**Stack 1 Red Hat – JBoss runbook**

The **Stack 1 Red Hat – JBoss** runbook shows you how to install and configure TAFJ-Transact on JBoss EAP 7.3 with any of the following databases: Microsoft SQL Server 2019, NuoDB 4.x, Oracle 19c or PostgreSQL 11/12.When you complete the deployment, you will be able to perform business operations in Transact using UXP Browser and BrowserWeb.

**Note:**

This runbook does not tell you how to install third-party software. For more information, see the relevant vendor's documentation.

**Stack tables and other Stack runbooks**

To view the software that is supported as part of stack 1, together with our other stack runbooks, see the R21 stacks table in either the Temenos [customer](https://tcsp.temenos.com/SV/SitePages/R21Stack.aspx) or [partner](https://tpsp.temenos.com/StackVerification/SitePages/R21.aspx) support portal.

**Scope**

This runbook covers:

* Installing Database
* Installing TAFJ
* Installing Transact
* Configuring JBoss 7.3 EAP
* Deploying Transact and TAFJ artefacts in JBoss

**Audience**

This runbook has been written for consultants who are responsible for deploying TAFJ-Transact on JBoss application server.

**Skills and knowledge**

To get the most from this runbook, you need to be familiar with the following technologies and products:

* TAFJ
* Transact
* JBoss
* The chosen database management system: Microsoft SQL Server, NuoDB, Oracle Database, or PostgreSQL

**Introduction**

This runbook helps you set up a development or test environment for Transact and TAFJ on Red Hat Linux 8.2. It shows you how to install and configure TAFJ, Transact and JBoss and how to access Transact through both BrowserWeb and UXP Browser.

**The architecture**

In this exercise, a two-tier architecture has been created which consists of an application server and a database server. The application tier contains JBoss EAP (version 7.3) application server with TAFJ and Transact. The database tier contains one of the following DBMSes: MS SQL Server 2019, NuoDB 4.1.0.1, Oracle 19c, or PostgreSQL 11/12. For better understanding, the architecture is given below.

**UXP Browser Generate On Demand (GOD) mode**

In releases prior to R20, there was a large directory in UXP Browser on the web server file system (BRP/generated) which contained over 250,000 IRIS and Edge files. To manage such a large number of files securely and keep the system scalable was difficult.

In R20 and later releases, model-derived artefacts are no longer pre-generated. Instead, they are generated on-demand and saved in the Resource Server database. As a result, the start-up time of the system has been significantly reduced.

The first time you access a screen, there is a one-off delay as the system generates the screen. Once this completes, the screen is stored for all subsequent requests. The generated screens are stored in the database permanently until the underlying definition changes in Transact.

**Components deployed in the GOD mode of UXP Browser**

You must configure and deploy the following components to deploy and run UXP Browser in the GOD mode.

**Authenticator (Authenticator.war)**

This component validates Transact user credentials. It authenticates user names and passwords against Transact using the tComponent framework and ensures there is secure communication between Browser and IRIS.

**UXP Browser (Browser.war)**

This component renders the new UXP Browser user interface. It communicates with the Resource Server to render front end screens and with the IRIS layer to use Transact data from the database. Its main benefits are: better scalability, the SaaS capability and support for multi tenancy.

**Resource server (ResourceServer.war)**

This component uses a database schema (within each per-tenant Transact database) providing an alternative to the file system for the storage and retrieval of tenant-specific variants of resources that are external to Browser.war, for example artefacts generated from models (projects, menus, templates etc.) and static resources (CSS stylesheets, HTML files, etc.).

**Interaction Framework Resource Provider service (irf-rp-services.war)**

This service processes requests and responses through the Resource Provider (RP) component apart from through traditional Open Financial Service (OFS).

Using the Interaction Framework Resource Provider service, you can communicate with Transact to perform various Transact operations such as create, view, delete, update and so on, for any versions and fetch any enquiry results without any hassle of metadata generation.

**Interaction Framework Catalog services (irf-t24catalog-services.war)**

This service was designed in a generic way to retrieve metadata information for any Transact artefact in an XML formatted response. The Interaction Framework Catalog service is used by UXP Browser deployed in the Generate On Demand mode as well as Design Studio on DSF Packager.

**Enterprise Java Beans (EJB files)**

The EJBs are packaged as JAR files and are deployed on the Transact application server to support remote EJB connectivity. They enable APIs and Transact running on separate or remote application servers to communicate with each other. The following EJBs are required:

* Authentication service EJB (t24-EB\_AuthenticationService-ejb.jar)
* Catalog service EJB (t24-EB\_CatalogService-ejb.jar)
* Resource Provider service EJB (t24-EB\_ResourceProviderService-ejb)

**Prerequisites**

Ensure that your environment meets the requirements listed in this section before you install TAFJ and Transact. This section lists the software prerequisites and the assumptions made regarding the installation requirements.

**Software prerequisites**

**Third-party software**

| Software | Version |
| --- | --- |
| OpenJDK | 1.8 |
| JBoss EAP | 7.3 |
| Microsoft SQL Server | 2019 |
| NuoDB | 4.1.0.1 |
| Oracle 19c | 12.2.0.3 |
| PostgreSQL | 12.3 |
| Axis2 | 1.6.2 |

**Temenos artefacts**

| Artefacts | File name | Description |
| --- | --- | --- |
| MS SQL bak file | MB.202008.MSSQL\_2014.WIN.TAFJ202008.30-SEP-2021.bak.tar.gz | Contains the database. |
| NuoDB Backup file | MB.202008.NuoDB\_4.0.1.1.LINUX.TAFJ202008.30-SEP-2021.backup.tar.gz | Contains the database. |
| Oracle dump file | MB.202008.ORACLE12.2.UNIX.TAFJ202008.30-SEP-2021.dmp.tar.gz | Contains the database. |
| PostgreSQL dump file | MB.202008.PostgresSQL\_11.8.TAFJ202008.30-SEP-2021.sql.tar.gz | Contains the database. |
| Transact | MB.202008.TAFJ202008.bnk.tar.gz | Contains the bnk directory that holds the Transact libraries. |
| TAFJ | TAFJ.DEV.202008.1.tar.gz | Contains the TAFJ runtime .jar file, TAFJ patch script and TAFJ setup script. |
| Transact Browser | BrowserWeb-202008.00.tar | Old browser components. |
| UXP Browser | UXP-Browser.202008.zip  This installation package contains the following required components:   * Authenticator-202008.0.1.war * Browser-202008.0.1.war * irf-rp-services-202008.0.1.war * ResourceServerWar-202008.0.1.war | New browser components. |
| Additional EJB JAR files | t24-EB\_AuthenticationService-ejb.jar  t24-EB\_CatalogService-ejb.jar  t24-EB\_ResourceProviderService-ejb.jar  These files are available in the subfolders under $TEMENOS\_HOME/T24/bnk/Extensions. | EJB JAR files required for deploying UXP Browser in the Generate On Demand (GOD) mode |
| IRIS Catalog Services | IRISR18\_CatalogServices\_TAFJ\_202008.0.1.zip | Contains catalog services. |
| Temenos Encryption Utility | EncryptPassword.jar | The utility for encrypting passwords. |

**Hardware prerequisites**

Ensure that you have at least 16 GB RAM available on the application server. UXP Browser requires at least 8 GB RAM.

**Before you start**

Before you start installing and configuring the stack, verify that your technical infrastructure has been set up properly.

**Completing preinstallation tasks**

**Procedure**

1. Load Transact data from a file into the RDBMS of your choice. Use one of the following files:
   * **Microsoft SQL Server**: MB.202008.MSSQL\_2014.WIN.TAFJ202008.30-SEP-2021.bak.tar.gz
   * **NuoDB**: MB.202008.NuoDB\_4.0.1.1.LINUX.TAFJ202008.30-SEP-2021.backup.tar.gz
   * **Oracle Database**: MB.202008.ORACLE12.2.UNIX.TAFJ202008.30-SEP-2021.dmp.tar.gz
   * **PostgreSQL**: MB.202008.PostgresSQL\_11.8.TAFJ202008.30-SEP-2021.sql.tar.gz
2. Load TAFJ-related stored procedures and Java functions into the database. For more information, see the database-specific installation document that is stored in $TAFJ\_HOME/doc.
3. Install the required third-party software. For more information about the installation paths, see [Prerequisite directory structure](http://172.16.204.9/R21UG/Solutions/Runbooks/Stack-01-Runbook-R21/t_before-you-start.htm?tocpath=_____4#Prerequi).
4. Copy all Temenos artefacts into the chosen installation directory. For more information about the installation paths, see [Prerequisite directory structure](http://172.16.204.9/R21UG/Solutions/Runbooks/Stack-01-Runbook-R21/t_before-you-start.htm?tocpath=_____4#Prerequi).
5. Install Korn shell – at the command line, issue the following command:

yum install ksh

1. If you are going to use a non-standard port on the application server, ensure that you open it in the Linux firewall:
   * Add the port by running the following command:

firewall-cmd --zone=public --add-port=9089/tcp –-permanent

* + Reload the new configuration:

firewall-cmd --reload

* + Check that the new rule is enabled:

firewall-cmd --list-all

**Checking the system**

**Procedure**

1. Check the version of the operating system: verify that your system is running on Red Hat Enterprise Linux version 8.x. At the Linux command-line, execute the following command:

cat /etc/redhat-release



1. Check how much disk space is available. At the Linux command-line, execute the following command:

df -h

The minimum requirement is 20 GB of disk space.

