS
- Apache Tomcat 9.x
- Database and JDBC driver
- EBX® CD
- EBX® license key

See also *Supported environments* *[p 4]*

5.3 Tomcat Application Server installation

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

5.4 EBX® home directory configuration

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 have the following file:

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

3. Create an `ebx-license.properties` file in *EBX_HOME* as shown below:

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

   The content of the file:

```
################################################################
## EBX® License Key
##    Ensure the file is writable by EBX®
##    to enable updates using the UI license key wizard process.
################################################################
ebx.license=<your_license_key>
```

4. If needed, edit the `ebx.properties` file to override the default database. By default the standalone H2 database is defined. The property key `ebx.persistence.factory` must be uncommented for other supported databases and the `h2.standalone` one must be commented.

5.5 **Tomcat Application Server and Java Virtual Machine configuration**

1. Modify `$CATALINA_HOME/conf/server.xml` (or `$CATALINA_BASE/conf/server.xml`) file.

   Add the attribute `encodedSolidusHandling="passthrough"` to the `Connector` element.

   Add the following line to the `<Host>` element:

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

   In our example, we will have:

   `<Host name=...>`

   `... ...`

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

   `... ...`

   `</Host>`

2. Modify the `$CATALINA_HOME/conf/catalina.properties` (or `$CATALINA_BASE/conf/catalina.properties`) file by adding the following lines to the `tomcat.util.scan.DefaultJarScanner.jarsToSkip` property:

   `ebx.jar,\`

   `ebx-addons.jar,\`

   `ebx-lz4.jar,\`

3. Configure the Java Virtual Machine 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% -Dorg.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH=true %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 they have been configured. Otherwise this piece of text must be replaced by the Tomcat installation directory's path.

- 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 they have been configured. Otherwise this piece of text must be replaced by the Tomcat installation directory's path.

- 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} -Dorg.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH=true
 ${JAVA_OPTS}"
export CLASSPATH="<$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 they have been configured. Otherwise this piece of text must be replaced by the Tomcat installation directory's path.

> **Note**
>
> The Tomcat option `-Dorg.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH=true` avoid blocking URLs containing encoded backslashes.

Caution: 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`.

> **Note**
>
> For a Data compression library [p 12] native installation, ensure to only reference it in the *CLASSPATH* environment variable.

5.6 EBX® and third-party libraries deployment

1. Copy third-party libraries from the *EBX® CD* to `$CATALINA_HOME/lib/` (or `$CATALINA_BASE/lib/`) directory, except for the Data compression library [p 12]. 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/h2-2.2.224.jar` (default persistence factory) coming from `ebx.software/lib/lib-h2` directory.

The exact description of these components is given in chapter Software components [p 11]. Obviously, if those components are already deployed on the class-loading system, they do not have to be duplicated.

2. Create a directory dedicated to the Data compression library [p 12] (for example `$CATALINA_HOME/compress` or `$CATALINA_BASE/compress`) and copy it there.

> **Note**
>
> Ensure that the library is copied in the directory pointed out by the previously updated *CLASSPATH* environment variable.

3. 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`

5.7 EBX® web applications deployment

1. 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

   `$CATALINA_HOME/webapps/ebx-authentication.war`: authentication application

5.8 EBX® application start

1. After Tomcat launch, run EBX® web application by entering the following URL in the browser: http://localhost:8080/ebx/
2. At first launch, EBX® Wizard [p 71] helps to configure the default properties of the initial repository.

CHAPTER 6

Installation note for WebLogic 14c

This chapter contains the following topics:

6.1 Overview

Attention

- WebLogic 14c supports JDK 11, but not JDK 17 or 21 LTS. Until a WebLogic release supports JDK 17 or 21 LTS, EBX® will continue to support WebLogic 14c with JDK 11. Please be aware that, according to Oracle's Statement of Direction, future versions of supported software might not be compatible with EBX®.
- This chapter describes a *quick installation example* of TIBCO EBX® on WebLogic 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.
- The complete description of the components needed by EBX® is given in chapter Java EE deployment [p 11].
- 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.
- Refer to the *Security Best Practices* for information on navigating secure deployments.

6.2 Requirements

- Certified Oracle Java SE 11 (update 6 and above)
- WebLogic Server 14c
- Database and JDBC driver
- EBX® CD
- EBX® license key

See also *Supported environments* *[p 4]*

6.3 WebLogic Application Server installation

1. Download WebLogic 14c latest version from:
   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 directory, 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.

