TEMENOS™

TAFJ-AS TAFJ

R15

10/3/2015

Temenos

Amendment History:

Revision	Date Amended	Name	Description
1	1st April 2011	TAFJ team	Initial version
2	7st February 2012	H. Aubert	R12GA review
3	15 January 2013	JN. Charpin	R13GA review
4	20th February 2013	R. Vincent	R14GA review
5	19 March 2014	JN.Charpin	Servlet review
6	24th June 2014	JN. Charpin	Entry points documentation, Webservices
7	20th August 2014	JN. Charpin	TAFJ technical monitor and webapp basic authentication
8	26th September 2014	R. Vincent	Added TAFJ Sessions monitor documentation
9	5 February 2015	JN. Charpin	Technical monitor 1.54-TEMN
10	6th March 2015	H. Aubert	R15 AMR review
11	13th April 2015	JN. Charpin	New servlet functionalities
12	18th May 2015	JN.Charpin	Container managed EJB - review
13	2nd September 2015	JN. Charpin	Bean managed EJB.

Copyright

Errata and Comments

If you have any comments regarding this manual or wish to report any errors in the documentation, please document them and send them to the address below:
Technology Department

Temenos Headquarters SA
2 Rue de l'Ecole-de-Chimie,
CH - 1205 Geneva,
Switzerland

Tel SB: +41 (0) 22 708 1150
Fax: +41 (0) 22 708 1160

Please include your name, company, address, and telephone and fax numbers, and email address if applicable. TAFJdev@temenos.com

Table of Contents

Overview

TAFJ JEE application is packaged within TAFJJEE_EAR.ear file.

Depending on your application server version you have to get the corresponding ear file from TAFJ_HOME/appserver/YOUR_APPSERVER_PROVIDER/VERSION

- TAFJJEE_EAR.ear contains :
    - TAFJJEE_MDB.jar : Message Driven Beans to handle JMS messages
    - TAFJJEE_EJB.jar : Enterprise Java Beans to process T24 requests
    - TAFJJEE_WAR_TAFJ.war : Set of servlet and utilities
    - TAFJJEE_Services.war : Webservices to process T24 requests

MDB is used in this document for “Message Driven Bean”.

EJB is used in this document for “Enterprise Java Bean”.

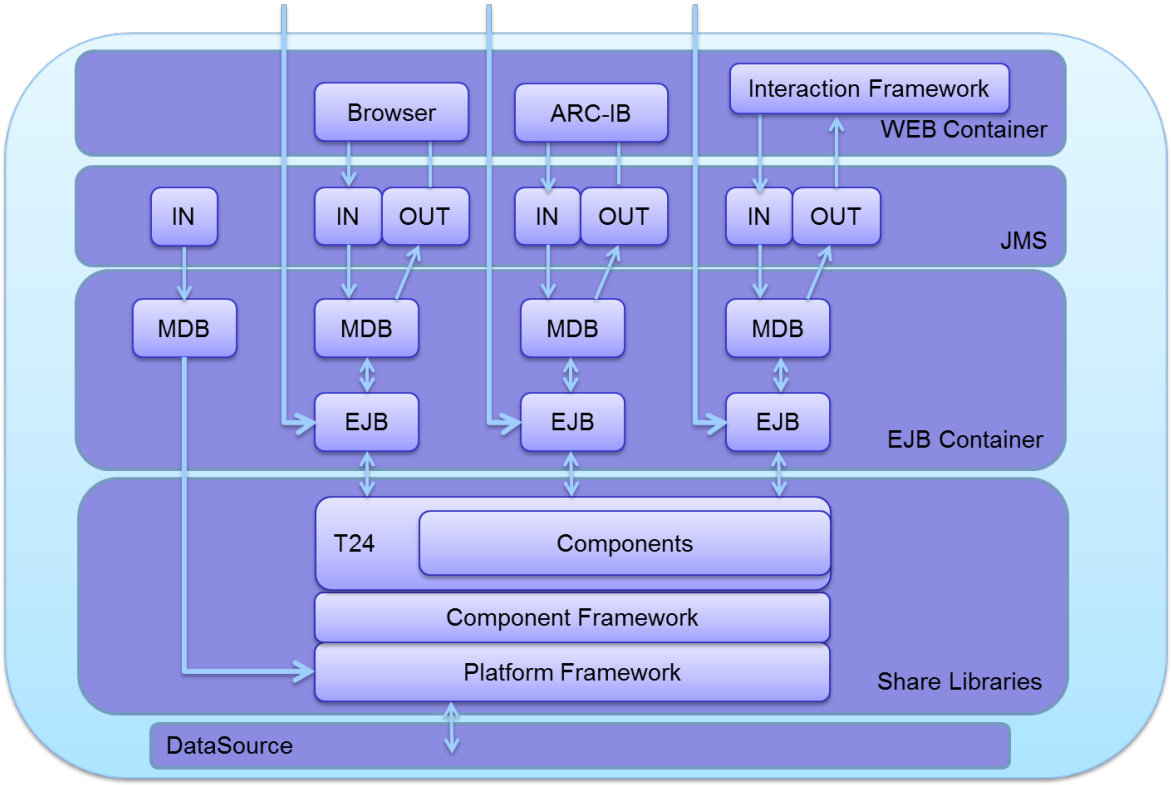
Application server configuration

Explanation about classpath, environment, JMS and JDBC resources configuration within the application server can be found in the TAFJ-AS specific application server documentation: TAFJ-AS jBossInstall, TAFJ-AS WebsphereInstall, TAFJ-AS WebLogicInstall.

Please refer to these documents for more information.

TAFJ-AS T24 component

T24 Java deployment architecture is the following:



Four types of channels are used in T24 java deployment:

1. OFS Request/Reply based on JMS request/reply queues.
   This channel is configured/instantiated by default for OFS, Browser, TWS, ARCMOB, TCIB, AML and SEAT.
2. CALLAT Request/Reply based on JMS request/reply queues. This channel will initialize a TAFJ Session with JF.INITIALIZE and do a CALL @ with parameters.
3. Phantom based on request JMS Queue. This channel is used to manage T24 phantom feature (start a thread in background).
4. TEC Events. T24 TEC is able to publish TEC Events on TEC JMS Topics. This JMS topic could be consumed by external third party software.

TAFJJEE_EAR.ear file contains four components:

1. TAFJJEE_MDB.jar configures all default MDB to operate T24 java deployment, OFS, TWS, TCIB, ARCMOB, AML, SEAT, CALLAT, Phantom MDB

2. TAFJJEE_EJB.jar configures all defaults EJB to operate T24 java deployment, OFS, TWS, TCIB, ARCMOB, AML, SEAT, CALLAT EJB.

   Note that there is no Phantom EJB.

3. TAFJJEE_WAR_TAFJ.war provides set of servlet and utilities

   – Diagnostic tools : tDiag / tShow

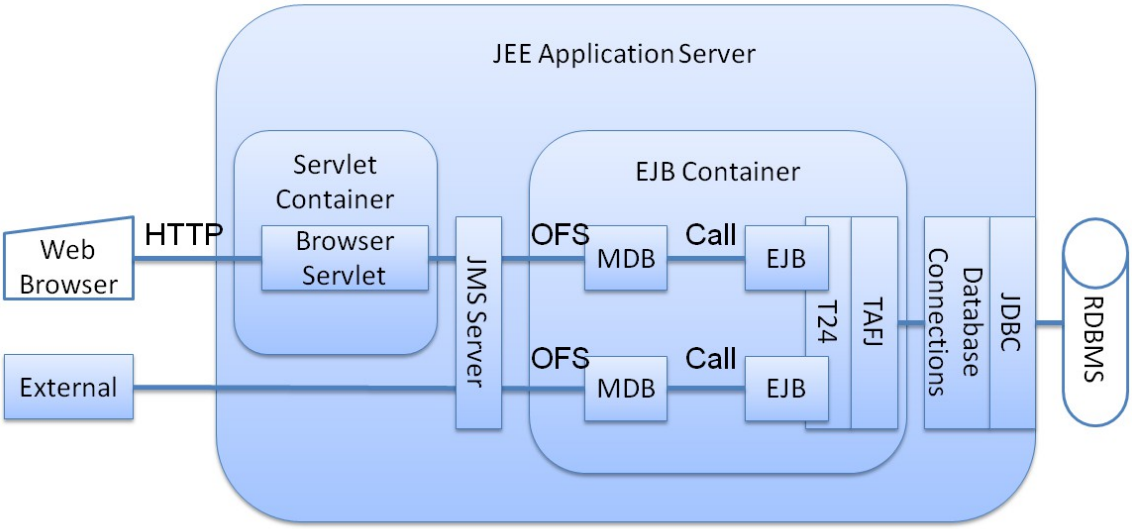
   – Execution tools: send message in the EXEC JMS Queue, Entry points documentation, DBTools in latest TAFJ version

   – Troubleshooting and monitoring tools

4. TAFJServices.war is an Axis based archive to provide a webservice access to OFS and CALL_AT functionalities to validate the T24 java deployment without any client application installation.

JMS Request/Reply channel

Component architecture:



For JMS Request/Reply channels, TAFJ sets up MDBs to listen on JMS Queues and consume JMS messages.

The MDB is calling an EJB responsible to process the T24 request.

EJB response is passed to the MDB which will send a response back to a reply queue.

The default channels are:

1. OFS Channel
2. BROWSER Channel
3. TWS Channel
4. ARCMOB Channel
5. TCIB Channels (TCIB, WEALTH, CORP)
6. AML Channel
7. CALLAT Channel
8. SEAT Channel (Regression purpose)

MDB and EJB mapping

Before describing how MDB and EJB can be configured, it is important to understand the configuration files involved in the setup.

MDB/EJB Configuration is done through standard JEE files **ejb-jar.xml** and application server specific file **jboss-ejb3.xml**, **ibm-ejb-jar-bnd.xml, weblogic-ejb-jar.xml** respectively for jBoss, IBM, Weblogic.

For each MDB/EJB, you could configure mapping between an application resource name (logical name) and a specific application server name (physical name – jndi name).

For example, the logical name for the T24 datasource is `jdbc/t24DataSource` whatever the deployment is. The physical name or JNDI name is specific to each application server provider.

i.e. : ejb-jar.xml declares the logical name: `jdbc/t24DataSource` which will be mapped to the JNDI name `java:/jdbc/t24DS` in jboss-ejb3.xml for JBoss 6EAP.

These files are packaged in the META-INF folder of TAFJJEE_MDB.jar and TAFJJEE_EJB.jar.

MDB Components configuration

MDB components are defined in TAFJJEE_EAR.ear\TAFJJEE_MDB.jar.

A request / response MDB is using the following **mandatory resources**:

- A connection factory to connect to the JMS provider.
- A Destination Queue to receive the incoming message.
- A Reply queue to send T24 response.
- An EJB to handle the request and do the T24 processing.

A request / response MDB is using the following **optional properties**:

- Message processing:
    - An optional formatter to transform the incoming message (i.e. OFSML formatter).
    - An optional principal to be passed to T24.
- Response processing:
    - A message response delivery mode : PERSISTENT - NON PERSISTENT – default is NON PERSISTENT for performance reason.
    - An option to send the response to the destination defined in the message property `getJMSReplyTo()` - disabled by defaut.
    - An option to use the incoming message message ID as the correlation ID for the reply – disabled by defaut.
- Error handling:
    - An option to don't send to T24 redelivered message.
    - An option to ignore poison message depending on the number of message re-delivery attempt - disabled by default as it should be a feature of the JMS provider.
    - An option to discard the MDB in case of JMS Exception.

TAFJJEE_MDB.jar/META-INF/ejb-jar.xml

The table below presents the main configuration part for a given MDB called `OFSTransactedMDB`, it could be found within the section `<enterprise-beans>` of the **ejb-jar.xml**

It allows defining the mandatory resources presented above and doesn't make use of optional properties.

Property value marked editable could be changed in case of definition of a new channel or specific need.

Xml path	Example value	Comment
display-name	Transacted Listener MDB for OFS	Self-explanation about the MDB goal - editable
ejb-name	OFSTransactedMDB	Application server name reference. editable
ejb-class	com.temenos.tafj.mdb.TransactedMDB	MDB class implementation.
messaging-type	javax.jms.MessageListener	JMS property
transaction-type	Container	Transaction management mode. Transaction is handled by the application server container.
message-destination-type	javax.jms.Queue	JMS property
Property configuration		
env-entry\useLocal	**false**	**Flag to call local or remote mode EJB. Local has better performance, not supported in Weblogic.** See comment in the file.
com.temenos.tafj.mdb.TransactedMDB/jndiNameLocal	**java:comp/env/ejb/OFSProcessingBeanLocal**	Define related EJB JNDI name local - name space may vary depending on appserver version. See comment in the file.
com.temenos.tafj.mdb.TransactedMDB/jndiNameRemote	**java:comp/env/ejb/OFSProcessingBeanRemote**	Define related EJB JNDI name remote - name space may vary depending on appserver version. See comment in the file.
EJB reference configuration		
ejb-ref\ejb-ref-name\	ejb/OFSProcessingBeanRemote	EJB Remote interface reference name use in the MDB which has to be mapped in Application

		Server specific configuration file
ejb-ref\ejb-ref-type	Session	EJB property
ejb-ref\remote	com.temenos.tafj.sb.OFS ProcessingBeanRemote	Remote interface class
ejb-ref\ejb-link	OFSProcessingBean	Specify which EJB is linked editable
ejb-local-ref\ejb-ref-name\	ejb/OFSProcessingBeanLocal	EJB Local interface reference name used in the MDB which has to be mapped in Application Server specific configuration file
ejb-local-ref\ejb-ref-type	Session	EJB property
ejb-local-ref\local	com.temenos.tafj.sb.OFS ProcessingBeanLocal	Local interface class
ejb-local-ref\ejb-link	OFSProcessingBean	Specify which EJB is linked editable
Resources reference configuration		
resource-ref\ res-ref-name	jms/TAFJQueueConnectionFactory	Configure the JMS Connection factory for the reply Queue. Needs to be mapped in Application server configuration file.
resource-ref\ res-type	Javax.jms.ConnectionFactory	JMS property
resource-ref\ res-auth	Container	JMS property
Message destination reference		
message-destination-ref\ message-destination-ref-name	jms/ReplyQueue	MDB resource name for reply queue injection. Needs to be mapped in Application server configuration file.
message-destination-ref\message-destination-type	Javax.jms.Queue	JMS property
message-destination-ref\message-destination-usage	Produces	JMS property
message-destination-ref \ message-destination--link	jms/OFSReplyQueue	Weblogic only - MDB resource name for reply queue injection. Needs to be mapped in Application server configuration file. editable