1. Create the required directory structure.
   1. At the Linux command-line, navigate to your installation directory and create the Temenos directory.

mkdir Temenos

* 1. Under the Temenos directory, create the following directories.

mkdir -p Temenos/TAFJ  
mkdir -p Temenos/T24  
mkdir -p Temenos/UXPBrowser  
mkdir -p Temenos/3rdParty/Java  
mkdir -p Temenos/3rdParty/AS  
mkdir -p Temenos/inst  
mkdir -p Temenos/inst/tafjinst  
mkdir -p Temenos/inst/t24inst

**Prerequisite directory structure**

| **Directory** | **Subdirectory** | **Description** |
| --- | --- | --- |
| 3rdParty |  | The 3rdParty directory contains all installed third-party software. |
| Java | This directory contains the JDK binary files. |
| AS | This is the JBoss application server installation directory. |
| inst |  | The inst directory contains all installers to set up TAFJ-Transact. |
| t24inst | t24inst is a subdirectory of inst, which contains Transact installation files. |
| tafjinst | tafjinst is a subdirectory of inst, which contains TAFJ installation files. |
| T24 |  | The T24 directory contains all Transact-related libraries. |
| TAFJ |  | The TAFJ directory contains TAFJ runtime libraries. |
| UXPBrowser |  | The UXPBrowser directory contains UXP browser artefacts (New Browser). |

**Setting the environment variables**

Before starting the Transact or TAFJ installation, you need to set up the system environment variables.

**Procedure**

1. In the Linux shell, navigate to the user directory.

cd /home/*user\_home\_dir*

1. Open the .bash\_profile file in a text editor.

vi .bash\_profile

1. Enter the system environment variables as shown below. The bnk directory does not yet exist in your environment now but will be required later.
2. #TEMENOS\_HOME
3. export TEMENOS\_HOME=/srv/Temenos
4. echo $TEMENOS\_HOME
5. #TAFJ\_HOME
6. export TAFJ\_HOME=$TEMENOS\_HOME/TAFJ
7. echo $TAFJ\_HOME
8. #T24\_HOME
9. export T24\_HOME=$TEMENOS\_HOME/T24/bnk/UD
10. echo $T24\_HOME
11. #UXPBROWSER
12. export UXPBROWSER=$TEMENOS\_HOME/UXPBrowser
13. echo $UXPBROWSER
14. #BRP\_HOME
15. export BRP\_HOME=/srv/Temenos
16. echo $BRP\_HOME
17. #JAVA\_HOME
18. export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.265.b01-0.el8\_2.x86\_64
19. #JBOSS\_HOME
20. export JBOSS\_HOME=/srv/Temenos/3rdParty/AS/jboss-eap-7.3
21. echo $JBOSS\_HOME
22. #PATH
23. export PATH=$PATH:$JAVA\_HOME/bin:$TAFJ\_HOME/bin:$JBOSS\_HOME/bin
24. Execute .bash\_profile to export the variables.

**Verifying the system for third-party software**

**Procedure**

1. Verify that JDK 1.8 is installed by executing java –version in the Linux shell.

A black screen with white text

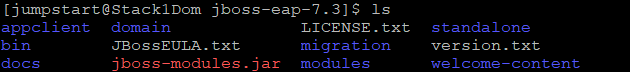
Description automatically generated

1. Add the JAVA\_HOME environment variable and $JAVA\_HOME/bin to the $PATH variable.
2. Check that you can connect to your database. Use the Oracle SQL Developer tool to connect to an Oracle database or SQL Server Management Studio to connect to an SQL Server database.

A screenshot of a computer

Description automatically generated

1. Verify that JBoss EAP 7.3 is installed and JBOSS\_HOME has been set as an environment variable. You should see the following folders in $JBOSS\_HOME.



 Dark Mode

**Installing Database**

Database Installation guides documents are provided below:



Note:- For R22 release onwards, we are suppose to run the DBscritps as per the below attached document.



**Installing TAFJ runtime**

This section shows how to set up the TAFJ runtime.

**Preparing the Transact and TAFJ installation files**

**Procedure**

1. Obtain the MB.202008.TAFJ202008.bnk.tar.gz and TAFJ.DEV.202008.1.tar.gz files from your account manager.
2. Upload the Transact artefacts (MB.202008.TAFJ202008.bnk.tar.gz) from the temporary folder on your machine to the $TEMENOS\_HOME/inst/t24inst directory on the application server.
3. Copy the TAFJ artefacts (TAFJ.DEV.202008.1.tar.gz) from the temporary folder on your machine to the $TEMENOS\_HOME/inst/tafjinst directory on the application server.

**Extracting Transact**

**Procedure**

1. In the Linux shell, navigate to $TEMENOS\_HOME/inst/t24inst.
2. Extract the gz file. It will be extracted as a bnk folder.

tar –xvf MB.202008.TAFJ202008.bnk.tar.gz

1. Move the bnk directory to the T24 directory.

mv bnk $TEMENOS\_HOME/T24

**Installing TAFJ**

**Before you begin**

Ensure that you have set JAVA\_HOME to the correct path.

**Procedure**

1. In the Linux shell, navigate to $TEMENOS\_HOME/inst/tafjinst.
2. Extract the .gz file.

tar –xvf TAFJ.DEV.202008.1.tar.gz

A black screen with white text

Description automatically generated

1. Change the permissions on Setup\_TAFJ.DEV.202008.1.sh and TAFJ.DEV.202008.1.jar to execute.

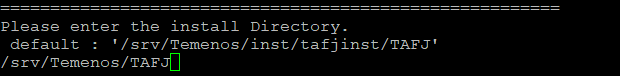
chmod 755 Setup\_TAFJ.DEV.202008.1.sh TAFJ.DEV.202008.1.jar

1. Execute Setup\_TAFJ.DEV.202008.1.sh and press Enter..

A screenshot of a computer program

Description automatically generated

1. When prompted to enter the installation directory, specify the path to $TAFJ\_HOME and press Enter.



1. Optional: Specify the Eclipse home directory and press Enter.

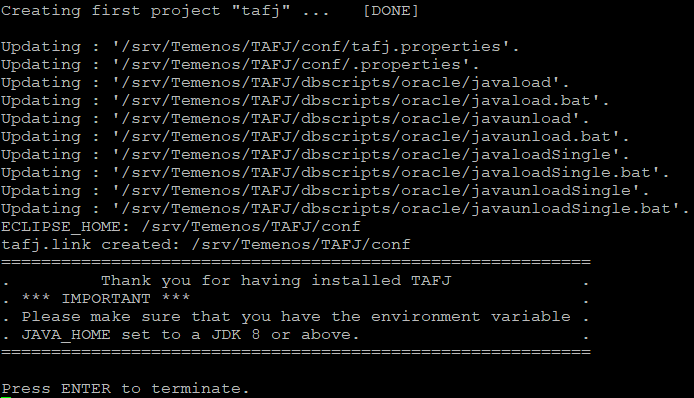
A screen shot of a computer screen

Description automatically generated

1. Enter y to confirm the path to the conf directory and press Enter.



1. Press Enter to finish the installation of TAFJ runtime.



The following directories are available in $TAFJ\_HOME.

A screen shot of a computer program

Description automatically generated

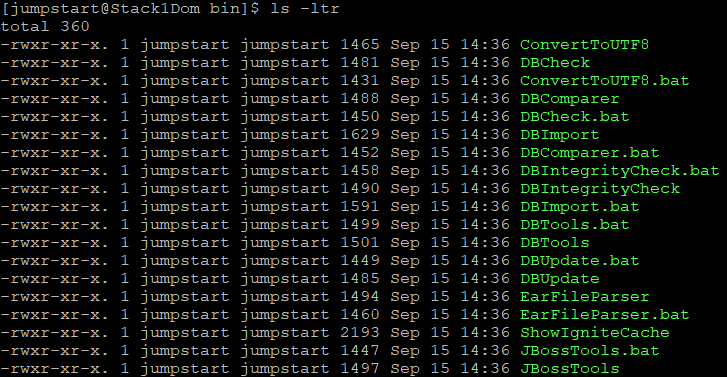
1. Change the permissions on the $TAFJ\_HOME/bin directory to execute.

chmod 755 $TAFJ\_HOME/bin/\*

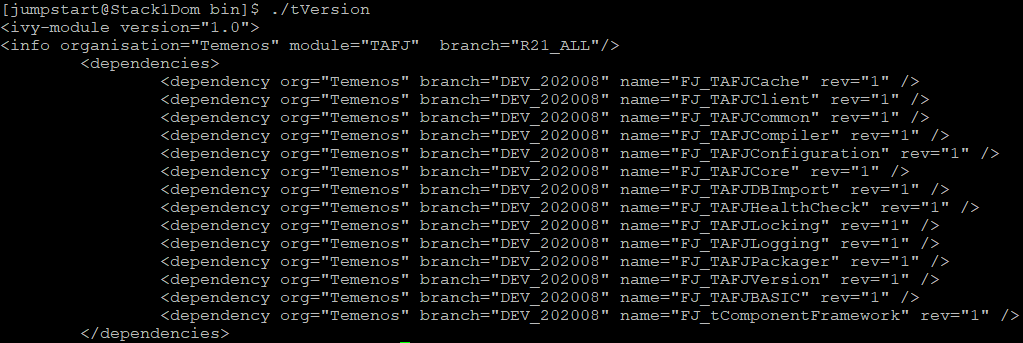
1. Verify that the chmod command completed successfully.

cd bin

ls -ltr



1. Execute tVersion to verify the TAFJ version.



**Deploying the database driver**

TAFJ uses JDBC drivers to establish a connection with the Transact database. Database drivers are available in the $TAFJ\_HOME/dbdrivers directory. For your deployment you must use the latest version of the respective database driver downloaded from the vendor’s web site.