   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 EBX® home directory configuration

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 have the following file:

   `C:\EBX\home\ebx.properties`, or `/home/ebx/ebx.properties`
3. Create an `ebx-license.properties` file in *EBX_HOME* as shown below:

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

   The content of the file:

```
#################################################################
## EBX® License Key
##    Ensure the file is writable by EBX®
##    to enable updates using the UI license key wizard process.
#################################################################
ebx.license=<your_license_key>
```

4. If needed, edit the `ebx.properties` file to override the default database. By default the standalone H2 database is defined. The property key `ebx.persistence.factory` must be uncommented for other supported databases and the `h2.standalone` one must be commented.

6.5 WebLogic Application Server and Java Virtual Machine configuration

1. 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}"
```

2. 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"
```

6.6 EBX® and third-party libraries deployment

1. Copy third-party libraries from the *EBX® CD* to the `<DOMAIN_HOME>/lib` directory except for the Data compression library [p 12]. In our example, for an H2 standalone data base, we will have:

   `<DOMAIN_HOME>/lib/h2-2.2.224.jar` (default persistence factory) coming from `ebx.software/lib/lib-h2` directory.

   The complete description of the components needed by EBX® is given in chapter Java EE deployment [p 11]. 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 is already present in the WebLogic Server).

2. Create a directory dedicated to the Data compression library [p 12] (for example `<DOMAIN_HOME>/compress`) and copy it there.

   > **Note**
   >
   > Ensure that the library is copied in the directory pointed out by the previously updated *PRE_CLASSPATH* environment variable.

6.7 Data source and JDBC provider configuration

1. Start the 'Administration server' (for example 'AdminServer'), using:

   `<DOMAIN_HOME>/bin/startWebLogic.sh`

2. 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

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

   1. Set: Type Name: `EBX_REPOSITORY`, JNDI Name: `EBX_REPOSITORY`, Database Type: *Your database type*

      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'), using:

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

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

6.8 **EBX® application deployment**

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 in `C:\EBX\home\`, or `/home/ebx/` directory and click 'Next'
   2. Choose the installation type and scope: Click on 'Install this deployment as an application', 'Global' default scope and click 'Next'
   3. Select the deployment targets: Select a node (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 'Start' > 'Servicing all requests'
7. Start the application node (for example 'EbxServer'), using:

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

6.9 **EBX® application start**

1. After WebLogic Application Server launch, run the EBX® web application by entering the following URL in the browser:

   http://localhost:7003/ebx/
2. At first launch, EBX® Wizard [p 71] helps to configure the default properties of the 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 16]. Administrators can also perform further configuration through the user interface, which is then stored in the EBX® repository.

See also

Deployment details [p 16]

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 an EBX® license key properties file

See also *Initialization and first-launch assistant [p 71]*

The license key can be retrieved from the TIBCO eDelivery site.

```
#################################################################
## EBX® license file
```

```
## Define the license file using one of the following methods.
## If the license file is defined in multiple locations, the
## order of priority is shown below:
## 1. System property -Debx.license.file=<properties-file>.
## 2. Main ebx.properties configuration 'ebx.license.file'
##    property value.
## 3. Default: ${ebx.home}/ebx-license.properties
##    The file should have an 'ebx.license' property containing
##    the license key.
#################################################################
ebx.license.file=location_of_license_properties
```

The license key properties file must contain.

```
#################################################################
## EBX® License Key
##    Ensure the file is writable by EBX®
##    to enable updates using the UI license key wizard process.
#################################################################
ebx.license=<your_license_key>
```

You can update the license key without restarting the EBX® server. When the server detects a change, it writes a kernel log entry with the license status. If the key is invalid, the license wizard displays when a user accesses the interface.

> **Note**
>
> For a time-limited license key and some time before its expiration, a daily kernel WARN log informs that the license will expire soon.