Optional properties

The following properties could be added within an ejb-jar. `<message-driven>` section to refine a specific MDB setup. See table below for detailed explanation about them.

```
<!-- ADDITIONAL MESSAGE PARAMETERS - formatter - principal -->

<env-entry>
       <env-entry-name>com.temenos.tafj.mdb.TransactedMDB/defaultFormatter</env-entry-name>
       <env-entry-type>java.lang.String</env-entry-type>
       <env-entry-value>formatter class</env-entry-value>
</env-entry>

<env-entry>
       <env-entry-name>com.temenos.tafj.mdb.TransactedMDB/defaultPrincipal</env-entry-name>
       <env-entry-type>java.lang.String</env-entry-type>
       <env-entry-value>INPUTT/123456</env-entry-value>
</env-entry>

<!-- RESPONSE PARAMETERS -->

<env-entry>
       <description>Define response delivery mode (NON_PERSISTENT = 1 PERSISTENT = 2) - de-
       fault is non-persistent for performance reason</description>
       <env-entry-name>com.temenos.tafj.mdb.TransactedMDB/responseDeliveryMode</env-en-
try-name>
       <env-entry-type>java.lang.Integer</env-entry-type>
       <env-entry-value>1</env-entry-value>
</env-entry>

<env-entry>
       <description>Define whether message field JMSReplyTo should be used for response (tem-
       porary queue) - default is false</description>
       <env-entry-name>com.temenos.tafj.mdb.TransactedMDB/sendToJmsReplyTo</env-entry-name>
       <env-entry-type>java.lang.Boolean</env-entry-type>
       <env-entry-value>false</env-entry-value>
</env-entry>

<env-entry>
       <description>Define whether message ID should be used to fill-up response correlation
       id - default is false</description>
       <env-entry-name>com.temenos.tafj.mdb.TransactedMDB/messageIdAsCorrelationID</env-en-
try-name>
       <env-entry-type>java.lang.Boolean</env-entry-type>
       <env-entry-value>false</env-entry-value>
</env-entry>

<!-- ERROR HANDLING PARAMETERS -->

<env-entry>
       <description> Define whether re-delivered message should be ignored - default is true
       a re-delivered message is not sent to T24 - when set to false re-delivered message are
       sent to T24</description>
       <env-entry-name>com.temenos.tafj.mdb.TransactedMDB/ignoreRedeliveredMessage</env-en-
try-name>
       <env-entry-type>java.lang.Boolean</env-entry-type>
       <env-entry-value>false</env-entry-value>
</env-entry>

<env-entry>
       <description>Define the number of re-delivery attempt in case of failure - -1 means
       disabled by default the JMS provider handles the maximum number of JMS delivery at-
       tempt - 0 means no re-delivery attempt</description>
       <env-entry-name>com.temenos.tafj.mdb.TransactedMDB/maxJmsRedeliveryAttempt</env-entry-
name>
       <env-entry-type>java.lang.Integer</env-entry-type>
```

```
        <env-entry-value>1</env-entry-value>
</env-entry>

<env-entry>
        <description>Define whether setRollbackOnly is called or System exception thrown in
        case of JMS error - default is true MDB is not discarded in case of JMSException</de-
        scription>
        <env-entry-name>com.temenos.tafj.mdb.TransactedMDB/recoverFromJmsFailure</env-en-
try-name>
        <env-entry-type>java.lang.Boolean</env-entry-type>
        <env-entry-value>true</env-entry-value>
</env-entry>
```

Parameter	Description	Default value

`defaultFormatter`	Message formatting parameter.
	String parameter.
	Formatter to be applied on the message to transform it.
	An OFSML formatter is available in the ear file.
	To apply a specific formatter, simply add in the value field the fully qualified class name of the formatter and make it available in the application server classpath.
		None
		
		To use OFSML formatter apply:
		`<env-entry-value>OFSML</env-entry-value>`
		
		To use specific formatter
		`<env-entry-value>com.temenos.formatter.MyCustomFormatter</env-entry-value>`

`defaultPrincipal`	Message processing parameter. String parameter. You can define a default principal for the associated message channel.	None To define a principal `<env-entry-value>INPUTT/123456</env-entry-value>`
`responseDeliveryMode`	Response processing parameter. Integer parameter. Defines whether response message is delivered as a PERSISTENT or NON-PERSISTENT message. Closed set of values : 1- Means NON-PERSISTENT 2- Means PERSISTENT	1
`sendToJmsReplyTo`	Response processing parameter. Boolean parameter. Defines whether response message is sent to the destination defined through message property getJMSReplyTo Closed set of values : true : use the message property false : use the jmsReplyQueue defined at MDB level	false

`messageIdAsCorrelationID`	Response processing parameter. Boolean parameter. Defines whether response message correlation ID is set with the original message ID. Closed set of values : true : use the message ID false: use the message correlation ID	false
`ignoreRedeliveredMessage`	Error handling parameter. Boolean parameter. In case of message redelivery this flag could be enabled to avoid message re-processing in T24. MDB will receive the message but not pass it to EJB. Could be used in case of Bean managed EJB where there will be a transaction demarcation between MDB and EJB. Closed set of values : false: redelivered message are sent to EJB/T24 true: message are ignored and a response is sent to the client that this is a redelivered message.	true
`maxJmsRedeliveryAttempt`	Error handling parameter. Integer parameter. In case of message redelivery, if this value is greater or equal than 0, the message property `JMSXDeliveryCount` is checked against this value to protect against poison message. This feature is disabled by default as it must be already covered by the JMS provider. When using this feature, message is simply ignored and information is logged but there is no dead letter queue associated.	-1 means disabled To define no re-delivery set this value to 0. To define 1 re-delivery only set this value to 1.
`recoverFromJmsFailure`	Error handling parameter. Boolean parameter. In case of JMS exception at MDB level this flag when set to true will call setRollbackOnly or throw an EJB exception to discard the MDB when set to false.	true

TAFJJEE_MDB.jar/META-INF/jboss-ejb3.xml

The table below presents the corresponding JBOSS EAP6 configuration part for the MDB `OFSTransactedMDB` presented above. It could be found within the section `<enterprise-beans>` of the **jboss-ejb3.xml.**

Xml path	Example value	Comment
ejb-name	OFSTransactedMDB	Has to match the ejb-name of ejb-jar.xml editable
activation-config-property-name	destination	JMS property
activation-config-property-value	java:/queue/t24OFSQueue	Name of the queue, in jboss jndi tree, the MDB is listening on editable
ejb-ref\ejb-ref-name	ejb/OFSProcessingBeanRemote	Has to match the Ejb-ref\ejb-ref-name\ of ejb-jar.xml
ejb-ref\jndi-name	java:global/TAFJJEE_EAR/TAFJJEE_EJB/OFSProcessingBean!com.temenos.tafj.sb.OFSProcessingBeanRemote	OFS EJB remote JNDI name in jBOSS application server. Configured by TAFJJEE_EJB component.
ejb-ref\ejb-ref-name	ejb/OFSProcessingBeanLocal	Has to match to the Ejb-ref\ejb-ref-name\ of ejb-jar.xml
ejb-ref\jndi-name	java:app/TAFJJEE_EJB/OFSProcessingBean!com.temenos.tafj.sb.OFSProcessingBeanLocal	OFS EJB local JNDI name in jBOSS application server. Configured by TAFJJEE_EJB component.
Resources reference configuration		
resource-ref\ res-ref-name	jms/TAFJQueueConnectionFactory	Configure the JMS Connection factory for the response reply Queue.
resource-ref\ res-type	Javax.jms.ConnectionFactory	JMS property
resource-ref\ jndi-name	ConnectionFactory	Match the default JMS resource Connection factory in JBOSS editable
Message destination reference		
message-destination-ref\ message-destination-ref-name	jms/ReplyQueue	Mapping with the reply queue.
message-destination-ref\jndi-name	java:/queue/t24OFSReplyQueue	Name of the reply queue in jboss jndi tree editable

TAFJJEE_MDB.jar/META-INF/ibm-ejb-jar-bnd.xml

The table below presents the corresponding Websphere 8.5 configuration part for the MDB `OFSTransactedMDB` presented above. It could be found within the main section of the **ibm-ejb-jar-bnd.xml**

Xml path	Example value	Comment
message-driven name	OFSTransactedMDB	Has to match the ejb-name of ejb-jar.xml editable
ejb-ref\ejb-ref-name	ejb/OFSProcessingBeanRemote	Has to match the Ejb-ref\ejb-ref-name\ of ejb-jar.xml
ejb-ref\binding-name	ejb/TAFJJEE_EAR/TAFJJEE_EJB.jar/OFSProcessingBean#com.temenos.tafj.sb.OFSProcessingBeanRemote	OFS EJB remote JNDI name in WAS application server. Configured by TAFJJEE_EJB component.
ejb-ref\ejb-ref-name	ejb/OFSProcessingBeanLocal	Has to match to the Ejb-ref\ejb-ref-name\ of ejb-jar.xml
ejb-ref\binding-name	ejblocal:TAFJJEE_EAR/TAFJJEE_EJB.jar/OFSProcessingBean#com.temenos.tafj.sb.OFSProcessingBeanLocal	OFS EJB local JNDI name in WAS application server. Configured by TAFJJEE_EJB component.
Jca adapter		
activation-spec-binding-name	jms/t24OFSMessageMDB	Websphere internal activation spec. name editable
destination-binding-name	jms/t24OFSQueue	Name of the queue, in websphere jndi tree, the MDB is listening on editable
Resources reference configuration		
resource-ref\ res-ref-name	jms/TAFJQueueConnectionFactory	Configure the JMS Connection factory for the response reply Queue.
resource-ref\ jndi-name	ConnectionFactory	Match the default JMS resource Connection factory in JBOSS editable
Message destination reference		
name	jms/ReplyQueue	Mapping with the reply queue.
binding-name	jms/t24OFSReplyQueue	Name of the reply queue in websphere jndi tree editable

TAFJJEE_MDB.jar/META-INF/weblogic-ejb-jar.xml

The table below presents the corresponding Weblogic 12.1.X configuration part for the MDB `OFSTransactedMDB` presented above. It could be found within the `<wls:weblogic-ejb-jar>` section of the **weblogic-ejb-jar.xml.**

Xml path	Example value	Comment
Ejb-name	OFSTransactedMDB	Has to match the ejb-name of ejb-jar.xml editable
Destination-jndi-name	jms/t24OFSQueue	Name of the queue, in weblogic jndi tree, the MDB is listening on editable
Connection-factory-jndi-name	jms/ConnectionFactory	Name of the connection factory, in weblogic jndi tree. editable
ejb-ref\ejb-ref-name	ejb/OFSProcessingBeanRemote	Has to match the Ejb-ref\ejb-ref-name\ of ejb-jar.xml
ejb-ref\jndi-name	ejb/OFSProcessingBeanRemote	OFS EJB remote JNDI name in weblogic application server. Configured by TAFJJEE_EJB component.
ejb-ref\ejb-ref-name	ejb/OFSProcessingBeanLocal	Has to match to the Ejb-ref\ejb-ref-name\ of ejb-jar.xml
ejb-ref\jndi-name	ejb/OFSProcessingBeanLocal	OFS EJB local JNDI name in weblogic application server. Configured by TAFJJEE_EJB component.
Resources reference configuration		
res-ref-name	jms/TAFJQueueConnectionFactory	Configure the JMS Connection factory for the response reply Queue.
jndi-name	jms/ConnectionFactory	Match the default JMS resource Connection factory in weblogic editable
Message destination descriptor		
Message-destination-name	jms/OFSReplyQueue	Has to match the reply queue name defined in the message-destination-link in ejb-jar.xml
Destination-jndi-name	jms/t24OFSReplyQueue	Name of the reply queue in weblogic jndi tree

		editable

EJB Components configuration

EJB components are defined in TAFJJEE_EAR.ear\TAFJJEE_EJB.jar.

There is two different EJB types, OFS EJB and CALLAT EJB.

An OFS EJB is using the following **mandatory resources**:

- Data-sources:
    - T24 data source
    - T24 Locking data source when using JDBC locking or database locking.
    - T24 read only data source when using read only data model
- A JMS connection factory for TEC events
- A JMS topic for TEC events

An OFS EJB is using the following **properties**:

- Request processing:
    - The OFS source
    - The OFS timeout
    - Whether the session should be clean up between two invocation
- Request monitoring optional
    - The request type for monitoring classification purpose.

TAFJJEE_EJB.jar/META-INF/ejb-jar.xml

The sample below presents the main configuration part for a given EJB called `OFSProcessingBean`, it could be found within the section `<enterprise-beans>` of the **ejb-jar.xml,** this EJB is associated to the `OFSTransactedMDB` presented above but could also be invoked by direct EJB invocation.

It allows defining the mandatory resources presented above.

Xml path	Example value	Comment
ejb-name	OFSProcessingBean	EJB name. editable
business-local	com.temenos.tafj.sb.OFSProcessingBeanLocal	EJB TAFJ Local interface.
business-remote	com.temenos.tafj.sb.OFSProcessingBeanRemote	EJB TAFJ Remote interface.
ejb-class	com.temenos.tafj.sb.OFSProcessingBean	EJB TAFJ Remote interface.

session-type	Stateless	EJB property
transaction-type	Bean	Transaction management mode – The bean handle the transaction.
Property configuration		
env-entry\env-entry-name	com.temenos.tafj.sb.OFSProcessingBean/ofsSource	Property name. Used to configure OFS source.
env-entry\env-entry-type	**GCS**	**OFS Source value** editable
env-entry\env-entry-name	com.temenos.tafj.sb.OFSProcessingBean/ofsTimeout	Property name. Configure OFS timeout.
env-entry\env-entry-type	**60**	**OFS timeout in seconds** editable
Resources reference configuration		
resource-ref\ res-ref-name	jdbc/t24DataSource	Configure the Application server JDBC resource for T24 database. Needs to be mapped in Application server configuration file.
resource-ref\ res-ref-name	jdbc/t24LockingDataSource	Configure the Application server JDBC resource for T24 locking in database. Needs to be mapped in Application server configuration file.
resource-ref\ res-ref-name	jdbc/t24RODataSource	Configure the Application server JDBC resource for T24 read only database. Needs to be mapped in Application server configuration file.
resource-ref\ res-ref-name	jms/TopicConnectionFactory	Configure the JMS TOPIC factory for TEC events publishing. Needs to be mapped in Application server configuration file.
resource-ref\ res-ref-name	jms/tecEventsTopic	Configure the JMS TOPIC for TEC events publishing. Needs to be mapped in Application server configuration file.

Mandatory properties

As described above there is two mandatory properties which should be defined for each channel:

```
<env-entry>
        <description>OFS OFS Source</description>
        <env-entry-name>com.temenos.tafj.sb.OFSProcessingBean/ofsSource</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>GCS</env-entry-value>
</env-entry>

<env-entry>
        <env-entry-name>com.temenos.tafj.sb.OFSProcessingBean/ofsTimeout</env-entry-name>
        <env-entry-type>java.lang.Integer</env-entry-type>
        <env-entry-value>60</env-entry-value>
</env-entry>
```

Optional properties

The following properties could be added within an ejb-jar. `<session>` section to refine a specific EJB setup. See table below for detailed explanation about them.

```
<env-entry>
        <env-entry-name>com.temenos.tafj.sb.OFSProcessingBean/resetSession</env-entry-name>
        <env-entry-type>java.lang.Boolean</env-entry-type>
        <env-entry-value>true</env-entry-value>
</env-entry>

<!--  To be used in conjunction with MonitoringClassifierInterceptor to get a request classi-
fication per EJB -->
<env-entry>
        <description>Request type for monitoring classification</description>
        <env-entry-name>com.temenos.tafj.sb.OFSProcessingBean/requestType</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>SOME-CLASSIFIER</env-entry-value>
</env-entry>
```

Parameter	Description	Default value
`ofsSource`	Request processing parameter. String parameter. Has to match an existing T24 OFS source.	None – needs to be setup
`ofsTimeout`	Request processing parameter. Integer parameter. Time in seconds before request expiration at T24 level.	0 – means no timeout
`resetSession`	Request processing parameter. Boolean parameter. When set to true the T24 session gets cleaned between each invocation. Impact the performance, only used for TWS channel.	false
`requestType`	Request monitoring parameter.	None

	String parameter. The value associated to this parameter will be used as a classifier in TAFJEE monitoring tool. When not set the OFS source is used.	