**Deploying the driver for MS SQL Server**

**Procedure**

1. Download the JDBC SQL driver from Microsoft web site.
2. Copy the sqljdbc42.jar file to the $TAFJ\_HOME/ext directory.

**Deploying the driver for NuoDB**

Copy the nuodb-jdbc-21.0.0.jar file to the $TAFJ\_HOME/ext directory.

cp $TAFJ\_HOME/dbdrivers/nuodb-jdbc-21.0.0/nuodb-jdbc-21.0.0.jar $TAFJ\_HOME/ext.

**Deploying the drivers for Oracle Database**

**Procedure**

1. Download the Oracle 19c drivers from the Oracle web site. The following are the required drivers:
   * ojdbc8.jar
   * oraclepki.jar
   * orai18n.jar
   * orai18n-collation.jar
   * orai18n-mapping.jar
   * ucp.jar
   * xdb6.jar
   * xmlparserv2\_sans\_jaxp\_services.jar
2. Copy the JAR files to $TAFJ\_HOME/ext directory.

cp -r $TAFJ\_HOME/dbdrivers/oracle-12c $TAFJ\_HOME/ext

**Deploying the driver for PostgreSQL**

**Procedure**

1. Download the JDBC SQL driver from the [www.postgresql.org](https://www.postgresql.org/) web site.
2. Copy the postgresql-42.2.6.jar file to the $TAFJ\_HOME/ext directory.

**Configuring TAFJ**

This section shows how to configure TAFJ runtime to set up a standalone (outside an application server) Transact instance.

TAFJ does not use environment variables. Instead, it uses a properties file (\*.properties) for its internal configuration. Properties files are stored under the $TAFJ\_HOME/conf directory. The default property file is tafj.properties.

**Configuring tafj.properties**

**Procedure**

1. In the Linux shell, navigate to $TAFJ\_HOME/conf.
2. Open the tafj.properties file in a text editor.
3. Verify that tafj.home points to the correct location.

A white rectangular frame with yellow stars

Description automatically generated

1. Set temn.tafj.directory.precompile to the path of Transact libraries.

A close up of text

Description automatically generated

1. Set the database connection parameters.
   1. Under the **Database setup** header section, provide the relevant temn.tafj.jdbc.url.
   2. Provide the relevant driver information in temn.tafj.jdbc.driver.
   3. Provide database user name and password.

**MS SQL Server**

In temn.tafj.jdbc.url change **integratedSecurity** to false.

A screenshot of a computer

Description automatically generated

**NuoDB**

A screenshot of a computer code

Description automatically generated

**Oracle Database**

A computer screen shot of a computer code

Description automatically generated

**PostgreSQL**

A screenshot of a computer

Description automatically generated

1. Set the UD directory path.

A white background with black and red text

Description automatically generated

1. Set the value of temn.tafj.jdbc.write.use.merge and temn.tafj.jdbc.write.use.merge.no.xml to false to improve the database performance.

A screenshot of a computer program

Description automatically generated

1. Set the value of temn.tafj.locking.mode to DATABASE.

A screenshot of a computer

Description automatically generated

**Note:**

* 1. DATABASE locking mode is recommended for releases 201809 and higher. TAFJ locks records using native database locks when this locking mode is used.
  2. For information on how to create and populate the TAFJ\_HASHLOCKS table which is required for DATABASE locking mode see the *TAFJ Lock Manager* guide (TAFJ-Lock Manager.pdf) that is in $TAFJ\_HOME/doc.

1. To enable SAMPLE to work with PostgreSQL, set the **temn.tafj.jdbc.use.rownum.at.end** property to true.



**Verifying TAFJ installation**

**Procedure**

1. In the Linux shell, navigate to $TAFJ\_HOME/bin.
2. Execute the tDiag command to verify that your TAFJ installation is successful and that the project properties file is correctly set. This command shows you:
   * What location $TAFJ\_HOME is set to.
   * JDK version.
   * TAFJ version.
   * The default project, and the details of all the projects that are set in $TAFJ\_HOME/conf.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

This completes the TAFJ runtime setup.

**Sanity check**

Verify that your Transact standalone installation is successful before you proceed to deploy the application server. This section shows how to perform two sanity checks.

**Accessing Classic**

You can use Transact Classic, which is one of the Transact user interfaces, to access Transact.

**Procedure**

1. In the Linux shell, navigate to $TAFJ\_HOME/bin.
2. Run the command tRun EX.

A screen shot of a computer screen

Description automatically generated

1. Log in using a valid Transact user name and password.

A screen shot of a computer

Description automatically generated

1. Launch any Transact application, for example SPF S SYSTEM.

A screenshot of a computer

Description automatically generated

**Launching DBTools**

DBTools is a TAFJ tool that provides a console to execute database commands. Only authenticated users can access the console.

**Procedure**

1. In the Linux shell, navigate to $TAFJ\_HOME/bin.
2. Create a TAFJ user by passing user name and password parameters to the tUserMgnt.sh utility.

./tUserMgnt --Add -u tafjuser -p Temenos@1234

The USER CREATION SUCCESSFUL message is displayed.

1. Launch DBTools passing the TAFJ user credentials as parameters.

./DBTools -u tafjuser -p Temenos@1234

The DBTools console is launched.

1. View the database name, database user name and IP address, which are displayed in the lower left part of the screen.

A screenshot of a computer program

Description automatically generated

 Dark Mode

**Configuring JBoss modules**

JBoss EAP 7.3 uses a modular class loading system for controlling the class paths of deployed applications. The modular class loader separates all Java classes into logical groups called modules. Each module can define dependencies on other modules.

To install a module on JBoss EAP 7.3, create a path under the $JBOSS\_HOME/modules directory. Under this path, install the JAR libraries that are part of the module and the module.xml file that describes the module and the dependencies on other modules. You need to create three modules on your application server.

* Database driver module
* TAFJ module
* Transact module

**Configuring the database driver module**

For Transact to connect to a data source, a module should contain your data source vendor’s JDBC drivers for JBoss to use.

**Configuring the module for Microsoft SQL Server**

**Procedure**

1. In the Linux shell, navigate to $JBOSS\_HOME/modules.
2. Create the following directory structure under the modules directory.

mkdir -p com/microsoft/sqlserver/main

1. Copy module.xml from $TAFJ\_HOME to the main directory.

cp $TAFJ\_HOME/appserver/jboss/jboss7eap/modules/com/microsoft/sqlserver/main/module.xml $JBOSS\_HOME/modules/com/microsoft/sqlserver/main

1. Copy the corresponding database driver to the main directory.
2. Verify that the copy was successful.



1. Open the copied module.xml file to check the name of driver JAR file. The module name should be same as the folder path for that module.

A screenshot of a computer code

Description automatically generated

**Configuring the module for NuoDB**

**Procedure**

1. In the Linux shell, navigate to $JBOSS\_HOME/modules.
2. Create the following directory structure under the modules directory.

mkdir -p com/nuodb/jdbc/main

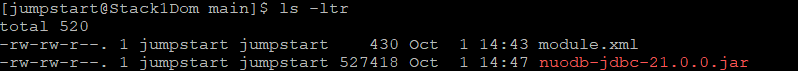
1. Copy module.xml from $TAFJ\_HOME to the main directory.

cp $TAFJ\_HOME/appserver/jboss/jboss7eap/modules/com/nuodb/jdbc/main/module.xml $JBOSS\_HOME/modules/com/nuodb/jdbc/main

1. Copy the corresponding database driver to the main directory.

cp $TAFJ\_HOME/ext/nuodb-jdbc-21.0.0.jar $JBOSS\_HOME/modules/com/nuodb/jdbc/main

1. Verify that the copy was successful.



1. Open the copied module.xml file to check the name of driver JAR file. The module name should be same as the folder path for that module.

A screenshot of a computer code

Description automatically generated

**Configuring the module for Oracle Database**

**Procedure**

1. In the Linux shell, navigate to $JBOSS\_HOME/modules.
2. Create the following directory structure under the modules directory.

mkdir -p com/oracle/ora18c/main

1. Copy module.xml from $TAFJ\_HOME to the main directory.

cp $TAFJ\_HOME/appserver/jboss/jboss7eap/modules/com/oracle/ora12c/main/module.xml $JBOSS\_HOME/modules/com/oracle/ora18c/main

1. Copy the corresponding Oracle drivers from $TAFJ\_HOME/ext to the $JBOSS\_HOME/modules/com/oracle/ora18c/main directory.

cp $TAFJ\_HOME/dbdrivers/oracle-12c/\* $JBOSS\_HOME/modules/com/oracle/ora18c/main

1. Check whether the copy was successful.

A screen shot of a computer screen

Description automatically generated

1. Open the copied module.xml file and change the module name to ora18c.

A screenshot of a computer code

Description automatically generated

**Configuring the module for PostgreSQL**

**Procedure**

1. In the Linux shell, navigate to $JBOSS\_HOME/modules.
2. Create the following directory structure under the modules directory.

mkdir -p com/postgres/jdbc/main

1. Copy module.xml from $TAFJ\_HOME to the main directory.

cp $TAFJ\_HOME/appserver/jboss/jboss7eap/modules/com/postgres/jdbc/main/module.xml $JBOSS\_HOME/modules/com/postgres/jdbc/main

1. Copy the corresponding Oracle drivers from $TAFJ\_HOME/ext to the $JBOSS\_HOME/modules/com/postgres/jdbc/main directory.
2. Check whether the copy was successful.