7.3 **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).
## If true, property ebx.license from ebx.license.file should also be set to a valid license.
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 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 non-encrypted password used for the 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 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 <login> <password_to_encrypt>
ebx.install.admin.password.encrypted=ff297ae08f7eeb63230b55f7c45a720a017bc71d22eaaec...
```

7.4 Setting the EBX® root directory

The EBX® root directory contains the Lucene indexes directory, the archives and, when the repository is persisted on H2 standalone mode, the H2 database files. It also contains the legacy XML audit trail (deprecated).

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

See also *Monitoring and clean up of the file system*

7.5 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 13].

See also

Repository administration

Rules for the database access and user privileges

Supported databases [p 7]

Data source of the EBX® repository [p 17]

Database drivers [p 13]

```
################################################
## 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 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 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 Microsoft SQL Server.
##################################################
#ebx.persistence.factory=sqlserver

## Specific properties to be set 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 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 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
```

7.6 **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
```

7.7 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*

7.8 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 = /tmp/java
ebx.temp.directory = \\${java.io.tmpdir}

## 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
```

7.9 Activating the audit trail

By default, the audit trail logging is deactivated. It can be activated by configuring the "audit" log category to log INFO level messages. For instance the following will log audit trail messages using a specific 'ebxFile' appender [p 53]:

```
ebx.log4j.category.log.audit = INFO, ebxFile:audit
```

See also

Configuring the EBX® logs [p 51]

Audit trail

7.10 Activating the legacy XML audit trail (deprecated)

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 *Legacy XML Audit trail*

7.11 Configuring the EBX® logs

See also *Log configuration*

EBX® main file mechanism

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 55]). **Note:** This category must not use the Custom SMTP appender [p 54] 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 *memory monitoring*.
ebx.log4j.category.log.request	Logs for `Request`[API] and `Query`[API] events in EBX®: executions having a duration exceeding `ebx.logs.request.durationThreshold` milliseconds, requests / queries where the optimization phase exceeds `ebx.logs.request.optimizationThreshold` milliseconds and executions that internally throw an exception while iterating the results. All requests / 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*.
ebx.log4j.category.log.audit	Logs for the *audit trail* feature.

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
##
## Note: Apache Log4j 2 native configuration mechanisms may be used instead, or in combination,
## of these ones. For more information, see 'Logs configuration' section in
## 'Administration overview' chapter from EBX® documentation.
################################################
#ebx.log4j.debug=true
#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
ebx.log4j.category.log.audit=INFO, ebxFile:audit
```

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 size (in megabytes) 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 size (in megabytes) 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 55]*

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`

Migrate from EBX® main file mechanism to Apache Log4j™ 2 and reverse

Supposing that Apache Log4j™ 2 is not configured programmatically, then moving from one approach to another is as simple as cleaning the previously used file, converting every logger configuration into the new format and installing the target file at the proper place.

To define EBX® loggers configuration, refer to Configuring the EBX® logs [p 51].

To define Apache Log4j™ 2 loggers configuration, remove the *ebx.log4j.category* prefix from EBX® loggers' name, replace the *ebx.log4j.rootCategory* configuration by the root logger one and refer to Apache Log4j™ 2 configuration documentation for the remaining.

Considering the underlying EBX® logs configuration:

```
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=%d{yyyy-MM-dd HH:mm:ss.SSS} %-5p %-6pid{[0]} --- [%15.15t]
 %-40.40c{39} : %m%n

ebx.log4j.category.log.kernel=  INFO, Console
ebx.log4j.rootCategory=         INFO, Console
```

The Apache Log4j™ 2 equivalent configuration will be as the following:

```
{
  "configuration": {
    "appenders": {
```

```
      "Console": {
        "name": "Console",
        "PatternLayout": {
          "pattern": "%d{yyyy-MM-dd HH:mm:ss.SSS} %-5p %-6pid{[0]} --- [%15.15t] %-40.40c{39} : %m%n"
        }
      }
    },
    "loggers": {
      "logger": [
        {
          "name": "log.kernel",
          "level": "INFO",
          "AppenderRef": {
            "ref": "Console"
          }
        }
      ],
      "root": {
        "level": "INFO",
        "AppenderRef": {
          "ref": "Console"
        }
      }
    }
  }
}
```

> **Note**
>
> The Custom 'ebxFile' appender [p 53] does not exist natively in Apache Log4j™ 2, but its behavior can be mimic.

> **Note**
>
> Ensure to migrate every logger from one mechanism to another to get as close functional coverage as possible.