TAFJJEE_EJB.jar/META-INF/jboss-ejb3.xml

The table below presents the corresponding JBOSS EAP6 configuration part for the EJB `OFSProcessingBean` presented above. It could be found within the section `<enterprise-beans>` of the **jboss-ejb3.xml**

TAG/Property name	Example value	Comment
ejb-name	OFSProcessingBean	Has to match the ejb-name of ejb-jar.xml editable
resource-ref\ res-ref-name	jdbc/t24DataSource	Maps the res-ref-name from ejb-jar.xml
resource-ref\jndi-name	java:/jdbc/t24DS	Match the T24 JDBC resource in JBOSS application server. editable
resource-ref\ res-ref-name	jdbc/t24LockingDataSource	Maps the res-ref-name from ejb-jar.xml
resource-ref\jndi-name	java:/jdbc/t24LockingDS	Match the T24 Locking JDBC resource in JBOSS application server. editable
resource-ref\ res-ref-name	jdbc/t24RODataSource	Maps the res-ref-name from ejb-jar.xml
resource-ref\jndi-name	java:/jdbc/t24RODS	Match the T24 read only JDBC resource in JBOSS application server. editable
resource-ref\ res-ref-name	jms/TopicConnectionFactory	Maps the res-ref-name from ejb-jar.xml
resource-ref\jndi-name	Java:/ConnectionFactory	Match the default JMS connection factory in jboss jndi tree. editable
resource-ref\ res-ref-name	jms/tecEventsTopic	Maps the res-ref-name from ejb-jar.xml
resource-ref\jndi-name	Java:/topic/tecEvents	Match the T24 JMS topic in JBOSS jndi tree. editable

TAFJJEE_EJB.jar/META-INF/ibm-ejb-jar-bnd.xml

The table below presents the corresponding Websphere 8.5 configuration part for the EJB `OFSPRocessingBean` presented above. It could be found within the main section of the **ibm-ejb-jar-bnd.xml**

Xml path	Example value	Comment
Session name	OFSProcessingBean	Has to match the ejb-name of ejb-jar.xml editable
Resources reference configuration		
resource-ref\ res-ref-name	jdbc/t24DataSource	Maps the res-ref-name from ejb-jar.xml
resource-ref\ binding-name	jdbc/t24DataSource	Match the T24 JDBC resource in WAS application server. editable
resource-ref\ res-ref-name	jdbc/t24LockingDataSource	Maps the res-ref-name from ejb-jar.xml
resource-ref\ binding-name	jdbc/t24LockingDataSource	Match the T24 Locking JDBC resource in WAS application server. editable
resource-ref\ res-ref-name	jdbc/t24RODataSource	Maps the res-ref-name from ejb-jar.xml
resource-ref\ binding-name	jdbc/t24RODataSource	Match the T24 read only JDBC resource in WAS application server. editable
resource-ref\ res-ref-name	jms/TopicConnectionfactory	Maps the res-ref-name from ejb-jar.xml
resource-ref\ binding-name	jms/t24ConnectionFactory	Match the connection factory in WAS jndi tree. editable
resource-ref\ res-ref-name	jms/tecEventsTopic	Maps the res-ref-name from ejb-jar.xml
resource-ref\ binding-name	jms/tecEventsTopic	Match the T24 JMS topic in WAS jndi tree. editable

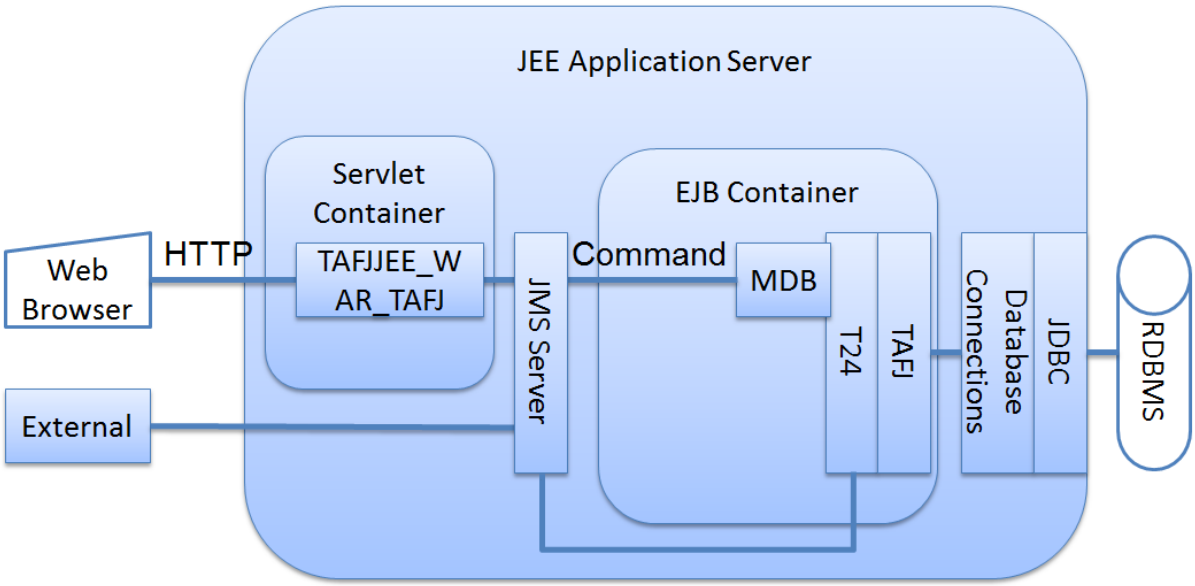
TAFJJEE_EJB.jar/META-INF/weblogic-ejb-jar.xml

The table below presents the corresponding Weblogic 12.1.X configuration part for the EJB `OFSProcessingBean` presented above. It could be found within the `<wls:weblogic-ejb-jar >` section of the **weblogic-ejb-jar.xml**

Xml path	Example value	Comment
Ejb-name	OFSProcessingBean	Has to match the ejb-name of ejb-jar.xml editable
business-interface-jndi-name-map/business-remote	com.temenos.tafj.sb.OFSProcessingBeanRemote	EJB remote interface
business-interface-jndi-name-map/jndi-name	ejb/OFSProcessingBeanRemote	EJB jndi name for remote invocation
Resources reference configuration		
res-ref-name	jdbc/t24DataSource	Maps the res-ref-name from ejb-jar.xml
jndi-name	jdbc/t24DS	Match the T24 JDBC resource in weblogic application server. editable
res-ref-name	jdbc/t24LockingDataSource	Maps the res-ref-name from ejb-jar.xml
jndi-name	jdbc/t24LockingDS	Match the T24 locking JDBC resource in weblogic application server. editable
res-ref-name	jdbc/t24RODataSource	Maps the res-ref-name from ejb-jar.xml
jndi-name	jdbc/t24RODS	Match the T24 read only JDBC resource in weblogic application server. editable
res-ref-name	jms/TopicConnectionFactory	Maps the res-ref-name from ejb-jar.xml
jndi-name	jms/ConnectionFactory	Match the connection factory in weblogic jndi tree. editable

Phantom channel

Component architecture:



For the JMS Phantom channel, TAFJ sets up a MDB to listen on JMS EXEC Queue and consume JMS messages in **Bean managed mode**. XA transactions are not possible in this architecture because T24 needs to republish commands in the EXEC Queue. Phantom MDB will call directly T24 code.

For example, to start TSM, You could send a “START.TSM” command message in the EXEC Queue. The Phantom MDB will consume the command and launch a thread which will execute the TSM program. If a COB record exists in TSA.SERVICE with field START set to true, TSM will send command “tSA 1” in the EXEC Queue. Phantom MDB will consume the command and launch a thread to execute the COB.

Phantom MDB EJB-JAR Component configuration

Like other request / reply MDBs, Phantom MDB is configured in TAFJJEE_EAR.ear\TAFJJEE_MDB.jar META-INF**ejb-jar.xml** within the `<enterprise-beans>` section.

Xml path	Example value	Comment
ejb-name	TAFJPhantomListenerMDB	MDB name
ejb-class	com.temenos.tafj.mdb.PhantomListener	TAFJ implementation class
transaction-type	Bean	Transaction management type, not editable.
mapped-name	jms/t24EXECQueue	Weblogic-only defines the physical queue the MDB is listening on.
messaging-type	javax.jms.MessageListener	JMS property
message-destination-type	javax.jms.Queue	JMSProperty
activation-config-property-name	acknowledgeMode	JMS property
activation-config-property-name	Auto-acknowledge	JMS property
Resources reference configuration		
resource-ref\res-ref-name	jms/TAFJQueueConnectionFactory	Configure the JMS Connection factory for the response reply Queue. Needs to be mapped in Application server configuration file.
resource-ref\res-ref-name	jdbc/t24DataSource	Configure the Application server JDBC resource for T24 database. Needs to be mapped in Application server configuration file.
resource-ref\res-ref-name	jdbc/t24LockingDataSource	Configure the Application server JDBC resource for T24 locking in database. Needs to be mapped in Application server configuration file.
resource-ref\res-ref-name	jdbc/t24RODataSource	Configure the Application server JDBC resource for T24 read only database. Needs to be mapped in Application server configuration file.
resource-ref\res-ref-name	jms/TopicConnectionFactory	Configure the JMS TOPIC factory for TEC events publishing.

		Needs to be mapped in Application server configuration file.
resource-ref\ res-ref-name	jms/tecEventsTopic	Configure the JMS TOPIC for TEC events publishing. Needs to be mapped in Application server configuration file.
resource-ref\ res-ref-name	jms/t24EXECQueue	Configure the JMS EXEC Queue to send new command. Needs to be mapped in Application server configuration file.

Optional properties

The following property could be added within `<message-driven>` section to refine PhantomListener MDB setup.

```
<!-- ERROR HANDLING PARAMETERS -->

<env-entry>
        <description>Define the number of re-delivery attempt in case of failure - -1 means
        disabled by default the JMS provider handles the maximum number of JMS delivery at-
        tempt - 0 means no re-delivery attempt</description>
        <env-entry-name>com.temenos.tafj.mdb.TransactedMDB/maxJmsRedeliveryAttempt</env-entry-
name>
        <env-entry-type>java.lang.Integer</env-entry-type>
        <env-entry-value>0</env-entry-value>
</env-entry>
```

Parameter	Description	Default value
`maxJmsRedeliveryAttempt`	Error handling parameter. Integer parameter. In case of message redelivery, if this value is greater or equal than 0, the message property `JMSXDeliveryCount` is checked against this value to protect against poison message. By default re-delivered message are ignored.	0 means do re-delivery in case of failure

PhantomListener MDB makes use of same resources than OFS EJB: T24 data sources and TEC events topic and connection factory.

Please refer to EJB configuration section about jboss, websphere and weblogic for application server mapping explanation.

It's important to note that Phantom MDB is not only using EXECQueue to receive message but also to send commands.

TAFJJEE_WAR_TAFJ - Servlet Component

This is a web module which contains following servlets:

- ExecuteServlet: to send message in the EXEC JMS Queue, for example start a COB with START.TSM command as seen above.
- tDiagServlet: to get information about the installation
- tShowServlet: to get information about a T24 routine if present in the classpath
- LoggerServlet: to change dynamically log level and access log files
- ComoServlet: to access como content with some filtering capabilities
- DBToolsServlet: to execute DBTools command through the application server data sources

It also contains some documentation about interaction capabilities with TAFJEE from external application or standalone client and a technical monitor.

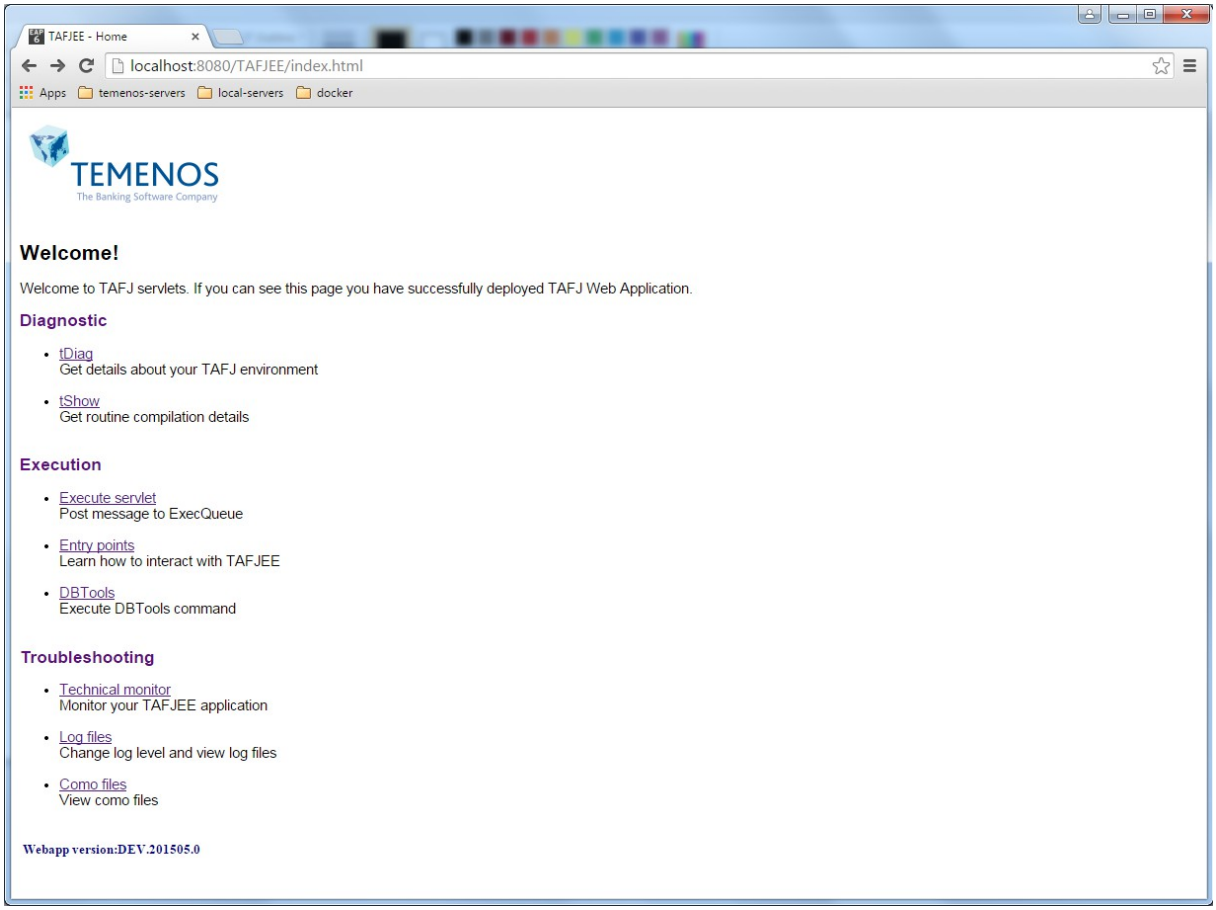
TAFJJEE_WAR_TAFJ is configured in WEB-INF**web.xml** by following properties under xml path **web-app**:

Xml path	Example value	Comment
servlet\servlet-name	ExecuteServlet	Servlet name
servlet\servlet-class	com.temenos.tafj.jee.war.ExecuteServlet	class name
servlet-mapping\servlet-name	ExecuteServlet	Map “servlet mapping” with the servlet
servlet-mapping\url-pattern	**/Execute**	**Mapping url**
servlet\servlet-name	tShowServlet	Servlet name
servlet\servlet-class	com.temenos.tafj.jee.war.TShowServlet	class name
servlet-mapping\servlet-name	tShowServlet	Map “servlet mapping” with the servlet
servlet-mapping\url-pattern	**/tShow**	**Mapping url**
servlet\servlet-name	tDiagServlet	Servlet name
servlet\servlet-class	com.temenos.tafj.jee.war.TDiagServlet	class name

servlet-mapping\servlet-name	tDiagServlet	Map “servlet mapping” with the servlet
servlet-mapping\url-pattern	**/tDiag**	**Mapping url**
servlet\servlet-name	LoggerServlet	Servlet name
servlet\servlet-class	com.temenos.tafj.jee.war.LoggerServlet	class name
servlet-mapping\servlet-name	LoggerServlet	Map “servlet mapping” with the servlet
servlet-mapping\url-pattern	**/logger**	**Mapping url**
servlet\servlet-name	ComoServlet	Servlet name
servlet\servlet-class	com.temenos.tafj.jee.war.ComoServlet	class name
servlet-mapping\servlet-name	ComoServlet	Map “servlet mapping” with the servlet
servlet-mapping\url-pattern	**/comp**	**Mapping url**
servlet\servlet-name	DBToolsServlet	Servlet name
servlet\servlet-class	com.temenos.tafj.jee.war.DBToolsServlet	class name
servlet-mapping\servlet-name	DBToolsServlet	Map “servlet mapping” with the servlet
servlet-mapping\url-pattern	**/DBTools**	**Mapping url**
resource-ref\res-ref-name	jms/TAFJQueueConnectionFactory	Configure the JMS Connection factory for the response reply Queue. Needs to be mapped in Application server configuration file.
resource-ref\res-ref-name	jms/T24EXECQueue	Configure the JMS EXEC Queue to send new command. Needs to be mapped in Application server configuration file.