1. Open the copied module.xml file and ensure the module name is set to com.postgres.jdbc.

A screenshot of a computer code

Description automatically generated

**Configuring the TAFJ module**

This module contains the TAFJ libraries that are in the lib and ext directories. They have a dependency on the database driver module.

**Procedure**

1. In the Linux shell, navigate to $JBOSS\_HOME/modules/com.
2. Create the following directory structure under the com folder.

mkdir -p temenos/tafj/main

1. Copy the module.xml file from a directory under $TAFJ\_HOME to the following directory structure under $JBOSS\_HOME/modules.

cp $TAFJ\_HOME/appserver/jboss/jboss7eap/modules/com/temenos/tafj/main/module.xml $JBOSS\_HOME/modules/com/temenos/tafj/main

1. Change to $JBOSS\_HOME/modules/com/temenos/tafj/main.
2. Create symbolic links to the $TAFJ\_HOME/lib, $TAFJ\_HOME/ext and $TAFJ\_HOME/RulesEngine directories.

ln -s $TAFJ\_HOME/lib lib  
ln -s $TAFJ\_HOME/ext ext  
ln -s $TAFJ\_HOME/RulesEngine RulesEngine

1. Open the copied module.xml file in a text editor, uncomment the database driver module dependency under the <dependencies> section according to the database used.
   * If you are using Microsoft SQL Server, uncomment the following dependency:

A screen shot of a computer code

Description automatically generated

* + If you are using NuoDB, uncomment the following dependency:

A screen shot of a computer code

Description automatically generated

* + If you are using Oracle Database, uncomment the following dependency:

A screen shot of a computer code

Description automatically generated

* + If you are using PostgreSQL, uncomment the following dependency:

A screen shot of a computer code

Description automatically generated

1. In the module.xml file, add the following line as the last element under <resources>.

<resource-root path="./RulesEngine/RulesEngine.jar" />

A close-up of a computer code

Description automatically generated

 Dark Mode

**Configuring the Transact module**

This module contains the Transact JAR files and has a dependency on the TAFJ module that you created earlier.

**Procedure**

1. In the Linux shell, navigate to $JBOSS\_HOME/modules/com/temenos.
2. Create the following directory structure under the temenos directory.

mkdir -p t24/main

1. Change to $JBOSS\_HOME/modules/com/temenos/t24/main.
2. Create a symbolic link to $TEMENOS\_HOME/T24/bnk/t24lib.

ln -s $TEMENOS\_HOME/T24/bnk/t24lib lib

**Generating the Transact module.xml file**

You can use the JBossTools utility to generate the module.xml file. This utility is provided by Temenos and it is available in the $TAFJ\_HOME/bin directory.

**Procedure**

1. In the Linux shell, navigate to $TAFJ\_HOME/bin.
2. Create the module.xml file by running the following command.

JBossTools com.temenos.t24 $TEMENOS\_HOME/T24/bnk/t24lib $JBOSS\_HOME/modules/com/temenos/t24/main lib –tafjdep

The following screen capture shows the successful completion of executing the above command.

A screen shot of a computer program

Description automatically generated

1. Open module.xml and add the TAFJ module as a dependency in the <dependencies> section.

<module name="com.temenos.tafj" />

A screenshot of a computer code

Description automatically generated

**JBossTools command for generating module.xml**

**Command syntax**

JBossTools *module\_name* *path\_to\_jars* *dest* *root\_prefix* -tafjdep

| Options | Description |
| --- | --- |
| *module\_name* | Name of the JBoss module. It must be the same as the directory path created for the module. |
| *path\_to\_jars* | The list of directories that you want to parse. |
| *dest* | The path to the directory where the module.xml is to be generated. |
| *root\_prefix* | Name of the link if module.xml is at the same level as the link. |
| -tafjdep | Option to add TAFJ module dependencies in the dependencies section. |

# Configuring standalone profile on the application server

JBoss Enterprise Application Platform runs in one of two operating modes: as a standalone server or in a managed domain. This runbook covers standalone installation.

**Starting JBoss with J2EE 8 full profile**

**Procedure**

1. In the Linux shell, navigate to $JBOSS\_HOME/standalone/configuration.

There are several standalone profiles in JBoss. standalone-full.xml is a sample configuration file for a standalone server that includes support for every possible subsystem except for those required for high availability.

1. To start the application server using a full J2EE profile, copy and rename the standalone-full.xml file that is in the configuration directory to *name*.xml (For example, Transact.xml).

cp standalone-full.xml Transact.xml

1. Change to $JBOSS\_HOME/bin.
2. Open the standalone.conf file in a text editor and, on a new line, enter the following new JAVA\_OPTS definition.

JAVA\_OPTS="$JAVA\_OPTS -Dfile.encoding=UTF-8 -Dtafj.home=$TAFJ\_HOME"

This definition adds mandatory system properties tafj.home and file.encoding. TAFJ uses UTF-8 encoding.

Example:

A black screen with white text

Description automatically generated

1. Refine your maximum heap size (-mx) based on your expected load on the application server.

Example:

A screen shot of a computer program

Description automatically generated

**Note:**

When setting up the max heap size (-mx), set the initial heap size (-ms) to the same value as the max heap size. This prevents the JVM from consuming cycles to expand the heap since Transact starts with a large memory footprint.

1. Start JBoss using the following command. To use the configuration in Transact.xml, you must amend the JBoss startup command and explicitly refer to this configuration file using the option --server-config.

standalone.sh -b 0.0.0.0 -bmanagement 0.0.0.0 --server-config=Transact.xml

**Configuring the standalone full profile**

To deploy Transact on TAFJ in EAP, you must configure various subsystems in the Transact.xml file. TAFJ uses the management command line interface tool to automate the process of configuring the J2EE profiles.

You can launch JBoss CLI from the $JBOSS\_HOME/bin directory by running the jboss-cli.sh script. It takes the following two arguments: --file and --properties

The --file argument enables CLI commands to be provided from a text file. The $TAFJ\_HOME/appserver/jboss/jboss7eap/jboss-cli directory has a CLI script file T24Setup.cli. This script has commands to update the profiles specific for Transact deployment.

**Procedure**

1. In the Linux shell, navigate to $TAFJ\_HOME/appserver/jboss/jboss7eap/jboss-cli and open T24Setup.cli.
2. In the Drivers section, uncomment the driver parameters according to the database used.

**Microsoft SQL Server**

A screenshot of a computer code

Description automatically generated

**NuoDB**

A screenshot of a computer code

Description automatically generated

**Oracle Database**

Change the driver name to ora18c.

A close-up of a computer code

Description automatically generated

**PostgreSQL**

A close-up of a code

Description automatically generated

**Note:**

The --properties argument allows to pass user-specific properties to the script file from a property file. You will find tafj.properties in the same location.

1. Open tafj.properties.
2. Provide the TAFJ\_HOME, DB\_URL, DB\_DRIVER, DB\_USER and DB\_PWD parameters.

**Microsoft SQL Server**

A computer code on a white background

Description automatically generated

**NuoDB**

A screen shot of a computer

Description automatically generated

**Oracle Database**

A screenshot of a computer code

Description automatically generated

**PostgreSQL**

A screenshot of a computer code

Description automatically generated

1. To resolve the user-specific system properties in tafj.properties that will be passed as command argument to JBoss CLI command, set the value of **resolve-parameter-values** to true in $JBOSS\_HOME/bin/jboss-cli.xml. Check whether **validate-operation-requests** is set to true.

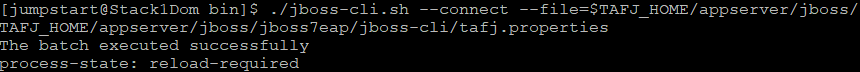
A screenshot of a computer code

Description automatically generated

1. In the Linux shell, navigate to $JBOSS\_HOME/bin and run the jboss-cli command.

jboss-cli.sh --connect --file=$TAFJ\_HOME/appserver/jboss/jboss7eap/jboss-cli/T24Setup.cli --properties=$TAFJ\_HOME/appserver/jboss/jboss7eap/jboss-cli/tafj.properties

On successful execution, the command updates the required subsystems in Transact.xml which start up JBoss. The following screen capture shows the successful execution of the jboss-cli command.



The scripts will add the following to the standalone configuration file. (Transact.xml)

* + System properties
  + Drivers
  + Data sources
  + Global modules (TAFJ and Transact)
  + JMS Pools
  + JMS Queues

1. Stop the JBoss instance as the configuration of JBoss with Transact and TAFJ is now complete. The next step is to deploy the artefacts.