7.12 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 17]*

```
################################################
## 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 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=
## SMTP socket connection timeout value in milliseconds (default is 600000).
#ebx.mail.smtp.connectionTimeout=600000
## SMTP socket read timeout value in milliseconds (default is 600000).
#ebx.mail.smtp.timeout=600000
## SMTP socket write timeout value in milliseconds (default is 600000).
#ebx.mail.smtp.writeTimeout=600000

## Activate SSL (true or false, default is false).
```

```
#ebx.mail.smtp.ssl.activate=true
```

7.13 **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

## Specifies the default pagination size value for the select
## operations. This configuration is used by the SOAP connector.
## Default value is 10.
#ebx.dataservices.pagination.pageSize.default=10

## 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
```

7.14 **Activating and configuring JMS**

See also *JMS for data services* *[p 18]*

```
################################################
## 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
```

7.15 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

7.16 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
```

7.17 Activating and configuring staging

Staging is enabled/disabled via configuration. Some options can be set so as to optimize memory and disk usage. The properties are configured as follows:

```
################################################
## Staging configuration
################################################

## Activates staging (true or false, default is true).
#ebx.staging.activated=true

## Defines the max size on disc
## dedicated to staging temporary folders.
## 0 defines an infinite size.
## If unset, the default value is 0.
#ebx.staging.maxTemporaryFolderSizeInBytes=0

## Max size for canonicalization of attachments in bytes.
## Canonicalization consumes memory and CPU on large xml and json for attachments.
## If not set, default value is 10 Mo (10485760).
#ebx.staging.maxCanonicalizationSizeInBytes=10485760

## Defines the maximum number of staging element per list when browsing a repository
## If not defined, the default value is 1000.
#ebx.staging.maxElementPerList=1000

# Defines a set of folder relative paths to ${ebx.repository.directory}/startup, separated by commas,
# used to configure the import of staging archives on startup.
# The folders must be strictly located in the startup folder with no deviations or shortcuts
# Each folder is a configuration of an import which must contain a file
# named "instructions.json" and the staging archives to import.
# Note that the instructions file can only mention archives strictly under the same folder
# and can not be on the root of (${ebx.repository.directory}/startup).
# In case of errors, an error.json report is added to the folder.
# In case of success, an output.json report is added to the folder
# and the folder would be ignored in the next startup.
```

```
# The content of the file is a json in the following format:
# {
#  "imports": [
#    {
#      "archiveFilePath": "myArchive1.zip"
#    },
#    {
#      "archiveFilePath": "myArchive2.zip",
#      "advanced_options": {
#        "forceUpdateOnIdenticalComponent": false,
#        "importDefinition": true
#      }
#    }
#  ]
#}
# [forceUpdateOnIdenticalComponent] boolean, when true, forces the update of a component even
# if it is the same in the repository.
# [importDefinition] boolean, when true, imports the domain definition in the staging domain
# configurations to be able to re-export the same domain.
#ebx.staging.importOnStartup=hr,catalog
```

See also *Staging*

7.18 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®.

See also *URL policy (deprecated)*

```
################################################
## 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/
## {http.useHttpsSettings}:
## If true, force the use of SSL security even if the incoming requests do not
## {authentication.redirectToHttps}
## If set to true, an HTTP request to the login form is automatically redirected to force it
## to HTTPS.
## Default is false.
## Usually, this property should be set to false if EBX® is behind a reverse proxy or a firewall
## that takes care of HTTPS encryption.
##
## 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.http.authentication.path=ebx-authentication/
#ebx.servlet.http.authentication.redirectToHttps=false
#ebx.servlet.http.useHttpsSettings=false

#ebx.servlet.https.host=
#ebx.servlet.https.port=
ebx.servlet.https.path=ebx/
#ebx.servlet.https.ui.path=ebx-ui/
#ebx.servlet.https.authentication.path=ebx-authentication/

################################################
## 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.http.useHttpsSettings=false

#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 incoming requests are always assumed to be HTTPS if `ebx.servlet.http.useHttpsSettings` and / or `ebx.externalResources.http.useHttpsSettings` properties are set to true. Their default values are `false`. These properties are useful if EBX® is behind a reverse proxy or a firewall that takes care of HTTPS encryption.

The main configuration file may be 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.http.authentication.path=authenticationServletAlias
#ebx.servlet.http.authentication.redirectToHttps=false
#ebx.servlet.http.useHttpsSettings=false

#ebx.servlet.https.host=
#ebx.servlet.https.port=
ebx.servlet.https.path=servletAlias
ebx.servlet.https.ui.path=uiServletAlias
ebx.servlet.https.authentication.path=authenticationServletAlias

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

ebx.externalResources.useLocalUrl=true

#ebx.externalResources.http.host=
#ebx.externalResources.http.port=
ebx.externalResources.http.path=resourcesAlias
#ebx.externalResources.http.useHttpsSettings=false

#ebx.externalResources.https.host=
#ebx.externalResources.https.port=
ebx.externalResources.https.path=resourcesAlias
```