Verify installation: TAFJEE main page

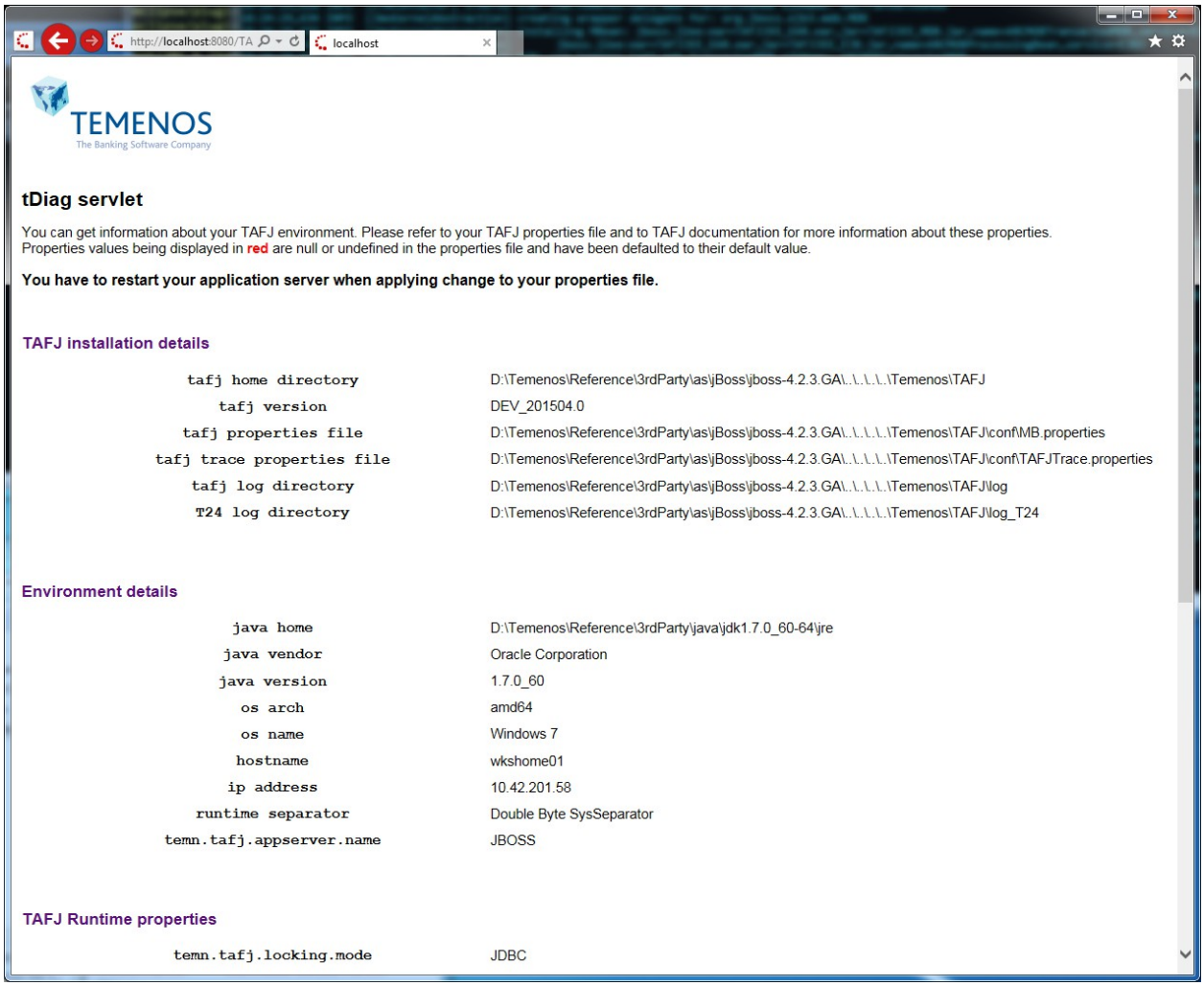
You could browse http://HOSTNAME:PORT/TAFJEE/ which will give you a response like:



Getting TAFJ installation details: tDiagServlet

http://localhost:8080/TAFJEE/tDiag

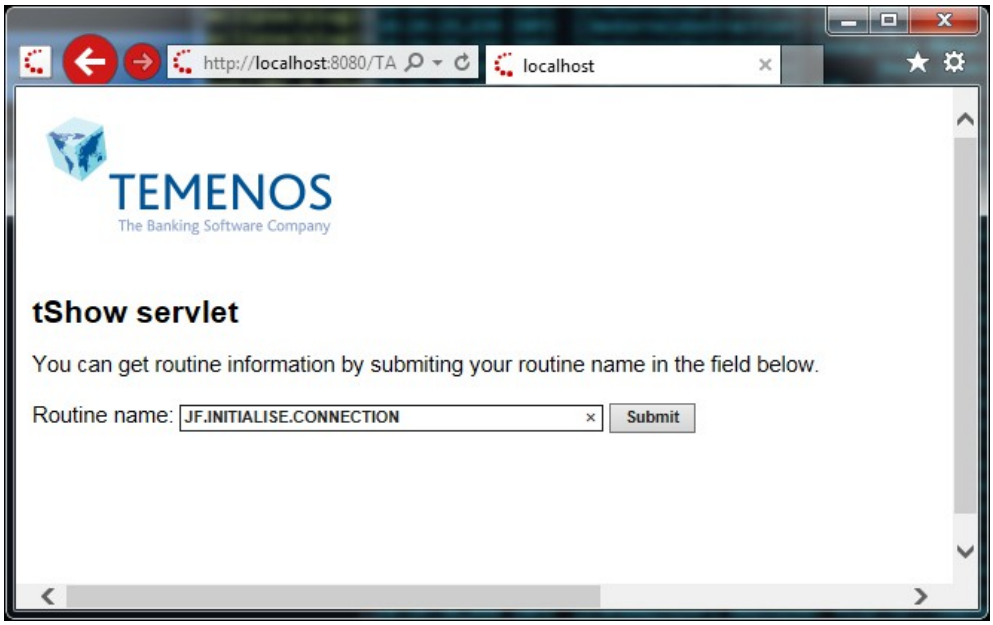
i.e.

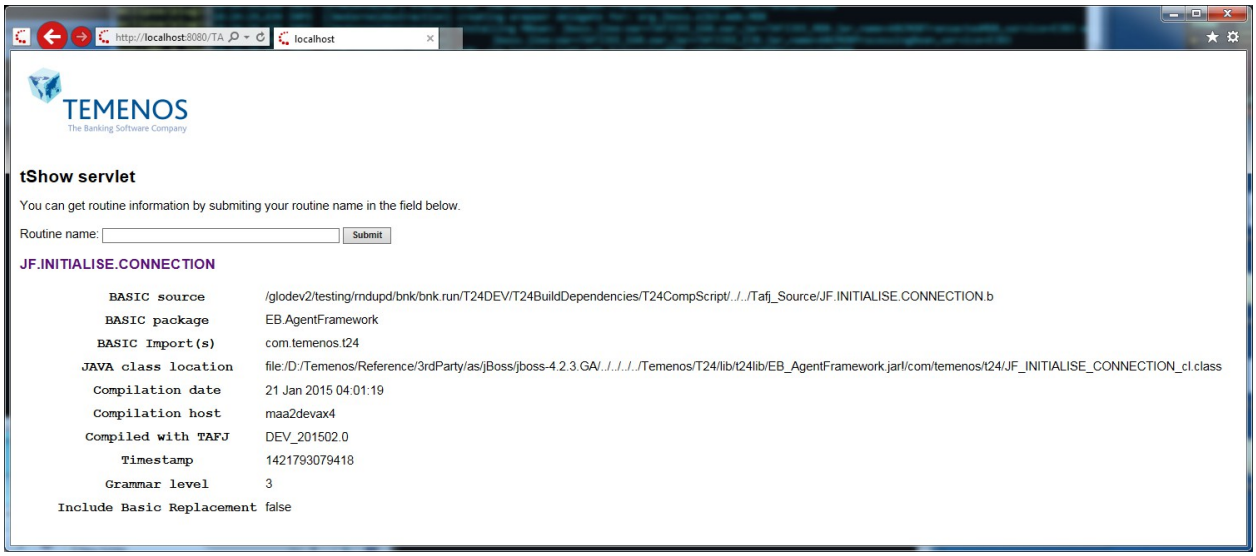


Getting routine compilation details: tShowServlet

http://localhost:8080/TAFJEE/tShow

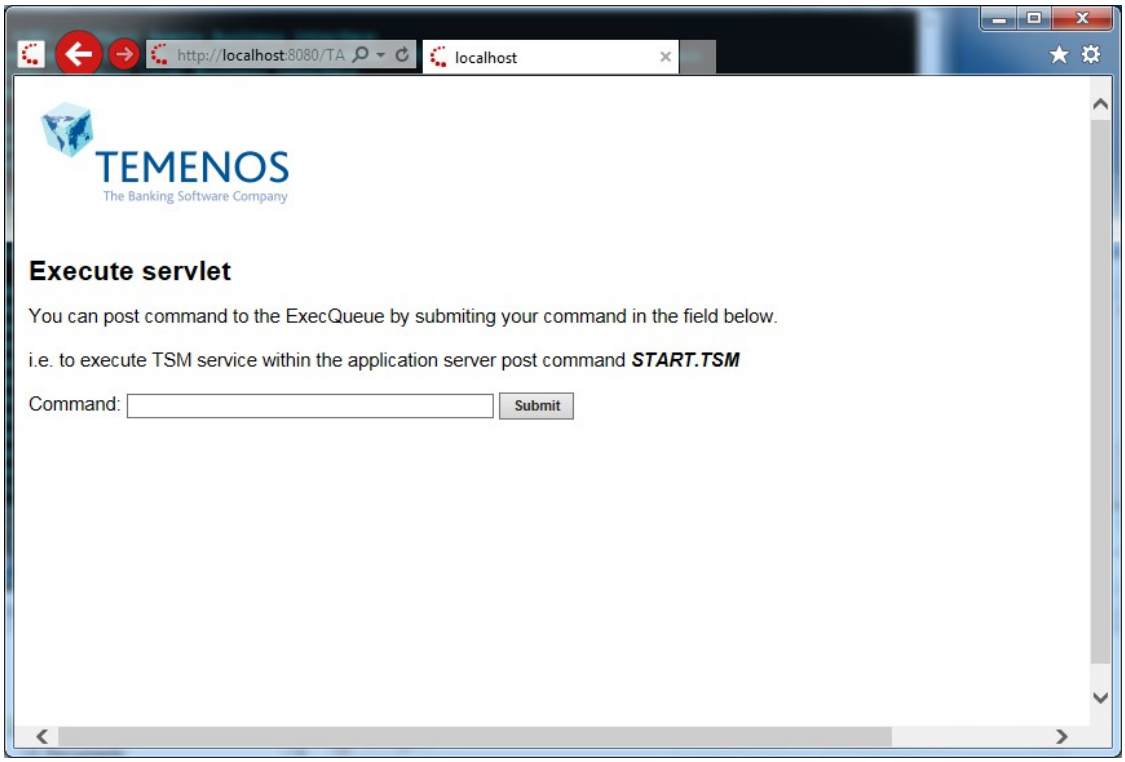
to get details about JF.INITIALISE.CONNECTION





Running a COB with TAFJ: ExecuteServlet

With this web application you could launch command to T24 specially START.TSM.



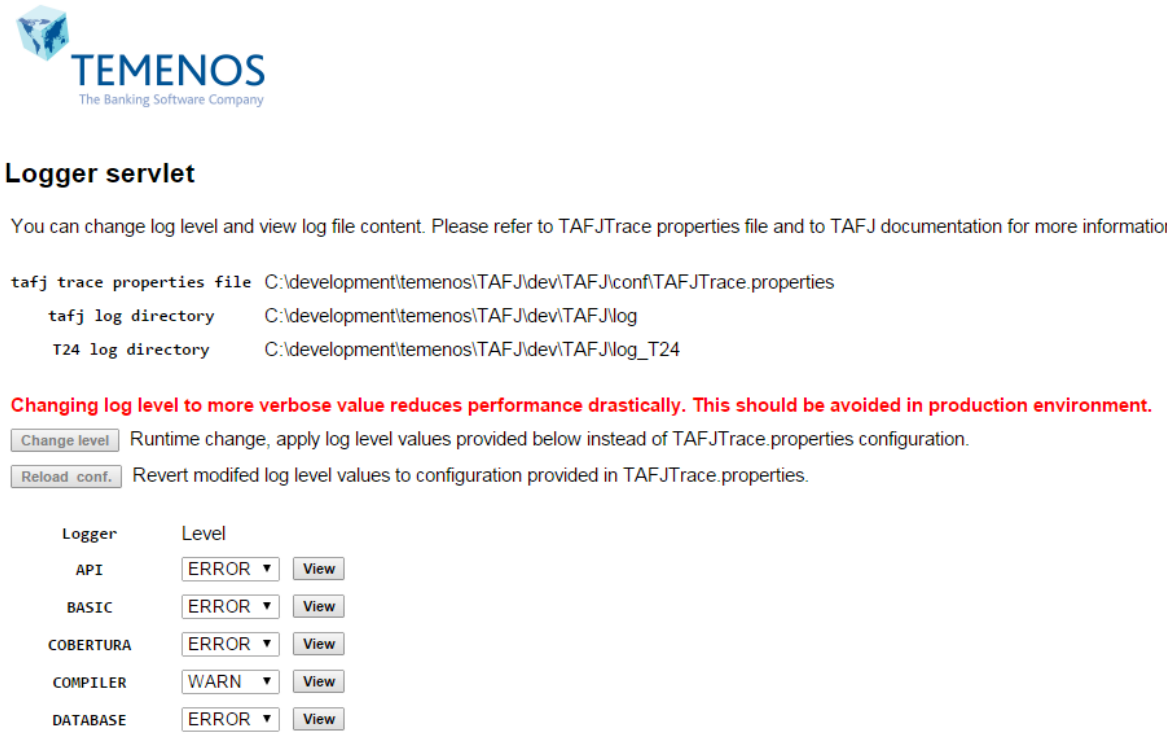
To launch the COB :

Post START.TSM in the form.

Check that the como folder is created and como's file was generated in it.

Changing log level and access log file content : LoggerServlet

http://localhost:8080/TAFJEE/logger



This servlet allows changing dynamically the log level of any TFAJ loggers without having to restart the application server.

This is a runtime change which doesn't impact the TAFJTrace properties file.

It could be useful in case of unexpected application behavior where you want to temporarily change the log level to a most verbose value to get more information.