**Extracting UXP Browser artefacts**

This section contains the steps that need to be done before deploying the Transact and TAFJ artefacts and are only required if you are deploying UXP Browser.

**Before you begin**

You need to obtain the UXP-Browser.202008.zip and IIRISR18\_CatalogServices\_TAFJ\_202008.0.1.zip files from your account manager. Both packages are shipped in the Model Bank package.

**Procedure**

1. Upload the UXP-Browser.202008.zip file to the $UXPBROWSER directory on the application server.
2. In the Linux shell, navigate to $UXPBROWSER and extract the files from the UXP-Browser.202008.zip file.

unzip UXP-Browser.202008.zip

The following files are extracted:

* + Authenticator-202008.0.1.war
  + Browser-202008.0.1.war
  + browser-iris-202008.0.1.war
  + browser-iris-tafc-202008.0.1.war
  + irf-rp-services-202008.0.1.war
  + ResourceServerWar-202008.0.1.war
  + StaticFiles-202008.1.zip
  + uxpb-god-properties-202008.0.1.zip
  + UXPB-Tools-202008.0.1.zip

1. Locate the IRISR18\_CatalogServices\_TAFJ\_202008.0.1.zip file in your Model Bank package on your local machine and extract it.
2. Upload the extracted irf-t24catalog-services-202008.0.1.war file to the $UXPBROWSER directory on your application server.
3. Optional: Rename the required artefact file names by removing the release version information:

mv Authenticator-202008.0.1.war Authenticator.war

mv Browser-202008.0.1.war Browser.war

mv irf-rp-services-202008.0.1.war irf-rp-services.war

mv irf-t24catalog-services-202008.0.1.war irf-t24catalog-services.war

mv ResourceServerWar-202008.0.1.war ResourceServer.war

**Configuring and deploying Transact and TAFJ artefacts**

The final step in this setup is to deploy the Transact UI components UXP Browser (UXPB) and BrowserWeb. UXPB is the Temenos' latest user interface for working with Transact. BrowserWeb is its antecedent. This runbook does not cover deployment of other product user interfaces such as Temenos Connect Internet Banking (TCIB) and Mobile Banking (TCMB).

**Deploying TAFJJEE\_EAR.ear and TAFJ Spooler plugins**

**Procedure**

1. In the Linux shell, navigate to $TAFJ\_HOME/appserver.
2. Copy the TAFJSpoolerPlugins.rar file to the $JBOSS\_HOME/standalone/deployments directory.

cp TAFJSpoolerPlugins.rar $JBOSS\_HOME/standalone/deployments

1. Change to $TAFJ\_HOME/appserver/jboss/jboss7eap.
2. Copy TAFJJEE\_EAR.ear to the $JBOSS\_HOME/standalone/deployments directory.

cp TAFJJEE\_EAR.ear $JBOSS\_HOME/standalone/deployments

**Deploying BrowserWeb**

**Before you begin**

You need to obtain the BrowserWeb-202008.00.tar file from your account manager. BrowserWeb is not shipped in the Model Bank package.

**Procedure**

1. Upload the BrowserWeb-202008.00.tar file to the $TEMENOS\_HOME/inst directory.
2. At the Linux command line, navigate to $TEMENOS\_HOME/inst and extract the TAR file.

tar -xvf BrowserWeb-202008.00.tar

1. Navigate to the extracted BrowserWeb-202008.00 directory and copy BrowserWeb.war to $JBOSS\_HOME/standalone/deployments.

cd BrowserWeb-202008.00

cp BrowserWeb.war $JBOSS\_HOME/standalone/deployments

**Modifying Browser.war**

**Procedure**

1. At the Linux command line, navigate to $UXPBROWSER.
2. Extract Browser.war in a temporary directory under $UXPBROWSER, for example tmp-uxpb.

mkdir tmp-uxpb

mv Browser.war tmp-uxpb

cd tmp-uxpb

jar -xf Browser.war

1. Navigate to WEB-INF and open SSOAPI.properties in an editor.

cd WEB-INF

vi SSOAPI.properties

1. Provide the port number and IP address or host name of your application server. Example:

SSOAPI.T24REMOTE.URL=http://**10.23.50.153:9089**/Authenticator/auth

1. Save the file.
2. Delete the previous version of the WAR file.

cd ..

rm Browser.war

1. Re-package the WAR file and copy it back to the UXPBrowser directory.

jar cf Browser.war \*

cp Browser.war $UXPBROWSER

**Modifying irf-rp-services.war**

**Procedure**

1. At the Linux command line, change to $UXPBROWSER.
2. Extract irf-rp-services.war to a temporary directory, for example tmp-irf-rp-srv.

mkdir tmp-irf-rp-srv

mv irf-rp-services.war tmp-irf-rp-srv

cd tmp-irf-rp-srv

jar -xf irf-rp-services.war

1. Navigate to WEB-INF/classes.
2. Open connection.properties in your text editor and provide the IP address or host name of your application server and the port. Example:
3. # Common EJB Remote Configurations
4. RemoteConnectionHosts=**10.23.50.153**
5. RemoteConnectionPorts=**9089**
6. transportLayerUser=**SSOUSER1**

transportLayerCredential=**123456**

1. Delete the previous version of the WAR file.

cd ../../

rm irf-rp-services.war

1. Re-package the WAR file and copy it back to the $UXPBrowser directory.

jar cf irf-rp-services.war \*

cp Browser.war $UXPBROWSER

**Modifying irf-t24catalog-services.war**

**Procedure**

1. At the Linux command line, change to $UXPBROWSER.
2. Extract irf-t24catalog-services.war to a temporary subdirectory in $UXP\_BROWSER, for example tmp-irf-cat-srv.

mkdir tmp-irf-cat-srv

cp irf-t24catalog-services.war tmp-irf-cat-srv

cd tmp-irf-cat-srv

jar -xf irf-t24catalog-services.war

1. Navigate to WEB-INF/classes.
2. Open connection.properties in your text editor and provide the IP address or host name of your application server and the port. Example:
3. # Common EJB Remote Configurations
4. RemoteConnectionHosts=**10.23.50.153**
5. RemoteConnectionPorts=**9089**
6. transportLayerUser=**SSOUSER1**

transportLayerCredential=**123456**

1. Delete the previous version of the WAR file.

cd ../../

rm irf-t24catalog-services.war

1. Re-package the WAR file and copy it back to the UXPBrowser directory.

jar cf irf-t24catalog-services.war \*

cp irf-t24catalog-services.war $UXPBROWSER

**Embedding the database driver in ResourceServer.war**

**Procedure**

1. At the Linux command line, change to $UXPBROWSER.
2. Extract ResourceServer.war to a temporary subdirectory in $UXP\_BROWSER, for example tmp-res-srv.

mkdir tmp-res-srv

cp ResourceServer.war tmp-res-srv

cd tmp-res-srv

jar -xf ResourceServer.war

1. Copy the database driver(s) to ResourceServerWar.war/WEB-INF/lib.

**MS SQL Server**

cp $TAFJ\_HOME/ext/sqljdbc42.jar $UXPBROWSER/tmp-res-srv/WEB-INF/lib

**NuoDB**

cp $TAFJ\_HOME/ext/nuodb-jdbc-21.0.0.jar $UXPBROWSER/tmp-res-srv/WEB-INF/lib

**Oracle**

cp $TAFJ\_HOME/ext/ojdbc8.jar $UXPBROWSER/tmp-res-srv/WEB-INF/lib

**PostgreSQL**

cp $TAFJ\_HOME/ext/postgresql-42.2.6.jar $UXPBROWSER/tmp-res-srv/WEB-INF/lib

1. Delete the previous version of the WAR file.

rm ResourceServer.war

1. Re-package the WAR file and copy it back to the UXPBrowser directory.

jar cf ResourceServer.war \*

mv ResourceServer.war $UXPBROWSER

**Modifying the required EJB JAR files**

**Procedure**

1. To prepare the required EJB JAR files, copy them to the $UXPBROWSER directory:

cd $TEMENOS\_HOME/T24/bnk/Extensions/EB\_AuthenticationService/t24ejb

cp t24-EB\_AuthenticationService-ejb.jar $UXPBROWSER

cd $TEMENOS\_HOME/T24/bnk/Extensions/EB\_CatalogService/t24ejb

cp t24-EB\_CatalogService-ejb.jar $UXPBROWSER

cd $TEMENOS\_HOME/T24/bnk/Extensions/EB\_ResourceProviderService/t24ejb

cp t24-EB\_ResourceProviderService-ejb.jar $UXPBROWSER

1. Modify t24-EB\_AuthenticationService-ejb.jar.
   1. Copy and extract the file and in a temporary subdirectory under $UXPBROWSER, for example tmp-ejb-auth.

cd $UXPBROWSER

mv t24-EB\_AuthenticationService-ejb.jar tmp-ejb-auth

cd tmp-ejb-auth

jar -xf t24-EB\_AuthenticationService-ejb.jar

* 1. Navigate to META-INF and open the ejb-jar.xml file in an editor.
  2. Change the <transactiontype> value from Container to **Bean**, as illustrated below.

A computer code with text

Description automatically generated

* 1. In <env-entry-value>, specify either IRISPA or IFPA as the OFS\_SOURCE.

A close-up of text

Description automatically generated

* 1. Save the file.
  2. Open jboss-ejb3.xml file in an editor and ensure that the security domain is set to other.