Attention

When proxy mode is used, the URL to the `ebx-dataservices` module must be configured through the lineage administration panel. Note that the provided URL must end its path with `/ebx-dataservices`.

Reverse-proxy mode

If URLs generated by EBX®, for requests and external resources, must contain a different protocol than the one from the incoming request, a specific server name, a specific port number or a specific path prefix, properties may be configured as follows:

```
################################################
#ebx.servlet.useLocalUrl=false

ebx.servlet.http.host=reverseDomain
#ebx.servlet.http.port=
ebx.servlet.http.path=ebx/
```

```
#ebx.servlet.http.ui.path=ebx-ui/
#ebx.servlet.http.authentication.path=ebx-authentication/
#ebx.servlet.http.authentication.redirectToHttps=false
#ebx.servlet.http.useHttpsSettings=false

ebx.servlet.https.host=reverseDomain
#ebx.servlet.https.port=
ebx.servlet.https.path=ebx/
#ebx.servlet.https.ui.path=ebx-ui/
#ebx.servlet.https.authentication.path=ebx-authentication/

################################################
## Web parameters (for external resources)
## if nothing is set, values are taken from servlet.
################################################
#ebx.externalResources.useLocalUrl=false

#ebx.externalResources.http.host=
#ebx.externalResources.http.port=
#ebx.externalResources.http.path=
ebx.externalResources.http.useHttpsSettings=true

ebx.externalResources.https.host=reverseDomain
#ebx.externalResources.https.port=
ebx.externalResources.https.path=
```

Attention

When reverse-proxy mode is used, the URL to the `ebx-dataservices` module must be configured through the lineage administration panel. Note that the provided URL must end its path with `/ebx-dataservices`.

7.19 **Security configuration**

These parameters are used to configure the external pages that EBX® can redirect the user to, as well as some options regarding the authorization cookie.

```
# Custom login page URL.
# The login page is displayed when an unidentified user accesses EBX.
# If no url is specified, a built-in page will be displayed.
#ebx.security.loginPage.url=

# When using a custom login page, should a 'resume' parameter be appended to the URL?
# Default value is: true
#ebx.security.loginPage.appendResumeUrl=true

# Custom exit page URL.
# The exit page is displayed when a user exits EBX normally (when logging out for instance)
# and, if the 'exit error' page is not defined, when the user exits EBX because of an error.
# If no url is specified, a built-in page will be displayed.
#ebx.security.exitPage.url=

# When using a custom exit page, should a 'resume' parameter be appended to the URL?
# Default value is: true
#ebx.security.exitPage.appendResumeUrl=true

# Custom 'exit error' page URL.
# The 'exit error' page is displayed when a user exits EBX because of an error.
# If no url is specified, the exit page will be displayed.
#ebx.security.exitErrorPage.url=

# When using a custom 'exit error' page, should a 'resume' parameter be appended to the URL?
# Default value is: true
#ebx.security.exitErrorPage.appendResumeUrl=true

# Custom 'access denied' page URL.
# The 'access denied' page is displayed when an exception is thrown when authenticating a user.
# If no url is specified, a built-in page will be displayed.
#ebx.security.accessDeniedPage.url=

# List of authorized domains to redirect to.
# This is a list of comma-separated URLs which has as many items as 'ebx.security.exitRestrictions.httpsOnly'.
# For each item in this list, there should be a related item in 'ebx.security.exitRestrictions.httpsOnly'.
# Example: ebx.security.exitRestrictions.domains=my-domain.com,safe-domain.com,trusted-domain.com
#ebx.security.exitRestrictions.domains=
```

```
# For each domain declared in 'ebx.security.exitRestrictions.domains', indicates if only the https protocol is
 allowed.
# This is a list of comma-separated booleans which has as many items as 'ebx.security.exitRestrictions.domains'.
# For each item in this list, there should be a related item in 'ebx.security.exitRestrictions.domains'.
# Example: ebx.security.exitRestrictions.httpsOnly=true,false,false
#ebx.security.exitRestrictions.httpsOnly=

# Indicates if the authorization cookie should have the 'Secure' attribute.
# Accepted values are:
#   - true
#   - false
#   - auto: depends on the of value of 'ebx.https.support' and the protocol of the incoming request
# Default value is: auto
#ebx.security.authorizationCookie.attribute.secure=auto

# Indicates the value of the 'SameSite' attribute of the authorization cookie.
# Accepted values are:
#   - Strict
#   - Lax
#   - None
# Default value is: Strict
#ebx.security.authorizationCookie.attribute.sameSite=Strict
```