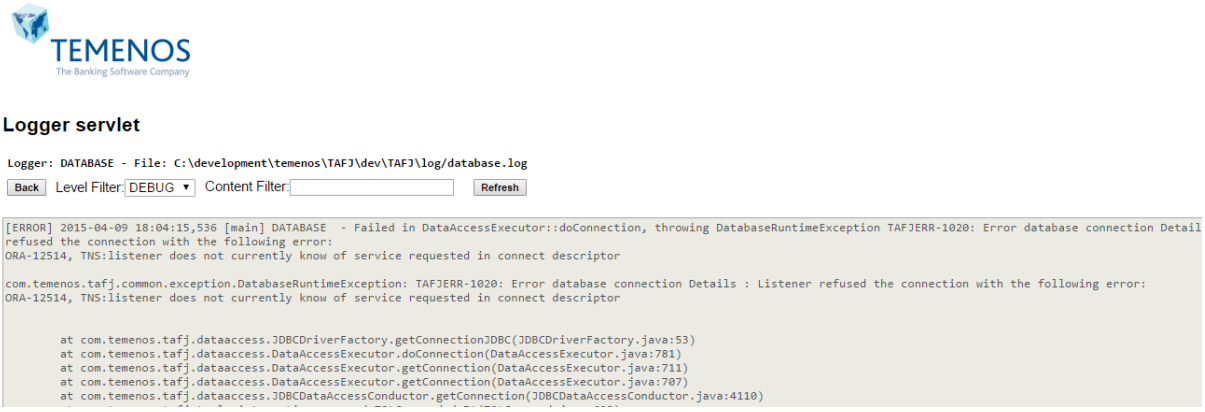
As logging is a performance killer, it should be used with care in production environment.

To change the log level for a specific logger, select from the combo box the appropriate level and apply the change by clicking the "Change level" button.

Once enough information has been collected, the initial level could be reverted by clicking "Reload conf" or changing the value from the combo box.

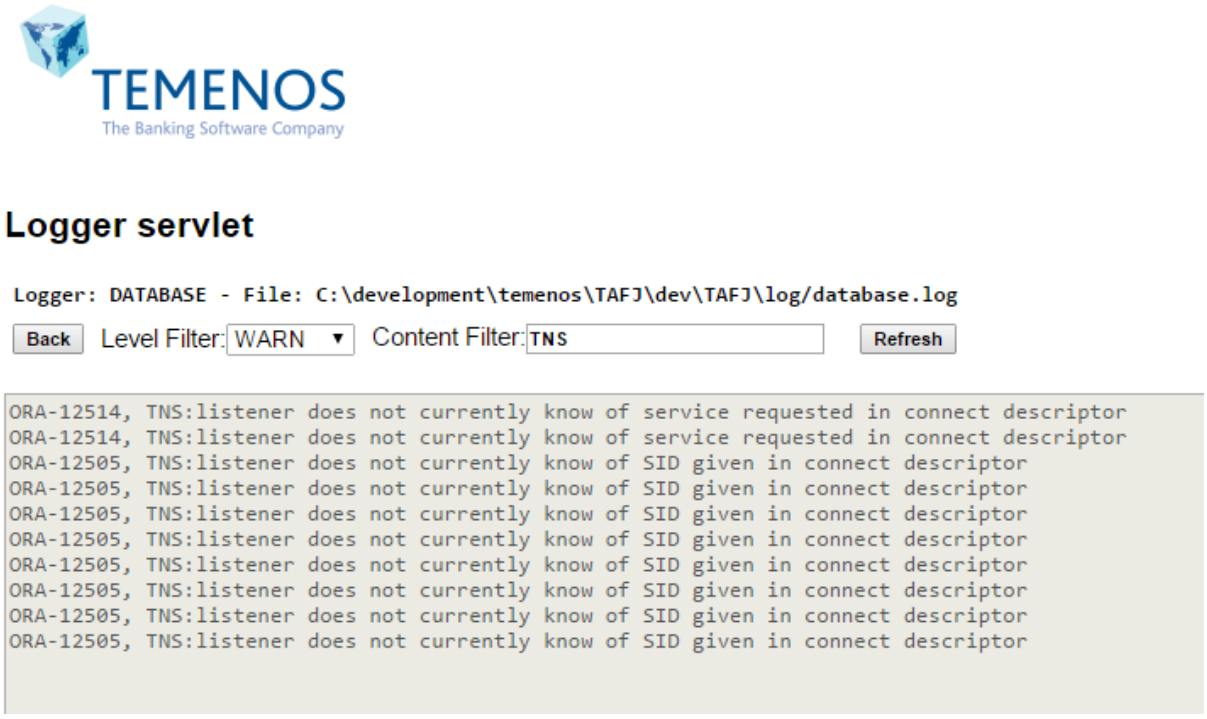
It's also possible to directly access the log file content from the servlet, by clicking the "View" button.

The current log file content will be displayed. If there is log file rotation because of many information reported you will have to access the physical log folder to collect the history.



This log viewer also provides filtering functionalities if you are interested in a specific information or log level.

You could narrow the log level with the combo box to keep only logging information equals or higher to the selected level, i.e. if you select “WARN” you will get WARNING and ERROR messages only.



Accessing COMOs : ComoServlet

http://localhost:8080/TAFJEE/como

You could browser como file content from this servlet.

This tool is mainly a helper to provide direct access to the como and **is not a COB monitor, there is no automatic refresh functionality, you will get a picture of the como folder at a point of time only.**

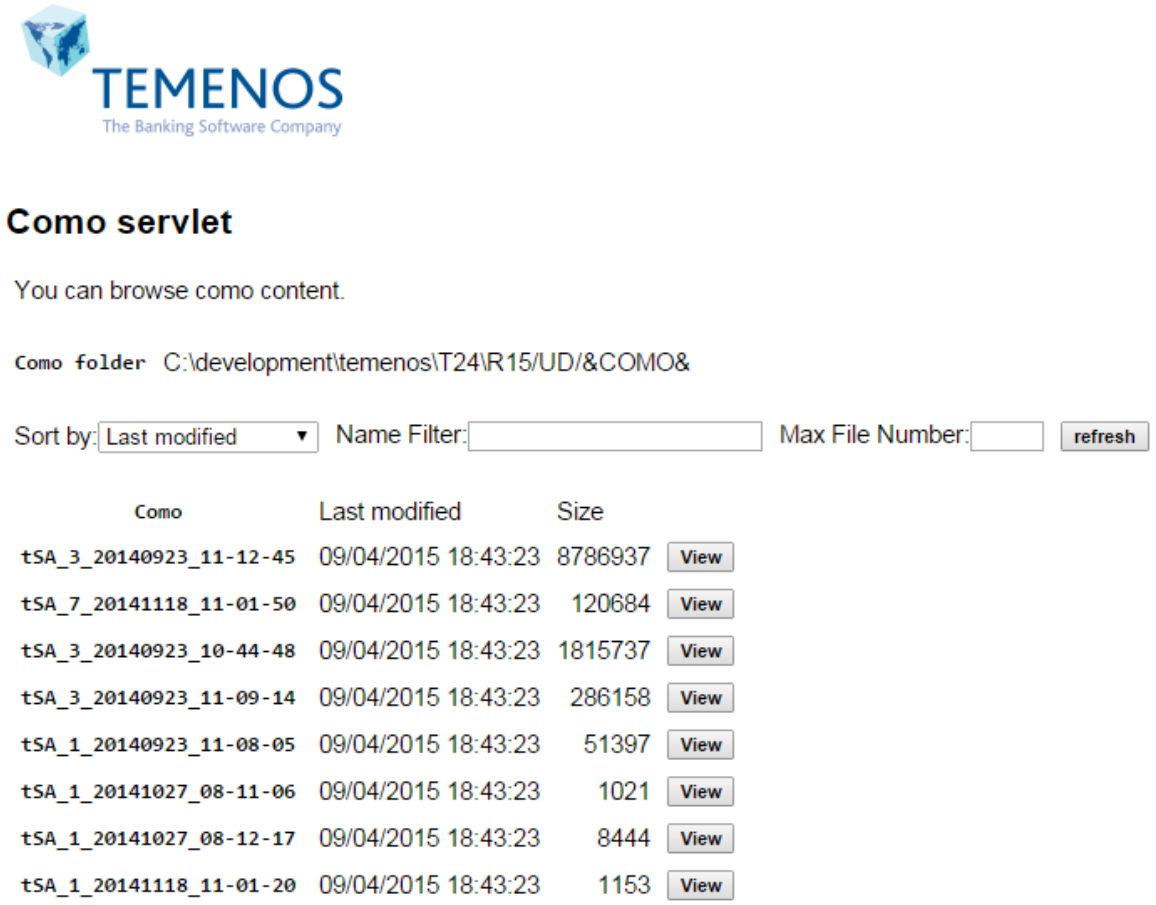
You could recall the servlet periodically to get an updated view.

Please note that the como size (in bytes) is being displayed and consider this information before viewing huge file from the servlet as the file content will be loaded into memory.

You could sort comos by name, last modified date by changing the sort option value in the combo box.

You could also request to see only a specific file by providing its name or the last 10 files for example.

The same browsing functionalities than described in the logger servlet section are available.



TEMENOS
The Banking Software Company

Como servlet

Como: tSA_3_20140923_11-12-45

Back Content Filter: Refresh

```
COMO tSA_3_20140923_11-12-45 established 11:12:45  23 SEP 2014
<como>
<agent>3</agent>
<processid>6188</processid>
<portno>2</portno>
Agent 3 started 23 SEP 14 11-12-45
Agent's Process id 6188
<servername>L2DHHQ12</servername>
Running on server L2DHHQ12 PortNumber 2
<service name = COB>

 7 Records selected

_BNK/REPORT.PRINT.APPLICATION_EB.EOD.REPORT.PRINT_3_23 SEP 2014_11:12:45:762_SELECT F.COMPANY WITH CONSOLIDATION.MARK NE 'C' Selected=7 time=0secs

 1533 Records selected

_BNK/REPORT.PRINT.APPLICATION_EB.EOD.REPORT.PRINT_3_23 SEP 2014_11:12:45:762_SSELECT F.BATCH BY BATCH.STAGE Selected=1533 time=0secs
```

TEMENOS
The Banking Software Company

Como servlet

Como: tSA_3_20140923_11-12-45

Back Content Filter: RGS Refresh

```
FILE FBNK.RGS.LCS.BY.EXPIRY  CLEARED
FILE FBNK.RGS.LC.ACPT.DUE  CLEARED
FILE FBNK.RGS.LC.DRAWINGS  CLEARED
FILE FBR1.RGS.LD0500  CLEARED
FILE FBR1.RGS.LD0110  CLEARED
FILE FBNK.RGS.LD0500  CLEARED
FILE FBNK.RGS.LCS.BY.RSKPTY  CLEARED
FILE FBNK.RGS.MM0005  CLEARED
FILE FBNK.RGS.LD0100  CLEARED
FILE FBR1.RGS.LD0200  CLEARED
FILE FBNK.RGS.LD0200  CLEARED
FILE FBNK.RGS.MM0001  CLEARED
FILE FBR1.RGS.LD0300  CLEARED
FILE FBR1.RGS.MM0004  CLEARED
FILE FBR1.RGS.MM0003  CLEARED
FILE FEU1.RGS.LD0200  CLEARED
```

DBTools servlet

You could execute DBTools command from TAFJJEE application by accessing the following servlet.

http://localhost:8080/TAFJEE/DBTools

Please refer to DBTools documentation to get information about DBTools capabilities and DBTools command syntax.

DBTools servlet has to be used in conjunction with DBTools.jar. If DBTools.jar is not part of the application server classpath, the functionality will not be available.

It's really important to note that DBTools provides a full database access and the tool should not be deployed on production environment or to a strictly restricted set of users.

The purpose of the tool is to provide access to DBTools command through the application server deployment to database administrator during development or testing phases.

By executing a DBTools command, a message will be posted to the execute channel (Exec queue) invoking DBTOOLS basic replacement deployed within DBTools.jar.

This is an asynchronous innovation as it could be a long running job.

For familiar DBTools users, behind the scene the command will be executed in script mode and with a log file as result renderer:

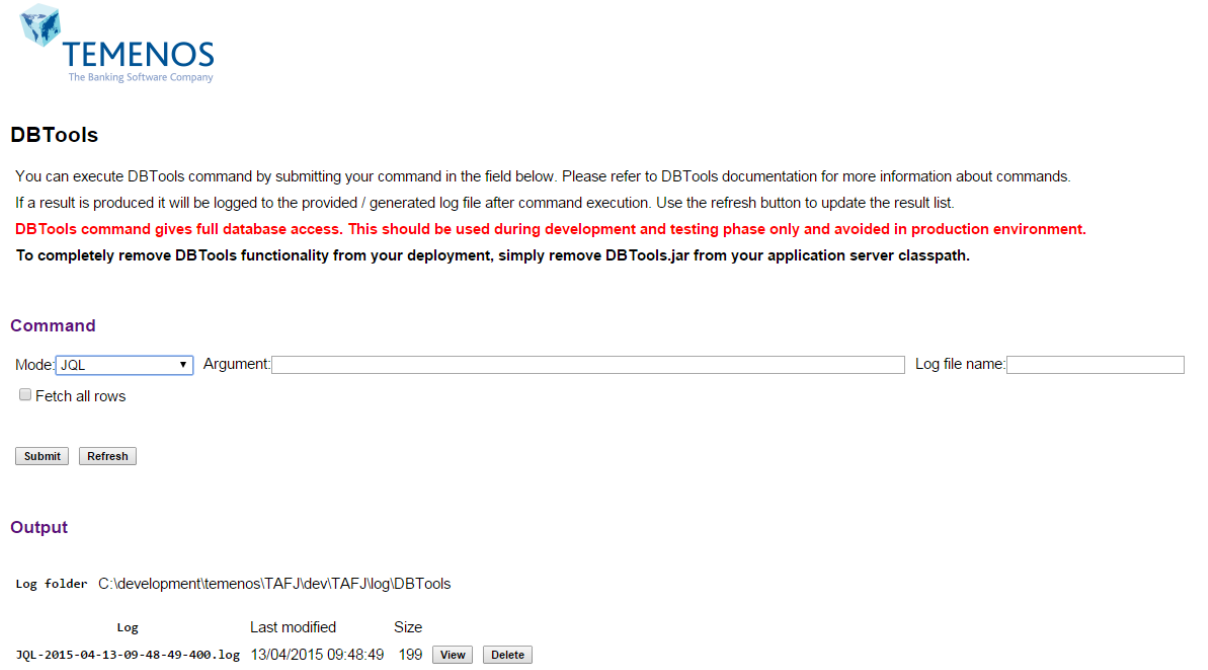
```
tRun DBTOOLS -s -log aLogFileName SQL SELECT \\* FROM FBNK_CURRENCY
```

The command will be executed in background and produce an output to a default generated log file (command type - timestamp) or to the user specified log file name.

To execute the command, select from the combo box the command type, enter in the argument field the command argument and optionally a log file name.

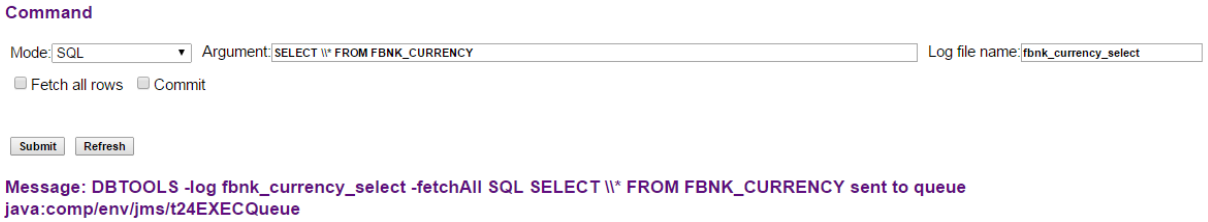
Click the "Submit" button, the JMS message corresponding to the command is being posted and displayed. Depending on the command an output will be generated and available in the output list after pressing the "Refresh" button.

When executing SQL command you will need to use the "Commit" check box for commands updating data.