A close-up of a computer code

Description automatically generated

* 1. Save the file.
  2. Re-package the JAR file and upload it to the UXPBrowser directory.

cd ..

rm t24-EB\_AuthenticationService-ejb.jar

jar cf t24-EB\_AuthenticationService-ejb.jar \*

mv t24-EB\_AuthenticationService-ejb.jar $UXPBROWSER

1. Modify t24-EB\_CatalogService-ejb.jar.
   1. Copy and extract the file and in a temporary subdirectory under $UXPBROWSER, for example tmp-ejb-cat-srv.

cd $UXPBROWSER

mv t24-EB\_CatalogService-ejb.jar tmp-ejb-cat-srv

cd tmp-ejb-cat-srv

jar -xf t24-EB\_CatalogService-ejb.jar

* 1. Navigate to META-INF and open the ejb-jar.xml file in an editor.
  2. Change the <transactiontype> value from Container to **Bean**, as illustrated below.

A computer code with text

Description automatically generated

* 1. In <env-entry-value>, specify either IRISPA or IFPA as the OFS\_SOURCE.

A close-up of a computer code

Description automatically generated

* 1. Save the file.
  2. Open jboss-ejb3.xml file in an editor and ensure that the security domain is set to other.

A close-up of a computer code

Description automatically generated

* 1. Save the file.
  2. Re-package the JAR file and upload it to the UXPBrowser directory.

cd ..

rm t24-EB\_CatalogService-ejb.jar

jar cf t24-EB\_CatalogService-ejb.jar \*

mv t24-EB\_CatalogService-ejb.jar $UXPBROWSER

1. Perform the same steps as above to modify ejb-jar.xml and jboss-ejb3.xml in t24-EB\_ResourceProviderService-ejb.jar:
   1. Copy and extract the file and in a temporary subdirectory under $UXPBROWSER, for example tmp-ejb-res-prov-srv.

cd $UXPBROWSER

mv t24-EB\_ResourceProviderService-ejb.jar tmp-ejb-res-prov-srv

cd tmp-ejb-res-prov-srv

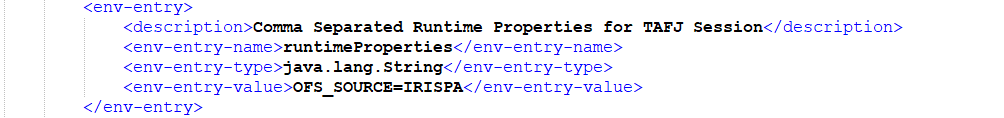
jar -xf t24-EB\_ResourceProviderService-ejb.jar

* 1. Navigate to META-INF and open the ejb-jar.xml file in an editor.
  2. Change the <transactiontype> value from Container to **Bean**, as illustrated below.

A screenshot of a computer program

Description automatically generated

* 1. In <env-entry>, specify either IRISPA or IFPA as the OFS\_SOURCE.



* 1. Save the file.
  2. Open jboss-ejb3.xml file in an editor and ensure that the security domain is set to other.

A close-up of a computer screen

Description automatically generated

* 1. Save the file.
  2. Re-package the JAR file and upload it to the UXPBrowser directory.

cd ..

rm t24-EB\_ResourceProviderService-ejb.jar

jar cf t24-EB\_ResourceProviderService-ejb.jar \*

mv t24-EB\_ResourceProviderService-ejb.jar $UXPBROWSER

**Configuring the SSOUSER1 user in JBoss**

You need to create the SSOUSER1 user which will be used by the Resource Provider (irf-rp-services.war) and Catalog (irf-t24catalog-services.war) services.

**Procedure**

1. At the Linux command line, navigate to $JBOSS\_HOME/bin.
2. Run the add-user.sh script:

./add-user.sh

1. Specify **Application user** (option b) as the type of user.
2. What type of user do you wish to add?
3. a) Management User (mgmt-users.properties)
4. b) Application User (application-users.properties)

**(a):b**

1. Provide the SSOUSER1 user name and password (here 123456).
2. aEnter the details of the new user to add.
3. Using realm 'ApplicationRealm' as discovered from the existing property files.
4. Username : **SSOUSER1**
5. Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.
6. - The password should be different from the username
7. - The password should not be one of the following restricted values {root, admin, administrator}
8. - The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)

Password :

1. Confirm that you want to use the password and re-enter it.
2. WFLYDM0099: Password should have at least 8 characters!
3. Are you sure you want to use the password entered yes/no? **yes**
4. Re-enter Password :
5. Specify the group the user should belong to; here t24user.

What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[ ]: **t24user**

1. Confirm the addition of the user to the realm ApplicationRealm by providing the answer yes.
2. About to add user 'SSOUSER1' for realm 'ApplicationRealm'

Is this correct yes/no? **yes**

The user has been added to JBoss configuration files.

Added user 'SSOUSER1' to file '/srv/Temenos/3rdParty/AS/jboss-eap-7.3/standalone/configuration/application-users.properties'

Added user 'SSOUSER1' to file '/srv/Temenos/3rdParty/AS/jboss-eap-7.3/domain/configuration/application-users.properties'

Added user 'SSOUSER1' with groups t24user to file '/srv/Temenos/3rdParty/AS/jboss-eap-7.3/standalone/configuration/application-roles.properties'

Added user 'SSOUSER1' with groups t24user to file '/srv/Temenos/3rdParty/AS/jboss-eap-7.3/domain/configuration/application-roles.properties'

1. For the last question, provide the answer yes.
2. Is this new user going to be used for one AS process to connect to another AS process?
3. e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server EJB calls.

yes/no? **yes**

The user has been created and the following message is displayed at the command line interface:

To represent the user add the following to the server-identities definition

<secret value="MTIzNDU2" />

1. Open the Transact.xml file that is in /srv/Temenos/3rdParty/AS/jboss-eap-7.2/standalone/configuration in a text editor and add under <server-identities> the tag that you generated in the previous step:
2. <server-identities>
3. <ssl>
4. <keystore path="application.keystore" relative-to="jboss.server.config.dir" keystore-password="password" alias="server" key-password="password" generate-self-signed-certificate-host="localhost"/>
5. </ssl>
6. **<secret value="MTIzNDU2" />**
7. </server-identities>

**Configuring UXP Browser in GOD mode**

**Before you begin**

You need to locate the encryption utility (EncryptPassword.jar) in your Model Bank package and then upload it to the $UXPBROWSER directory on the application server. The JAR file is available at the following location:  
*Model\_Bank\_package*.zip/UXP-Browser.zip/UXPB-Tools.zip/EncryptPassword.jar

**Procedure**

1. At the command line, change to $TEMENOS\_HOME and create the properties directory.

cd $TEMENOS\_HOME

mkdir properties

1. Change to the $UXPBROWSER directory and extract the uxpb-god-properties-202008.0.1.zip file.

cd $UXPBROWSER

unzip uxpb-god-properties-202008.0.1.zip

1. Ccopy the two properties files to the directory where they will be stored permanently:

cp BRPRuntimeProperties.properties $TEMENOS\_HOME/properties

cp RSDefaultProperties.properties $TEMENOS\_HOME/properties

1. Encrypt the IRIS system and JDBC user passwords using the Temenos Encryption Utility (EncryptPassword.jar) – run the following command:

java -jar EncryptPassword.jar *pwd\_to\_be\_encrypted*

Example:

java -jar EncryptPassword.jar 123456

1. Save the encrypted password in a secure location.
2. Edit the BRPRuntimeProperties.properties file in a text editor - perform the following steps:
   1. Provide the Transact user name and the encrypted password (here 123456):
   2. browser.options.irisSystemUser=**AUTHOR**

browser.options.irisSystemPassword=**vSd3/5UdSJc=**

* 1. Uncomment the following lines and provide the Resource Provider, Catalog services, Resource Server and Authenticator URLs with the IP address or host name of your application server server:
  2. browser.options.dynamicIRIS.rpService=**http://10.23.50.153:9089/irf-rp-services**
  3. browser.options.god.catalogService=**http://10.23.50.153:9089/irf-t24catalog-services**
  4. browser.options.resourceServer.url=**http://10.23.50.153:9089/ResourceServer**browser.options.remoteAuth.url=**http://10.23.50.153:9089/Authenticator/auth/logon**
  5. Save the file.