7.20 **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
```

7.21 **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 69]*

7.22 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 true.
################################################
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 `false`.

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

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

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
```

Activating delegation to launch a workflow by API

This parameter activates delegation to launch a workflow by API. The default value is `false` which means that permissions will be always checked unless this parameter is activated.

```
################################################
## Activate delegation to launch workflows by API
################################################
ebx.workflow.api.processLauncher.permissionCheckDelegated=true
```

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 cache activation

This parameter specifies whether or not to activate the cache for the internal request-to-query conversions that are required by the query engine when a `Request`[API] is executed. The activation of this cache will improve the performance on certain types of requests, especially if they are executed multiple times.

```
# Activates the request cache, which can improve the performance of certain requests. Possible values are
 DISABLED,
# WITH_CHECK (ensures that a cached request executes the same query plan as its non-cached version, recommended),
# EXCEPTION (throws an exception if the query plan for a request differs from its non-cached version), and
 ENABLED.
# The default value is WITH_CHECK.
#ebx.cache.request.mode=WITH_CHECK
```

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 repetition 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 two log entries 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.
##################################################
#ebx.logs.request.logAgainEvery=30000
```

Request optimization threshold for logs

This parameter specifies in milliseconds the threshold of the optimization phase 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`.

```
##################################################
## Specifies in milliseconds the threshold of optimization of Requests and Queries
## to be logged. Default value is 1000 ms.
##################################################
#ebx.logs.request.optimizationThreshold=1000
```

Request SQL representation for logs

This parameter specifies if the logs for a certain `Request`[API] shall include the corresponding SQL representation of the request. Notice that this conversion takes a toll in terms of performance, and not all requests can be converted into SQL. The default value is `false`.

```
################################################
## Specifies if the logs of requests shall include the SQL representation.
## Default is false.
################################################
#ebx.log4j.category.log.request.enableSQLConversion=false
```

Request optimization for sorting a table by foreign key labels

This parameter specifies whether the query optimizer rewrites certain combinations of sort and join operations to a specialized operator that carries out the join(s) in a sorted manner.

```
################################################
## Properties to activate/deactivate EnumerableSortedNestedLoopLeft/Right operators,
## which can improve the performance of requests sorted by foreign key labels.
## Default value is true.
################################################
#ebx.query.sortedNestedLoopLeft.enabled=true
#ebx.query.sortedNestedLoopRight.enabled=true
```

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 65]*

```
################################################
## 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.

When proxy and / or reverse-proxy mode are used, the `ebx.servlet.http[s].path` and `ebx.servlet.http[s].ui.path` properties must take into account this module public path prefix setting. Conversely, the `ebx.externalResources.http[s].path` property must end its path just before a potential prefix.

```
################################################
ebx.servlet.useLocalUrl=true

#ebx.servlet.http.host=
#ebx.servlet.http.port=
ebx.servlet.http.path=reverse-proxy/prefix/ebx/
ebx.servlet.http.ui.path=reverse-proxy/prefix/ebx-ui/
ebx.servlet.http.authentication.path=reverse-proxy/prefix/ebx-authentication/
#ebx.servlet.http.authentication.redirectToHttps=false
#ebx.servlet.http.useHttpsSettings=false

#ebx.servlet.https.host=
#ebx.servlet.https.port=
ebx.servlet.https.path=reverse-proxy/prefix/ebx/
ebx.servlet.https.ui.path=reverse-proxy/prefix/ebx-ui/
ebx.servlet.https.authentication.path=reverse-proxy/prefix/ebx-authentication/

################################################
## Web parameters (for external resources)
## if nothing is set, values are taken from servlet.
################################################
ebx.externalResources.useLocalUrl=true
```

```
#ebx.externalResources.http.host=
#ebx.externalResources.http.port=
ebx.externalResources.http.path=reverse-proxy/
#ebx.externalResources.http.useHttpsSettings=false

#ebx.externalResources.https.host=
#ebx.externalResources.https.port=
ebx.externalResources.https.path=reverse-proxy/

################################################
## 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=prefix/
```

See URLs computing [p 58] for more information.