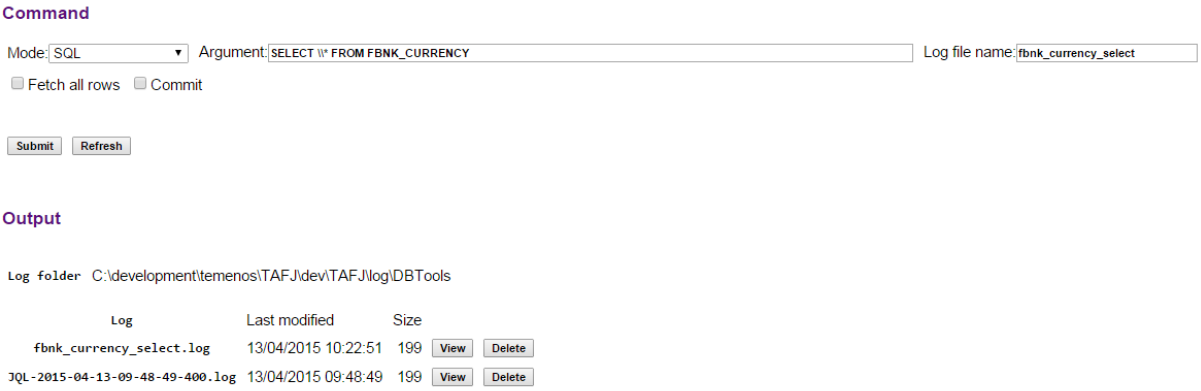


i.e. Enter a SQL command

Press “Submit”



Press “Refresh”, the log file fbnk_currency_select.log has been generated.



Please note that the log file size (in bytes) is being displayed and consider this information before viewing huge file from the servlet as the file content will be loaded into memory.

Log file could be deleted if they are not more needed.

TAFJ Entry points documentation

TAFJJEE_WAR_TAFJ contains an online documentation which explains how to interact with TAFJJ EE.

http://localhost:8080/TAFJEE/html/interaction.html

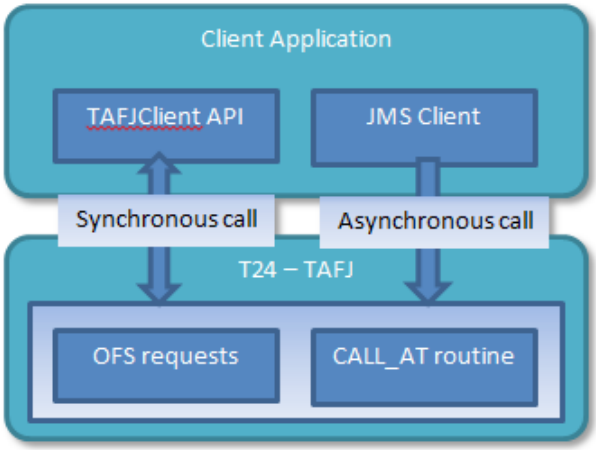
Refer to following section about TAFJ EE entry points.

TAFJ EE Entry points

TAFJEE provides entry points to process OFS requests and to invoke routines with parameters.

This could be done in two different ways:

- Synchronously by doing a Webservice or EJB invocation with TAFJClient API.
- Asynchronously by sending a JMS message with a JMS client or by using CALLJEE statement.



Synchronous invocation

There is two different entry points to synchronously process OFS requests and to CALL routines from a client application:

- Webservices
- EJB

Although both methods can be achieved easily by using TAFJClient API, EJB invocation requires a bit more configuration and deployment knowledge.

TAFJJEEClientFactory is the entry point to get a TAFJJEEClient either for Webservice or EJB invocation.

```
package com.temenos.tafj.j2ee.client.impl;

public class TAFJJEEClientFactory

public static TAFJJEEClient getWebServiceClient(String host-
name, String port)

public static TAFJJEEClient getEjbClient(AppServerProvider
appServer, String hostname, String port)
```

TAFJJEEClient is the interface which provides methods to process an OFS request or to call a subroutine.

```
package com.temenos.tafj.j2ee.client;

public interface TAFJJEEClient

String[] callAt(String routineName, String[] parameters);

String processOFS(String request);
```

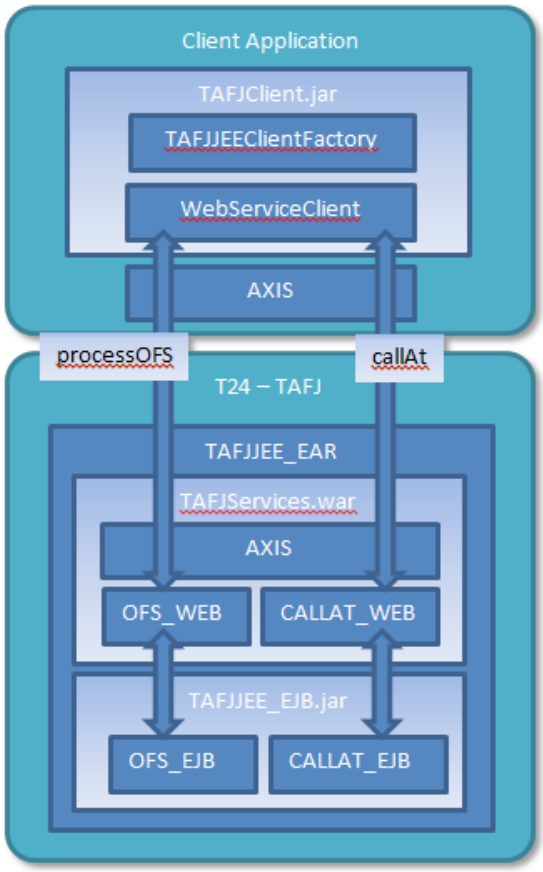
AppServerProvider is an enumeration used by the factory to retrieve the EJB client corresponding to the application server version. As a helper, it also defines ports which are usually used by default for HTTP request and EJB lookup.

```
package com.temenos.tafj.j2ee.client;

public enum AppServerProvider

JBOSS("8080", "1099"), //JBoss 4.2.3 - EAP 5.2
WEBLOGIC("7001", "7001"), //Weblogic 10.3 - 12.1.1
WEBSPHERE("9080", "2809"), //Websphere 7 - 8 - 8.5
JBOSS7("8080","4447"); //JBoss 7 - EAP 6.2

private final String defaultHttpPort;
private final String defaultEJBPort;
```

Synchronous invocation – Webservice

Webservice invocation is deployment agnostic. The call is done through the HTTP channel of the application server.
TAFJJEEClientFactory arguments to get a webservice client are the server hostname and the http port of the application server.

```
//Get a Webservice client from server 10.21.2.99
TAFJJEEClient client = TAFJJEEClientFactory.getWebService-
Client("10.21.2.99", "8080");

//Process an OFS request, method argument is the OFS request
String response = client.proces-
sOFS("ENQUIRY.SELECT,,INPUTT/123456,%CURRENCY");

//Invoke a subroutine, method arguments are the Subroutine name
and an array of subroutine parameters
String[] response = client.callAt("EXCHRATE", new String[] {
"1", "CHF", "500", "GBP", "", "", "", "", "", "" });
```



The client application classpath must contain **Axis 2 libraries**.
Axis 2 libraries can be extracted from **TAFJJEE_EAR.ear/TAFJServices.war/WEB-INF/lib**.

Synchronous invocation – EJB

As EJB lookup requires specific parameters such as initial context name and port, EJB invocation depends on the application server version.
TAFJJEEClientFactory arguments are the application server version,
see AppServerProvider, the server hostname and the application server port for EJB lookup.

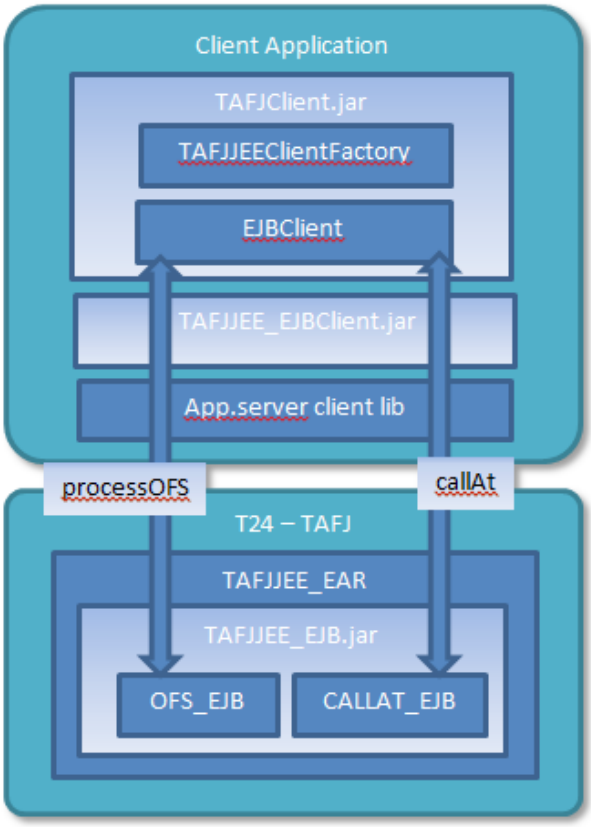
In case of unsupported application server version please refer to the custom EJB invocation section.

```
//Get an EJB client from a JBoss7 deployment on server
10.21.2.99
TAFJJEEClient client = TAFJJEEClientFactory.getEjb-
Client(AppServerProvider.JBOSS7, "10.21.2.99", "4447");

//Process an OFS request, method argument is the OFS request
String response = client.proces-
sOFS("ENQUIRY.SELECT,,INPUTT/123456,%CURRENCY");

//Invoke a subroutine, method arguments are the Subroutine name
and an array of subroutine parameters
String[] response = client.callAt("EXCHRATE", new String[] {
"1", "CHF", "500", "GBP", "", "", "", "", "", "" });
```

The client application classpath must contain **TAFJJEE_EJBClient.jar**.

The client application classpath must contain the **application server client libraries** to process the EJB lookup.
Please refer to the EJB classpath setup section to get more details.

TAFJJEE_EJBClient.jar can be extracted from **TAFJJEE_EAR.ear/APP-INF/lib**.



Custom EJB invocation

Custom EJB invocation allows a specific initial context configuration and an override of the default EJBs name used during lookup: **OFSProcessingBean** and **CallAtProcessingBean**.

TAFJJEEClientFactory arguments are the initial context name to invoke, the context provider URL and a map of properties to be applied on the initial context.

```
//TAFJJEEClientFactory method to get a custom EJB client
public static TAFJJEEClient getEjbClient(String INITIAL_CON-
TEXT_FACTORY, String PROVIDER_URL, Map<Object, Object> con-
textProperties)

//TAFJJEEClient methods to parametrize the OFS and CALLAT bean
name
public void setOFSBeanName(String beanName);

public void setCALLATBeanName(String beanName);
```

The sample below illustrate a JBoss 6.2 EAP custom invocation.

```
//Create the map of specific properties necessary to lookup an
initial context from the application server
Map<Object, Object> map = new HashMap<Object, Object>();
map.put("jboss.naming.client.ejb.context", true);

//Factory invocation, the initial context name and URL are pro-
vided with the map of specific properties
TAFJJEEClient client = TAFJJEEClientFactory.getEjb-
Client("org.jboss.naming.remote.client.InitialContextFactory",
"remote://localhost:4447", map);

//TAFJJEEClient setup - specify the EJBs name to be used during
lookup - note the cast from TAFJJEEClient to EJBClient
((com.temenos.tafj.j2ee.client.EJBClient)client).setOFSBeanName
("TAFJJEE_EAR/TAFJJEE_EJB//OFSProcessingBean!
com.temenos.tafj.sb.OFSProcessingBeanRemote");
((com.temenos.tafj.j2ee.client.EJBClient)client).setCALLATBean-
Name("TAFJJEE_EAR/TAFJJEE_EJB//CallAtProcessingBean!
com.temenos.tafj.sb.OFSProcessingBeanRemote");

//Classic OFS processing
String response = client.proces-
sOFS("ENQUIRY.SELECT,,INPUTT/123456,%CURRENCY");
//Classic subroutine invocation
String[] response = client.callAt("EXCHRATE", new String[] {
"1", "CHF", "500", "GBP", "", "", "", "", "", "" });
```

Client application classpath setup

The section below provides some application server client libaries examples. Please refer to the application server documentation for more details about its client libraries for remote invocation.
These jars could be added manually to the client application classpath or with maven whenever the dependencies are available in the maven repository.

JBoss

- JBoss 4.2.3

```
$JBOSS_HOME/client/jbossall-client.jar
```

- JBoss EAP 5.2

```
//Please note this jar just contain a MANIFEST referencing other
jars from $JBOSS_HOME/client, see META-INF/MANIFEST.MF.
$JBOSS_HOME/client/jbossall-client.jar
```

- JBoss 7 - EAP 6.2

```
Maven dependency:
<dependency>
  <groupId>org.jboss.as</groupId>
  <artifactId>jboss-as-ejb-client-bom</artifactId>
  <version>7.2.0.Final</version>
  <type>pom</type>
</dependency>
```

Weblogic

- Weblogic 10.3.6

```
$WEBLO_HOME/wlserver/server/lib/wlthint3client.jar
```

- Weblogic 12.1.1

```
$WEBLO_HOME/wlserver_12.1/server/lib/wlthint3client.jar
```

Websphere

- WAS 8.5.5

```
$WAS_HOME/runtimes/com.ibm.ws.ejb.thinclient_8.5.0.jar
$WAS_HOME/runtimes/com.ibm.ws.orb_8.5.0.jar
```

To be able to do remote EJB invocation in a Websphere environment from a thin client you would need to generate EJBs stubs for the TAFJ EJB client library and add it to the client application classpath.

Please refer to Create Stub command IBM documentation,

http://www-01.ibm.com/support/docview.wss?uid=swg21393419

i.e. ${was.home}/bin/createEJBStubs TAFJJEE_EJB.jar -newfile TAFJJEE_EJB_Stubs.jar -cp $CLASSPATH

TAFJServices.war – Webservice component

TAFJServices.war is an Axis based archive to provide a webservice access to OFS and CALL_AT functionalities.

OFS webservice

Process the ofs request specified in the argument *Request.*

WSDL

http://localhost:8080/TAFJServices/services/OFSService?wsdl

Invocation

Send http request on:

/TAFJServices/services/OFSService/Invoke?Request=THE_OFS_REQUEST

i.e.

http://localhost:8080/TAFJServices/services/OFSService/Invoke?Request=ENQUIRY.SELECT,,INPUTT/654321,ACCOUNT.DETAILS,CURRENCY:EQ=EUR

Subroutine Invoker webservice (CALL_AT)

Execute the routine specified in the argument *Subroutine* with parameters specified in argument(s) *Param.* Routine and arguments are separated with character *&.*

WSDL

http://localhost:8080/TAFJServices/services/InvokerService?wsdl

Invocation

Send http request on:

/TAFJServices/services/InvokerService/Invoke?Subroutine=THE_ROUTINE_NAME&Param=PARAM1_VALUE&Param=PARAM2_VALUE …

i.e. to execute EXCHRATE routine which takes 10 parameters.

http://localhost:8080/TAFJServices/services/InvokerService/Invoke?Subroutine=EXCHRATE&Param=1&Param=CHF&Param=500&Param=GBP&Param=&Param=&Param=&Param=&Param=&Param=

TEC Events

T24 could be configured to publish TEC events. TAFJ components are configured to publish TEC events via res-ref-name jms/TopicConnectionFactory for the JMS Topic connection factory and jms/tecEventsTopic for the JMS topic resource.

To disable the TEC Events publishing from TAFJ Runtime you could set the property temn.tafj.runtime.enable.jms.logger to false in the tafj.properties. It will stop the TEC events publishing without changing the T24 TEC configuration.