1. Edit the RSDefaultProperties.properties file in a text editor – provide the following and save the file:
   1. Database URL
   2. Database user name
   3. The encrypted password for the database user
   4. Blob type
   5. Long type

Example:

aresource.server.options.tenant.jdbc.url.1=**jdbc:oracle:thin:@10.23.50.151:1521/MB202008**

resource.server.options.tenant.jdbc.username.1=**t24**

resource.server.options.tenant.jdbc.password.1=**g7nVhB+BoUY=**

resource.server.options.tenant.jdbc.driver.1=**oracle.jdbc.OracleDriver**

resource.server.options.tenant.ddl.blobType.1=**blob**

resource.server.options.tenant.ddl.longType.1=**number(19)**

**Resource Server database parameters for supported databases**

**Microsoft SQL Server**

resource.server.options.tenant.jdbc.driver.1=**com.microsoft.sqlserver.jdbc.SQLServerDriver**

resource.server.options.tenant.ddl.blobType.1=**varbinary(max)**

resource.server.options.tenant.ddl.longType.1=**bigint**

**NuoDB**

resource.server.options.tenant.jdbc.driver.1=**com.nuodb.jdbc.Driver**

resource.server.options.tenant.ddl.blobType.1=**blob**

resource.server.options.tenant.ddl.longType.1=**bigint**

**Oracle Database**

resource.server.options.tenant.jdbc.driver.1=**oracle.jdbc.OracleDriver**

resource.server.options.tenant.ddl.blobType.1=**blob**

resource.server.options.tenant.ddl.longType.1=**number(19)**

**PostgreSQL**

resource.server.options.tenant.jdbc.driver.1=**org.postgresql.Driver**

resource.server.options.tenant.ddl.blobType.1=**bytea**

resource.server.options.tenant.ddl.longType.1=**bigint**

**Deploying UXP Browser artefacts in JBoss**

The UXP Browser artefact is Browser.war. UXP Browser requests are routed through Interaction Framework. Therefore, along with the Browser, you must also deploy the Authenticator application , Resource Provider and Interaction Framework catalog services and also Resource server. This section is not applicable if you have skipped [Extracting UXP Browser artefacts](http://172.16.204.9/R21UG/Solutions/Runbooks/Stack-01-Runbook-R21/t_extracting-uxp-browser-artefacts.htm).

**Procedure**

1. At the Linux command line, change to $UXPBROWSER.
2. Deploy the Authenticator component.

cp Authenticator.war $JBOSS\_HOME/standalone/deployments

1. Deploy Browser.war.

cp Browser.war $JBOSS\_HOME/standalone/deployments

1. Deploy Interaction Framework Resource Provider services.

cp irf-rp-services.war $JBOSS\_HOME/standalone/deployments

1. Deploy Interaction Framework Catalog services.

cp irf-t24catalog-services.war $JBOSS\_HOME/standalone/deployments

1. Deploy Resource Server.

cp ResourceServer.war $JBOSS\_HOME/standalone/deployments

**Deploying the EJB JAR files in JBoss**

**Procedure**

1. Deploy the Authentication service.

cd $UXPBROWSER

cp t24-EB\_AuthenticationService-ejb.jar $JBOSS\_HOME/standalone/deployments

1. Deploy the Catalog Service component.

cp t24-EB\_CatalogService-ejb.jar $JBOSS\_HOME/standalone/deployments

1. Deploy the Resource Provider service.

cp t24-EB\_ResourceProviderService-ejb.jar $JBOSS\_HOME/standalone/deployments

**Installing UXP Browser help text**

**Procedure**

1. Upload T24.202008.HELPTEXT.zip to a temporary directory on the application server.
2. Extract the T24.202008.HELPTEXT.zip file to $TEMENOS\_HOME.

unzip T24.202008.HELPTEXT.zip -d $TEMENOS\_HOME

The help text XML files are extracted to the HELP.TEXT directory.



1. Add the help text directory path argument to your JBoss startup script.

$JBOSS\_HOME/bin/standalone.sh --server-config=Transact.xml -Djboss.http.port=9089 -Djboss.node.name=node1 -b 0.0.0.0 -DedgeSystemPropertyFolder=/home/temenos/Temenos/properties -DedgeSystemPropertyUpdaters=com.temenos.connect.system.GODSystemTestProperties -DBRP\_HOME=/home/temenos/Temenos **-Dt24.helptext.xml.directory=/srv/Temenos/HELP.TEXT/HelpText**

1. Restart the application server.
2. Verify that the text is displayed correctly in the Transact user interface:
   1. Log in to UXP Browser interface and open an application, for example DATES.
   2. Double-click a label in the UI.

The context sensitive help is displayed

**Deploying Transact component services**

You will find the Transact component service APIs as Axis2 archives (t24-*component\_service*-jws.aar) in the extensions folder of the Transact distribution pack you have received. Deploy the archives into the Apache Axis2 web archive file. The release version of Apache Axis2 used in this runbook is 1.6.2. The topics in this chapter show you how to deploy axis2.war on your JBoss application server.

**Packaging Axis2 archives**

**Procedure**

1. Download axis2-1.6.2-war.zip from the Apache website to a directory on your computer.
2. Extract the contents of the compressed file.
3. Upload axis2.war to the $UXPBROWSER directory.
4. In the Linux shell, change to the $UXPBROWSER directory and create a new temporary directory, for example tmp-axis2.

mkdir tmp-axis2

1. Copy axis2.war to the tmp-axis2 directory.

cp axis2.war tmp-axis2

1. In the tmp8 directory, extract the contents of the axis2.war file and navigate to WEB-INF/lib.

cd tmp-axis2  
jar -xf axis2.war  
cd WEB-INF/lib

1. Add the following dependency JAR files to the lib directory.
   * commons-pool-1.5.5.jar
   * spring-2.5.6.jar
2. Copy the component service archives from $T24\_HOME/bnk/Extensions into the /WEB-INF/services directory of the axis2.war file.

cp $TEMENOS\_HOME/T24/bnk/Extensions/DS\_DesignStudioInstallerService/t24ws/t24-DS\_DesignStudioInstallerService-tafj-jws.aar $UXPBROWSER/tmp-axis2/WEB-INF/services

cp $TEMENOS\_HOME/T24/bnk/Extensions/EB\_CatalogService/t24ws/t24-EB\_CatalogService-tafj-jws.aar $UXPBROWSER/tmp-axis2/WEB-INF/services

cp $TEMENOS\_HOME/T24/bnk/Extensions/EB\_OFSConnectorService/t24ws/t24-EB\_OFSConnectorService-tafj-jws.aar $UXPBROWSER/tmp-axis2/WEB-INF/services

cp $TEMENOS\_HOME/T24/bnk/Extensions/EB\_ResourceProviderService/t24ws/t24-EB\_ResourceProviderService-tafj-jws.aar $UXPBROWSER/tmp-axis2/WEB-INF/services

cp $TEMENOS\_HOME/T24/bnk/Extensions/IF\_IntegrationFlowService/t24ws/t24-IF\_IntegrationFlowService-tafj-jws.aar $UXPBROWSER/tmp-axis2/WEB-INF/services

cp $TEMENOS\_HOME/T24/bnk/Extensions/IF\_IntegrationFrameworkService/t24ws/t24-IF\_IntegrationFrameworkService-tafj-jws.aar $UXPBROWSER/tmp-axis2/WEB-INF/services

cp $TEMENOS\_HOME/T24/bnk/Extensions/IF\_IntegrationLandscapeService/t24ws/t24-IF\_IntegrationLandscapeService-tafj-jws.aar $UXPBROWSER/tmp-axis2/WEB-INF/services

1. Open the /WEB-INF/services/services.list file and add the names of the component service archives that you added to the services directory.

A screen shot of a computer program

Description automatically generated

1. Open /WEB-INF/web.xml in an editor and configure database connectivity by adding a data source.
2. <!-- Temenos T24 Service Provider Specific Settings -->
3. <resource-ref id="ResourceRef\_t24DataSource">
4. <description>Used to get connections from T24 jdbc pool</description>
5. <res-ref-name>jdbc/t24DataSource</res-ref-name>
6. <res-type>javax.sql.DataSource</res-type>
7. <res-auth>Container</res-auth>
8. </resource-ref>
9. <resource-ref id="ResourceRef\_t24LockingDataSource">
10. <description>Used to get connections from T24 jdbc pool</description>
11. <res-ref-name>jdbc/t24LockingDataSource</res-ref-name>
12. <res-type>javax.sql.DataSource</res-type>
13. <res-auth>Container</res-auth>

</resource-ref>

A computer screen shot of a program

Description automatically generated

1. Add the JBoss-specific deployment descriptor jboss-web.xml under the WEB-INF directory and configure database connectivity by adding data sources.
2. <?xml version="1.0" encoding="UTF-8"?>
3. <jboss-web>
4. <resource-ref>
5. <res-ref-name>jdbc/t24DataSource</res-ref-name>
6. <res-type>javax.sql.DataSource</res-type>
7. <jndi-name>java:/jdbc/t24Axis2DS</jndi-name>
8. </resource-ref>
9. <resource-ref>
10. <res-ref-name>jdbc/t24LockingDataSource</res-ref-name>
11. <res-type>javax.sql.DataSource</res-type>
12. <jndi-name>java:/jdbc/t24LockingDS</jndi-name>
13. </resource-ref>
14. </jboss-web>

A screen shot of a computer code

Description automatically generated

1. Repackage axis2.war and place it in the UXPBrowser directory.

cd $UXPBROWSER/tmp-axis2  
rm –rf axis2.war  
jar cf axis2.war \*  
cp axis2.war $UXPBROWSER

**Deploying Axis2**

**Procedure**

1. In the Linux shell, navigate to $JBOSS\_HOME/standalone/configuration.
2. Open Transact.xml in an editor and add Axis2 data source under the datasource subsystem for axis2.war to get it deployed.