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 can be of type [pattern] or "m:[pattern]:".
## the list can be void
################################################
```

```
ebx.resource.exclude=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 62]*

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
```

By default, the content of the validation of a dataset or a table is logged. Logging the content of the validation reports can be deactivated using the following property:

```
##################################################
## Specifies whether the content of validation reports is logged when validating
## a dataset or a table.
## Default is true.
##################################################
ebx.validation.report.logContent=true
```

See also *Tuning the EBX® repository [p 62]*

Validation mode of child datasets

This property influences how the algorithm for validating a child dataset behaves. It only affects the performance. The contents of the resulting validation reports are independent of the property.

```
# Changes the way how a child dataset is validated. This affects only the performance of the validation,
# but not the content of the resulting validation report. In the FULL mode, the report of the tables
# will be cleared, then the tables will be validated as if performing an initial validation.
# In the DIFF mode, the algorithm calculates the differences since the last validation
# and processes them to perform the necessary updates on the validation report.
# Possible values are: FULL, DIFF.
# The default is DIFF.
#ebx.validation.childDataset.mode=DIFF
```

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 47] 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 11].

> **Note**
>
> The EBX® main configuration file must also be properly configured. See TIBCO EBX® main configuration file [p 45].

This chapter contains the following topics:

8.1 License key

When launching EBX®, the license key page displays automatically if no valid license key is available, that is, if there is no license key entered in the EBX® main configuration file, or if the current license key has expired.

The license key can be retrieved from the TIBCO eDelivery site (an account is mandatory in order to access eDelivery, please contact https://support.tibco.com to request one).

8.2 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 45]*

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

Managing TIBCO EBX® add-ons

This chapter contains the following topics:

9.1 Overview

The following sections cover:

Refer to each add-on's documentation for instructions on configuring and using the add-on.

9.2 Add-on compatibility

Each Add-on Bundle is compatible with specific versions and fix releases of EBX®. However, beginning with version 6.1.0, the major and minor versions of add-ons and EBX® must be the same. For example, EBX® version 6.1.0 is only compatible with add-ons that are also version 6.1.0. If this is not the case, add-on registration is aborted. For other releases, the *TIBCO EBX® Add-ons Versioning and Packaging Guide* includes a version compatibility matrix. Go to the official documentation site and use the **Bundle version** menu to view the version of this guide that corresponds with your Add-on Bundle.

9.3 Deploying an add-on module

Copy the add-on common JAR file (named `lib/ebx-addons.jar`) in the EBX® class-path; it must be accessible from the ebx.jar class-loader.

Copy the add-on common WAR file (named `wars/ebx-addon-common.war`) in the application server webapps folder or EAR as EBX® built-in web applications.

If an add-on is used, copy the EBX® add-on WAR file (named `wars/ebx-addon-<name>.war`) in the same folder or EAR as EBX® built-in web applications.

See Web applications [p 14], Deployment details [p 16] and Installation notes [p 19] for more information.

> **Note**
>
> The add-on log level can be managed in the main configuration file [p 54].

9.4 Registering an add-on module

Registering an add-on makes its configuration available in the **Administration** panel. Add-on features are only available to end-users when the add-on is also activated [p 74].

EBX® add-ons are automatically registered and enabled in the repository. To configure whether an add-on is active:

1. Navigate to the **Administration** panel.
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 **add-on** record you want to configure, and make any desired changes.
4. Select the add-on you are registering.
5. Click **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.

9.5 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**.

9.6 Deleting an add-on module

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

1. Navigate to the **Administration** panel.
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 you want to delete.
4. In the **Actions** menu, select **Delete**.
5. Stop the application server.
6. Add the deleted add-on's module name (ebx-addon-<name>) to the `ebx.module.undeployedModules` property in your `ebx.properties` file. See declaring modules as undeployed [p 65] for more information.
7. Remove the add-on's WAR file (ebx-addon-<name>.war) from the application server's `webapps` folder, or its EAR file if used as an EBX® built-in web application.