TAFJ Sessions Monitor

To setup the TAFJ Sessions Monitor, untar the file TAFJSessionMonitor.tar.gz in $TAFJ_HOME. From $TAFJ_HOME/TAFJSessionMonitor

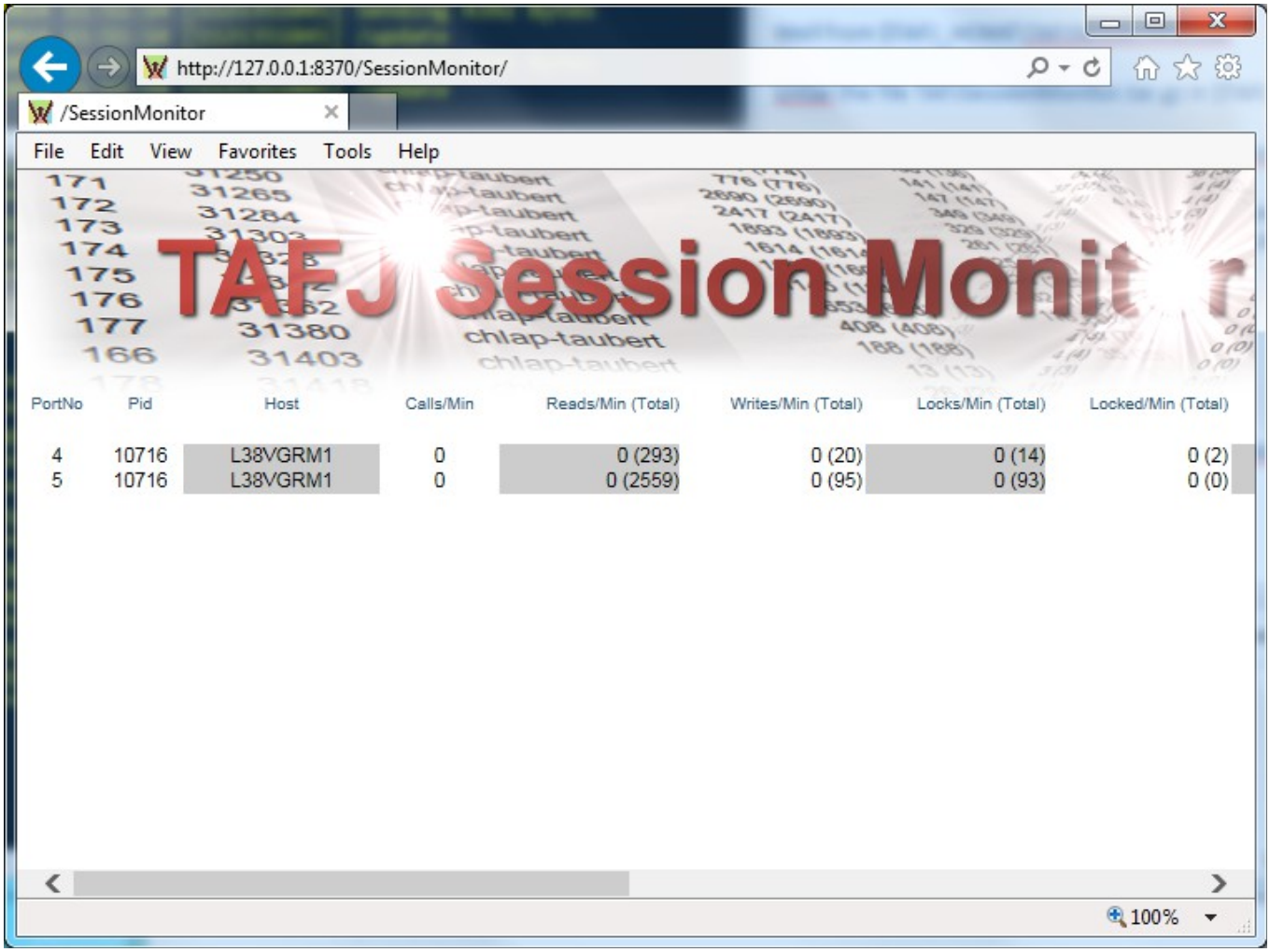
1. Check the file : $TAFJ_HOME /TAFJSessionMonitor/config/SessionMonitor.properties

   tcp_port=8377 <= the port used between TAFJ and TAFJSessionMonitor

   http_port=8370 <= the port used for the HTTP URL

2. In $TAFJ_HOME /TAFJSessionMonitor/bin start start.sh or start.bat
3. Browse http://<IP>:< http_port >/SessionMonitor/

i.e http://127.0.0.1:8370/SessionMonitor/



Now with TAFJ in the properties file setup :

```
#*******************************************************************
#
#  TAFJ Monitor
#
#*******************************************************************
# Enable the TAFJMonitorSession
#
temn.tafj.runtime.session.monitor.enable        = true  <= to enable the monitor


# Host name or IP of where is the TAFJMonitorSession
```

```
#
temn.tafj.runtime.session.monitor.host          = localhost <= where is your
TAFJSessionMonitor


# TCP port of the TAFJMonitorSession
#
temn.tafj.runtime.session.monitor.port          = 8377<= the tcp_port of TAFJSessionMonitor
```

TAFJ Technical Monitor

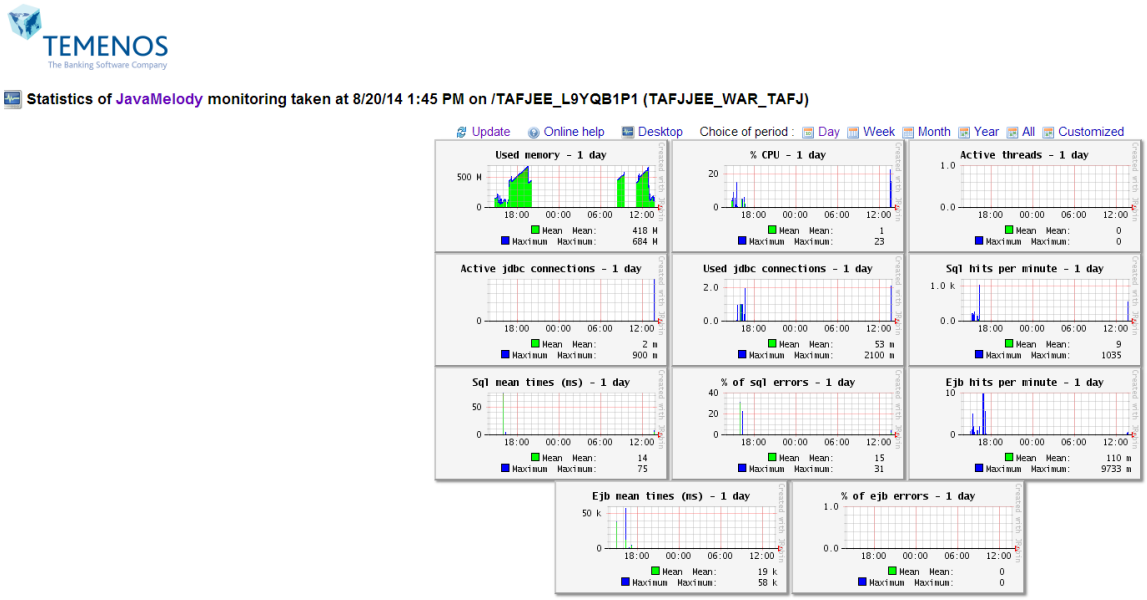
TAFJEE application could be monitored by browsing the following URL.

http://<host>:<port>/**TAFJEE/monitoring**

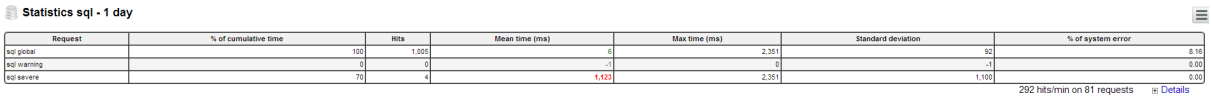
Monitoring is achieved using java melody which is an open source application to monitor java EE applications.

With very low overhead it gives some statistics and charts about:

- Memory, threads, CPU



- SQL statements (could be disabled with a parameter)



Statistics sql - 1 day

Request	% of cumulative time	Hits	Mean time (ms)	Max time (ms)	Standard deviation	% of system error
sql global	100	1,005	6	2,351	92	8.16
sql warning	0	0	-1	0	-1	0.00
sql severe	70	4	1,123	2,351	1,100	0.00

292 hits/min on 81 requests Details

- Numbers of EJBs call and response time (could be removed , specific configuration)

Statistics ejb - 1 day

Request	% of cumulative time	Hits	Mean time (ms)	Max time (ms)	Standard deviation	% of cumulative cpu time	Mean cpu time (ms)	% of system error	Mean hits sql	Mean time sql (ms)
ejb global	100	2	4,376	8,159	5,349	100	210	0.00	346	2,161
ejb warning	0	0	-1	0	-1	0	-1	0.00	-1	-1
ejb severe	0	0	-1	0	-1	0	-1	0.00	-1	-1

0 hits/min on 1 requests Summary by class Details

- Error logs (could be disabled with a parameter)

Statistics system errors logs - 1 day

Error	Hits
ERROR [EJB] Error initializing TAFJ session.	1

0 hits/min on 3 errors Details Last errors

- MBeans and active sessions

System information

Execute the garbage collector View memory histogram Hotspots
View deployment descriptor MBeans View OS processes JNDI tree Opened jdbc connections Database

Host: L9YQB1P1@10.242.1.26
Java memory used: 137 Mb / 455 Mb
Nb of active threads (current http requests): 0
Nb of active jdbc connections: 1
Nb of used jdbc connections (opened if no datasource): 2 Details

Threads

Threads on L9YQB1P1@10.242.1.26: Number = 75, Maximum = 77, Total started = 92 Details

Last collect time: 110 ms
Display time: 101 ms
Memory overhead estimate: < 1 Mb
Disk usage: 3 Mb

Return Update

MBeans

- JMImplementation
- TAFJ
  - Session
    - Id=214426193

(Management Bean.)

- AccountName	INPUTTER-
- AllJimiStats	[0, 0, 0, 0, 0, 0, 0, 0, 0]
- Background	false
- BasicProfiling	null
- CallStackLast	N/A
- CallStackTab	[]
- ConfigFileLoc	C:\product\AppServer\jboss-4.2.3.GA\..\..\TAFJ\DEV\conf\tafj.properties
- Configuration	[temn.tafj.appserver.mode = , temn.tafj.appserver.name = , temn.tafj.channel.class.0 = com.temenos.tafj.jlp.drivers.jPrinterDriver, temn.tafj.channel.class.1 = com.temenos.tafj.jlp.drivers.jPrinterDriver, temn.tafj.channel.device.0 = PDFCreator, temn.tafi.channel.device.1 = PDF Architect.

- Database informations (depends on the DB provider)

Return Update Sessions

Database : Sessions

SID NUMBER(22)	USERNAME VARCHAR2(30)	OSUSER VARCHAR2(30)	MACHINE VARCHAR2(64)	MODULE VARCHAR2(48)	STATUS VARCHAR2(8)	OPTIMIZER_MODE VARCHAR2(10)	SQL_TEXT VARCHAR2(1000)
9	TAFJ	SYSTEM	L9YQB1P1	JDBC Thin Client	INACTIVE	ALL_ROWS	DELETE FROM LOCK_RECORDS WHERE SESSIONID = :1
67	TAFJ	SYSTEM	L9YQB1P1	JDBC Thin Client	INACTIVE	ALL_ROWS	DELETE FROM LOCK_RECORDS WHERE SESSIONID = :1
201	TAFJ	SYSTEM	L9YQB1P1	JDBC Thin Client	ACTIVE	ALL_ROWS	select sesion.sid, username, osuser, machine, sesion.module, status, optimizer_mode, sql_text from v$sqlarea sqlarea, v$session sesion where sesion.sql_hash_value = sqlarea.hash_value(+) and sesion.sql_address = sqlarea.address(+) and sesion.username is not null order by username, sql_text
14	TAFJ	SYSTEM	L9YQB1P1	JDBC Thin Client	INACTIVE		
138	TAFJ	SYSTEM	L9YQB1P1	JDBC Thin Client	INACTIVE		
72	TAFJ	SYSTEM	L9YQB1P1	JDBC Thin Client	INACTIVE		

Java melody overview

Full java melody documentation could be found here:

https://code.google.com/p/javamelody/

Taken from java melody documentation:

"The goal of JavaMelody is to monitor Java or Java EE application servers in QA and production environments. It is not a tool to simulate requests from users, it is a tool to measure and calculate statistics on real operation of an application depending on the usage of the application by users.

JavaMelody is mainly based on statistics of requests and on evolution charts."

Java melody doesn't require a database to store events nor code instrumentation.

Therefore it could be enabled in production because of its very low overhead (from null to 5%).

Discussion about java melody overhead could be found here:

https://code.google.com/p/javamelody/wiki/Overhead

That being is said even if it's not advisable monitoring could be disabled and even totally removed from TAFJJEE application in case of need.

Java melody configuration

Java melody libraries could be found under TAFJJEE_EAR/APP_INF/lib:

- javamelody.jar, core library, which is a patched version of official java melody to cover additional need.
- jrobin.jar, RRD tool java implementation, to log data and do graph rendering.

Monitoring filter and listener

The monitoring functionalities are enabled through a servlet filter called MonitoringFilter, declared in TAFJJEE_EAR/TAFJJEE_WAR_TAFJ/webapp/WEB-INF/web.xml.

```
      <filter>
            <filter-name>monitoring</filter-name>
            <filter-class>net.bull.javamelody.MonitoringFilter</fil-
ter-class>
      </filter>
```

```
<filter-mapping>
      <filter-name>monitoring</filter-name>
      <url-pattern>/*</url-pattern>
</filter-mapping>
```

Java melody also defines a SessionListener to monitor http sessions. We don't make use of it for TAFJJEE.
Please refer to the known issue section when deploying in websphere to get the filter correctly initialized.

Parameters

There are several parameters that could be configured to refine java melody setup:

https://code.google.com/p/javamelody/wiki/UserGuide#6._Optional_parameters

This section presents those we use to monitor TAFJJEE.

Parameters could be defined in several ways: as filter parameter, context parameter or system properties at application server level. System property takes precedence on context parameter which takes precedence on filter parameter.

We define them in TAFJJEE_EAR/TAFJJEE_WAR_TAFJ/webapp/WEB-INF/web.xml as Context parameters to have a unified way to do it and to avoid application server restart to apply a value change.

Disable monitoring

To disable monitoring simply set `javamelody.disabled=true.`

```
<!-- turn off javamelody -->
<context-param>
<param-name>javamelody.disabled</param-name>
<param-value>false</param-value>
</context-param>
```

Counters setup

The parameter `javamelody.displayed-counters` is used to override the counters displayed by default. As TAFJJEE doesn't make use of http session we remove some of them.

```
<!-- counters - we don't want to display the default one
    "http,sql,error,log"
as we don't need http monitoring -->
<context-param>
<param-name>javamelody.displayed-counters</param-name>
<param-value>sql,log,ejb</param-value>
</context-param>
```

Data source setup

By default a jndi lookup is issued to retrieve all data sources declared at application level.

The optional parameter `javamelody.datasources` allows declaring the data sources to monitor if there is a need to exclude some (i.e. the locking data source or websphere internal EJB timer derby data source).

```
<!-- datasources to be monitored -->
      <!-- they could be retrieved dynamically through jndi lookup but we
might want to exclude some -->
      <!-- i.e. WAS internal derby DS for EJBTimer or locking DS -->
      <context-param>
            <param-name>javamelody.datasources</param-name>
            <param-value>java:comp/env/jdbc/t24DataSource,java </param-
value>
      </context-param>
```

When defining multiple data sources, they should be coma separated. The first one will be displayed in the “Database Informations” screen. They also need to be defined at TAFJJEE_WAR_TAFJ.war level to have the jndi lookup resolved (optional if the property is not used).

History storage directory

By default java melody stores .rrd files for graphs and .gz files for statistics on disk in the temporary directory of the application server or in the temporary directory of the host, defined by system property `java.io.tmpdir.`

This is the directory to clear to clean up all statistics and graphs.