**Microsoft SQL Server**

<datasource jta="true" jndi-name="java:/jdbc/t24Axis2DS" pool-name="Axis2DS" enabled="true" use-java-context="true" use-ccm="true">

<connection-url>**jdbc:sqlserver://10.23.50.XXX: 1433;databaseName=xxxx;integratedSecurity=false**</connection-url>

<connection-property name="defaultRowPrefetch">100</connection-property>

<driver>sqljdbc</driver>

<pool>

<min-pool-size>5</min-pool-size>

<max-pool-size>220</max-pool-size>

<flush-strategy>FailingConnectionOnly</flush-strategy>

</pool>

<security>

<user-name>t24</user-name>

<password>t24</password>

</security>

</datasource>

**NuoDB**

<datasource jta="true" jndi-name="java:/jdbc/t24Axis2DS" pool-name="Axis2DS" enabled="true" use-java-context="true" use-ccm="true">

<connection-url>**jdbc:com.nuodb://10.23.50.138:48004/MB202008?SCHEMA=USER**</connection-url>

<connection-property name="defaultRowPrefetch">100</connection-property>

<driver>**nuodb-jdbc**</driver>

<pool>

<min-pool-size>5</min-pool-size>

<max-pool-size>220</max-pool-size>

<flush-strategy>FailingConnectionOnly</flush-strategy>

</pool>

<security>

<user-name>dba</user-name>

<password>dba</password>

</security>

</datasource>

**Oracle Database**

<datasource jta="true" jndi-name="java:/jdbc/t24Axis2DS" pool-name="Axis2DS" enabled="true" use-java-context="true" use-ccm="true">

<connection-url> **jdbc:oracle:thin:@10.23.50.151:1521/MB202008**</connection-url>

<connection-property name="defaultRowPrefetch">100</connection-property>

<driver>**ora18c**</driver>

<pool>

<min-pool-size>5</min-pool-size>

<max-pool-size>220</max-pool-size>

<flush-strategy>FailingConnectionOnly</flush-strategy>

</pool>

<security>

<user-name>t24</user-name>

<password>t24</password>

</security>

</datasource>

**PostgreSQL**

<datasource jta="true" jndi-name="java:/jdbc/t24Axis2DS" poolname="

Axis2DS" enabled="true" use-java-context="true" use-ccm="true">

<connection-url>

**jdbc:postgresql://192.168.0.2:5432/XXXXXX?autosave=always**</connection-url>

<connection-property name="defaultRowPrefetch">100</connectionproperty>

<driver>**postgres-jdbc**</driver>

<pool>

<min-pool-size>5</min-pool-size>

<max-pool-size>220</max-pool-size>

<flush-strategy>FailingConnectionOnly</flush-strategy>

</pool>

<security>

<user-name>t24</user-name>

<password>t24</password>

</security>

</datasource>

1. Copy axis2.war from the UXPBrowser directory to $JBOSS\_HOME/standalone/deployments.

cp axis2.war $JBOSS\_HOME/standalone/deployments

**Starting the application server**

Use the standalone.sh script that is in $JBOSS\_HOME/bin to start JBoss. The script requires several parameters depending on what needs to be deployed.

**Starting JBoss with BrowserWeb**

Use the following command to access BrowserWeb.

./standalone.sh -b 0.0.0.0 -bmanagement 0.0.0.0 --server-config=Transact.xml -Djboss.http.port=9089

**Starting JBoss with UXP Browser**

To access UXP Browser use the following command. (Each command argument has been placed in a new line for brevity.)

$JBOSS\_HOME/bin/standalone.sh

--server-config=Transact.xml

-Djboss.http.port=9089

-Djboss.node.name=node1

-b 0.0.0.0

-DedgeSystemPropertyFolder=/srv/Temenos/properties

-DedgeSystemPropertyUpdaters=com.temenos.connect.system.GODSystemTestProperties

-DBRP\_HOME=/srv/Temenos

**Tip:**

Optionally, you may set JBoss HTTP port (-Djboss.http.port=9089) to start JBoss at a user-defined port instead of the default HTTP port 8080.

When the server is started, go to the JBoss deployments directory to see all the deployed artefacts.

A screen shot of a computer

Description automatically generated

**Mandatory command arguments**

| Arguments | Explanations |
| --- | --- |
| --server-config | Indicates standalone profile configuration file (for example Transact.xml) to use at JBoss startup. |
| -Djboss.http.port | Port number of the application server. |
| -Djboss.node.name | Unique identifier for the application server instance. |
| -DedgeSystemPropertyUpdaters | This argument is used to set the required properties to be loaded. For GOD mode, use the value below:  com.temenos.connect.system.GODSystemTestProperties |

 Dark Mode

[**Expand/Collapse**](http://172.16.204.9/R21UG/Solutions/Runbooks/Stack-01-Runbook-R21/t_starting-the-application-server.htm?tocpath=Sanity%20check%20(post%20deployment)%7C_____1)

[**Share**](http://172.16.204.9/R21UG/Solutions/Runbooks/Stack-01-Runbook-R21/t_starting-the-application-server.htm?tocpath=Sanity%20check%20(post%20deployment)%7C_____1)

[**Print**](http://172.16.204.9/R21UG/Solutions/Runbooks/Stack-01-Runbook-R21/t_starting-the-application-server.htm?tocpath=Sanity%20check%20(post%20deployment)%7C_____1)

In this topic

* [Starting the application server](http://172.16.204.9/R21UG/Solutions/Runbooks/Stack-01-Runbook-R21/t_starting-the-application-server.htm?tocpath=Sanity%20check%20(post%20deployment)%7C_____1#Startingtheapplicationserver)
  + [Starting JBoss with BrowserWeb](http://172.16.204.9/R21UG/Solutions/Runbooks/Stack-01-Runbook-R21/t_starting-the-application-server.htm?tocpath=Sanity%20check%20(post%20deployment)%7C_____1#StartingJBosswithBrowserWeb)
  + [Starting JBoss with UXP Browser](http://172.16.204.9/R21UG/Solutions/Runbooks/Stack-01-Runbook-R21/t_starting-the-application-server.htm?tocpath=Sanity%20check%20(post%20deployment)%7C_____1#StartingJBosswithUXPBrowser)
  + [Mandatory command arguments](http://172.16.204.9/R21UG/Solutions/Runbooks/Stack-01-Runbook-R21/t_starting-the-application-server.htm?tocpath=Sanity%20check%20(post%20deployment)%7C_____1#Mandatorycommandarguments)

**Accessing TAFJEE**

TAFJJEE\_EAR.ear file also contains a WAR file, which is a helper servlet for configuration, diagnostic, execution and troubleshooting TAFJ-related functions on the application server. You can access this helper servlet at http://*hostname\_or\_ipaddress*:9089/TAFJEE. TAFJEE has protected access and is subject to BASIC authentication.

**Procedure**

1. In the Linux shell, navigate to $JBOSS\_HOME/bin. Create an application user by executing add-user.sh.
2. Give the user **TAFJAdmin** role.
   1. Specify the type of user, here **Application** user.
   2. Provide the preferred user name and password (here the user is jumpstart and the password is Temenos@123456).
   3. Specify the groups the user should belong to; here TAFJAdmin.
   4. Confirm the addition of the user to the realm ApplicationRealm by providing the answer yes.
   5. For the last question, provide the answer no.

A screenshot of a computer program

Description automatically generated

1. Access the TAFJEE servlet by using the following URL:  
   http://*hostname\_or\_ipaddress*:9089/TAFJEE

A screenshot of a computer

Description automatically generated

1. Supply the application user credentials. You can get access to all the functions in TAFJEE.

**Note:**

Some of the tools are protected and require TAFJ user credentials.

1. Click the tShow servlet to view the compilation details of a Transact routine on the application server.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Accessing BrowserWeb**

**Procedure**

1. Access BrowserWeb using the following URL:  
   http://*hostname\_or\_ipaddress*:9089/BrowserWeb.
2. Log in using a valid Transact user name and password.

A screenshot of a computer

Description automatically generated

After a successful login, the landing page is displayed.

A screenshot of a computer

Description automatically generated

1. Launch any Transact application. For example, SPF S SYSTEM.

A screenshot of a computer

Description automatically generated

**Accessing UXP Browser**

**Procedure**

1. Access UXP Browser using the following URL:  
   http://*hostname\_or\_ipaddress*:9089/Browser/
2. Log in using a valid Transact user ID and password.

A screenshot of a computer

Description automatically generated

On successful login the landing page is displayed.

A screenshot of a computer

Description automatically generated

1. Launch a Transact application, for example, SPF S SYSTEM.

A screenshot of a computer

Description automatically generated

**Accessing Axis2 servlet**

**Procedure**

1. Access the Axis2 servlet using the following URL:  
   http://*hostname\_or\_ipaddress*:9089/axis2
2. Click **Validate**.

A screenshot of a computer

Description automatically generated

A page is displayed that confirms whether the web services are working properly.

A screenshot of a computer software

Description automatically generated

1. On the main page, click **Services** to check the available web services.

A screenshot of a computer program

Description automatically generated

**Performing an integrated test of Axis2 and Transact**

**Procedure**

1. In Axis2 servlet, go to **Services** and click the chosen service to get the WSDL file. For example, here the WSDL is of IntegrationFlowService.

A screenshot of a computer code

Description automatically generated

1. Create a new SOAP project using WSDL in a third-party tool such as SOAPUI.

A screenshot of a computer program

Description automatically generated

1. Create a new read flow request passing the necessary parameters.

A screenshot of a computer

Description automatically generated

The SOAP response indicates that IntegrationFlowService of Axis2 can communicate with Transact.

You have now completed the deployment of TAFJ-Transact on JBoss EAP 7.3 with a supported database (MS SQL Server 2019, Oracle 19c or PostgreSQL 11/12). You will now be able to perform business operations in Transact using UXP Browser (or BrowserWeb).