8. Restart the application server.
9. Close and purge the Administration datasets and dataspaces related to the deleted add-on.

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

CHAPTER 10

User authentication

This chapter contains the following topics:

10.1 The /ebx-authentication servlet

The /ebx-authentication servlet is the central point handling the user authentication in EBX®. Its role is to:

- Create an authorization token [p 77] when the user logs in for the first time.
- Revoke the authorization token [p 77] when the user logs out.
- Redirect the authorized user to the appropriate page.

Furthermore, depending on the configuration, it can also:

- Handle the login process by displaying a login screen.
- Redirect the unauthorized user to a custom login screen [p 78].
- Redirect the user logging out to a custom exit page.

10.2 The authorization token

When the /ebx-authentication servlet successfully authenticates a user, it creates a token containing the authentication information and stores it in a session cookie.

This means that the authorization is shared between all the browser tabs:

- If the user is already authenticated on a browser tab and opens EBX® on another tab, it will not be asked to authenticate again.
- It is not possible to be logged with different users in different tabs of the same browser.

Cookie properties

The cookie containing the token has the following properties:

`HttpOnly`	Not customizable. The cookie can't be read by javascript code in the browser.
`Secure`	Customizable via the property `ebx.security.authorizationCookie.attribute.secure` in `ebx.properties`. Defines if the cookie can be sent over HTTPS only, or if it can also be sent over HTTP.
`SameSite`	Customizable via the property `ebx.security.authorizationCookie.attribute.sameSite` in `ebx.properties`. Defines if the cookie can be sent when browsing from an external site.

See also *Security configuration [p 61]*

10.3 Customizing the authentication process

By default, EBX® uses a built-in directory and built-in login page. It is strongly recommended to replace the built-in directory by a custom one.

Customizing the directory

While EBX® provides a built-in directory, it is strongly recommended to replace it:

1. Create a class overriding `Directory`[API]
2. Create a class overriding `DirectoryFactory`[API]
3. Use the property `ebx.directory.factory` in `ebx.properties` to declare the factory.

See also *Security configuration [p 61]*

Customizing the login page

While EBX® provides a built-in login page, it is possible to replace it. To do so, use the property `ebx.security.loginPage.url` in `ebx.properties` to declare the custom login page.

It is up to the custom login page to authenticate the user. Once the user is authenticated, he should be redirected to the `/ebx-authentication/login` page. Then, EBX® will call the `Directory.authenticateUserFromHttpRequest`[API] method before creating an authorization token. The following points are required for this to work:

- The request pointing to `/ebx-authentication/login` should contain enough information to authenticate the user, for instance in a cookie, a HTTP header or a query parameter.

- The directory must be overridden (see Customizing the directory [p 78]) and the implementation of the `Directory.authenticateUserFromHttpRequest`[API] method should read the information from the incoming request in order to return the appropriate `UserReference`[API].

> **Note**
>
> When EBX® redirects the user to the custom login page, it also adds a `resume` query parameter to the URL. This `resume` query parameter contains an URL pointing to `/ebx-authentication/login` with some additional parameters and should be used in priority to redirect the user to `/ebx-authentication/login` after the authentication succeeds.

See also *Security configuration [p 61]*

CHAPTER 11

Documentation and Support

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

This chapter contains the following topics:

11.1 How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product.

11.2 Product-Specific Documentation

The documentation for the TIBCO EBX® is available on the TIBCO EBX® Product Documentation page. This page contains the latest version of each document.

The documentation for the TIBCO EBX® Add-ons is available on the TIBCO EBX® Add-ons Product Documentation page. This page contains the latest version of each document.

To view the documents for Add-on Bundles that are compatible with other versions of TIBCO EBX®, use the **Bundle version** menu to select the desired release.

11.3 How to Contact TIBCO Support

Get an overview of TIBCO Support. You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support website.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to TIBCO Support website. If you do not have a user name, you can request one by clicking **Register** on the website.

11.4 How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to TIBCO Community.

CHAPTER 12

Legal and Third-Party

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. See the readme 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 Cloud Software Group, Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (https://www.tibco.com/patents) for details.