The property `javamelody.storage-directory` is used to override this default temporary directory.

```
<!-- statistic storage directory -->
<context-param>
<param-name>javamelody.storage-directory</param-name>
<param-value>C:\javamelody-storage-dir</param-value>
</context-param>
```

Taken from the documentation

“If the name of the directory starts with '/' (or on Windows, with drive specifier followed by '\', or if its prefix is "\\"), it is considered as an absolute path, otherwise it is considered as relative to the temporary directory. If this parameter is changed it is recommended to rename the physical directory at the same time.”

Disable SQL and JDBC monitoring

The property `javamelody.no-database` will disable all SQL and JDBC monitoring when set to `true.` In that case SQL counters won’t be displayed.

```
<!-- turn off sql monitoring when set to true -->
<context-param>
<param-name>javamelody.no-database</param-name>
<param-value>false</param-value>
</context-param>
```

Error log level

The property `log-threshold-level` is a specific property in the patched TAFJJEE javamelody version.
By default java melody reports warning and error messages reported by the different application logging systems (java util logging, log4j, logback).
This parameter allows refining this setup. By default we want to report only ERROR messages.
It could also be used to turn off this functionality.
Making use of lowest logger level value is not advisable as it can be really verbose and this is not the intended purpose of this functionality which is to report only last 100 logs messages received.
Valid values for this parameter are ERROR - WARNING - INFO - DEBUG - OFF.
When set to OFF, the log counter won't be displayed.

```
<context-param>
<param-name>javamelody.log-threshold-level</param-name>
<param-value>ERROR</param-value>
</context-param>
```

Development mode parameters

This parameter is not advisable in production and should be used only during development phase.
By adding property `javamelody.sampling-seconds` hotspot detection will be enabled and stack-trace sampling will be executed every x seconds according the property value.
10 seconds is the recommended value for the lowest overhead, but could be lower (1, 0.1) in case of faster test result expected.
For more details see:
https://code.google.com/p/javamelody/wiki/UserGuideAdvanced#Enable_Hotspots_detection

```
<!-- hotspot detection -->
<context-param>
<param-name>javamelody.sampling-seconds</param-name>
<param-value>10</param-value>
</context-param>
```

`javamelody.sampling-excluded-packages` property allows refining the default list of package excluded during hot spot detection.

```
<context-param>
<param-name>javamelody.sampling-excluded-packages</param-name>
<param-value>java,sun,com.sun,javax,org.apache,org.hibernate,oracle,org.-
    postgresql,org.eclipse,org.jboss,com.arjuna,org.jnp</param-value>
</context-param>
```

EJB monitoring

To monitor EJBs requests an interceptor has to be declared in the ejb-jar.xml of the EJB to be monitored.

The default java melody interceptor is `net.bull.javamelody.MonitoringInterceptor.`

It can be replaced with a specific TAFJ interceptor to classify request per OFS source.

```
<interceptors>
      <interceptor>
            <interceptor-class> com.temenos.tafj.monitoring.interceptor.-
      MonitoringClassifierInterceptor
            </interceptor-class>
      </interceptor>
</interceptors>
```

This interceptor needs to be associated to the targeted EJB / MDB within the assembly descriptor section of the ejb-jar.xml.

i.e. to monitor all EJBs

```
<assembly-descriptor>
…
<interceptor-binding>
            <ejb-name>*</ejb-name>
            <interceptor-class>com.temenos.tafj.monitoring.interceptor.Mon-
itoringClassifierInterceptor
</interceptor-class>
</interceptor-binding>
```

By default TAFJJEE is configured to monitor all EJBs activity.

It could be completely disabled by removing this interceptor declaration or by restricting the ejb name to the EJB to be monitored instead of mapping them all:

```
     <ejb-name>*</ejb-name>

     Could be replaced with

     <ejb-name>BROWSERProcessingBean</ejb-name>
```

To monitor multiple specific EJBs, the interceptor binding section should be replicated per EJB.

Limitation and known issues

Websphere application server

Monitoring filter initialisation

When deploying java melody in websphere, up to version 8.5, the monitoring filter won't be automatically initialized at application startup as it should be according the servlet specification.

http://www-01.ibm.com/support/docview.wss?uid=swg1PM62909

You will need to add a custom property:

```
com.ibm.ws.webcontainer.invokeFilterInitAtStartup = true
```

to the webcontainer of the server, as mentioned in the link above. Otherwise the monitoring filter won't start until the TAFJEE servlet gets invoked.

SQL monitoring

By default javamelody core doesn't provide rewrapping to monitor websphere data source.

It will produce following error.

```
FFDC Exception:javax.naming.NamingException
SourceId:com.ibm.ws.naming.util.Helpers.processJavaObjectForBinding ProbeId:682
Reporter:java.lang.Class@a18ab83c


javax.naming.NamingException: com.sun.proxy.$Proxy43.getReference() returned null
in violation of the JNDI API
at com.ibm.ws.naming.util.Helpers.processJavaObjectForBinding(Helpers.java:670)
at com.ibm.ws.naming.jndicos.CNContextImpl.doRebind(CNContextImpl.java:2076)
at com.ibm.ws.naming.jndicos.CNContextImpl.rebind(CNContextImpl.java:697)
at com.ibm.ws.naming.util.WsnInitCtx.rebind(WsnInitCtx.java:233)
at com.ibm.ws.naming.util.WsnInitCtx.rebind(WsnInitCtx.java:245)
at org.apache.aries.jndi.DelegateContext.rebind(DelegateContext.java:177)
at javax.naming.InitialContext.rebind(InitialContext.java:452)
at
net.bull.javamelody.JdbcWrapperHelper.rebindDataSource(JdbcWrapperHelper.java:119
)
at net.bull.javamelody.JdbcWrapper.rebindDataSources(JdbcWrapper.java:452)
```

TAFJ is shipped with a patched version of javamelody to fix this issue.

JBoss 6 EAP

Mbeans

Mbeans are not being displayed in JBoss 6.1 EAP because of following JBoss redhat issue.

"error JMX JBAS019905 Should not get called"

https://issues.jboss.org/browse/WFLY-838

Clearing all statistics and graphs

Stop the server and clean up the directory defined in property `javamelody.storage-directory.`

Deleting the .rrd files will clean up the graphs, deleting .gz files will clean up the statistics.

Securing web applications

As presented in above sections, TAFJJEE_WAR_TAFJ.war and TAFJServices.war provide access to several functionalities that might need a protected access:

- TAFJEE/ExecuteServlet to post JMS message to the ExecQueue
- TAFJEE/monitoring to access TAFJ technical monitor
- TAFJServices/services/OFSService to process OFS request through webservice
- TAFJServices/services/InvokerService to process CALL_AT through webservice

A sample configuration is available to add a BASIC Authentication (role, user and password) to these applications. It consists in a common configuration part at webapp level and in a specific configuration part at webapp and application server level depending on the appserver provider.

Basic Authentication common configuration

Comment out in the web application **/webapp/WEB-INF/web.xml** the section related to security at the end of the file.

Sample below is extracted from TAFJJEE_WAR_TAFJ.war but same applies with TAFJServices.war.

This sample illustrates how to add a basic authentication for all URLs under TAFJEE context. It means only authenticated users with role TAFJAdmin could access these URLs. It could be refined to have a security role per URL.

```
<!-- Security Stuff A template configuration to secure the webApp -->
 <security-constraint>
       <web-resource-collection>
             <web-resource-name>TAFJEE</web-resource-name>
             <url-pattern>/*</url-pattern>
       </web-resource-collection>
       <auth-constraint>
             <role-name>TAFJAdmin</role-name>
       </auth-constraint>
 </security-constraint>
 <security-role>
       <role-name>TAFJAdmin</role-name>
 </security-role>
 <login-config>
       <auth-method>BASIC</auth-method>
```

```
            <realm-name>TAFJRealm</realm-name>
        </login-config>
```

Basic Authentication JBoss

Webapp

Comment out in the web application **/webapp/WEB-INF/jboss-web.xml** the section related to security at the end of the file.

```
<!-- A template configuration to secure the webApp -->
<security-domain>java:/jaas/TAFJRealm</security-domain>
```

Jboss 4/5 deployment

Define the security Realm declared in the jboss-web.xml file under

server/context/conf/login-config.xml

```
<application-policy name="TAFJRealm">

    <authentication>

      <login-module
    code="org.jboss.security.auth.spi.UsersRolesLoginModule"
    flag="required">

            <module-option name="usersProperties">props/tafj-
    users.properties</module-option>

            <module-option name="rolesProperties">props/tafj-
    roles.properties</module-option>

      </login-module>

    </authentication>

  </application-policy>
```

Create files

server/context/conf/props/tafj-user.properties

add a user and password, i.e.

```
tafj=password
```

server/context/conf/props/tafj-roles.properties

add a role TAFJAdmin and associate it to tafj user

```
tafj=TAFJAdmin
```

Jboss 7 (EAP6) deployment

Add a security domain in the security domains section of the standalone.xml file under **server/standalone/configuration**.

```
<security-domain name="TAFJRealm" cache-type="default">
      <authentication>
            <login-module code="UsersRoles" flag="required">
            <module-option name="usersProperties" value="${jboss.server.-
            config.dir}/tafj-users.properties"/>
            <module-option name="rolesProperties" value="${jboss.server.-
            config.dir}/tafj-roles.properties"/>
            </login-module>
      </authentication>
</security-domain>
```

Create files

server/standalone/configuration/ tafj-user.properties

add a user and password, i.e.

```
tafj=password
```

server/standalone/configuration/tafj-roles.properties

add a role TAFJAdmin and associate it to tafj user

```
tafj=TAFJAdmin
```

You could refer to

https://community.jboss.org/wiki/JBossAS7SecureMyWebAppHowDoI?_sscc=t

Basic Authentication Weblogic

Webapp

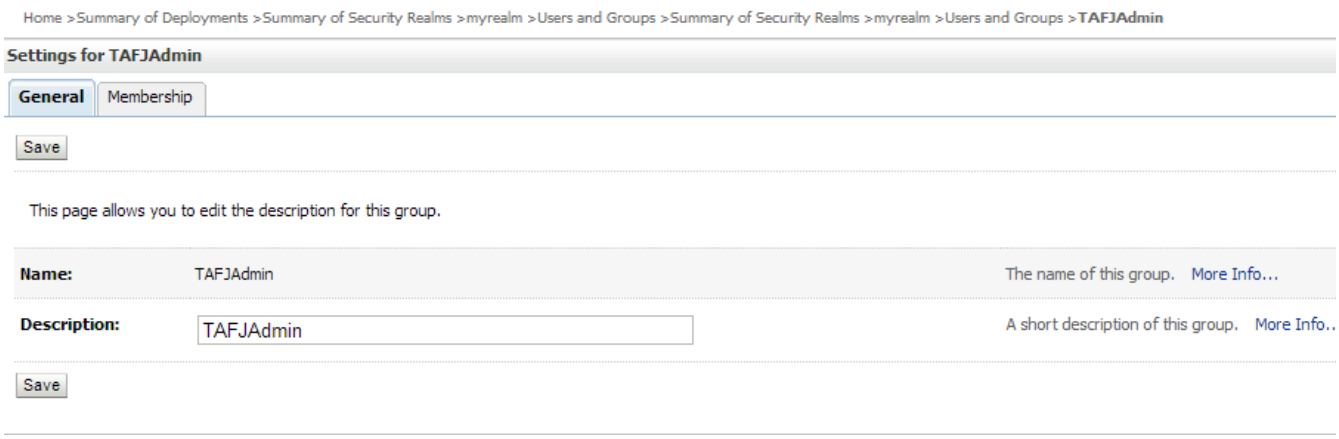
Comment out in the web application **/webapp/WEB-INF/weblogic.xml** the section related to security at the end of the file.

```
<!-- A template configuration to secure the webApp     -->
<security-role-assignment>
       <role-name>TAFJAdmin</role-name>
       <principal-name>TAFJAdmin</principal-name>
</security-role-assignment>
```

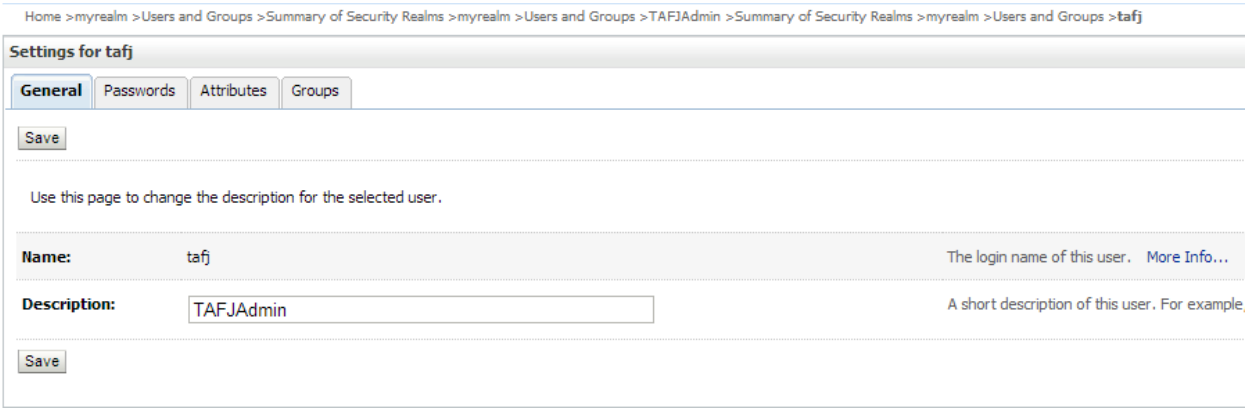
Weblogic 12c deployment

Add a security realm through weblogic console or use the default one “myrealm”.

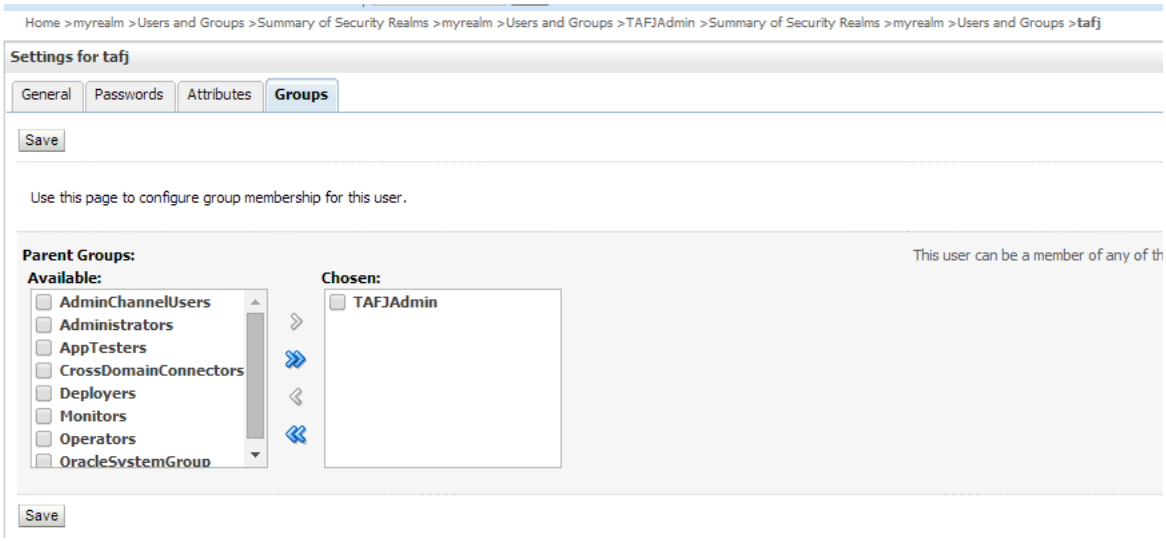
Add group TAFJAdmin



Add a user tafj



Add group TAFJAdmin to user tafj



Basic Authentication Websphere

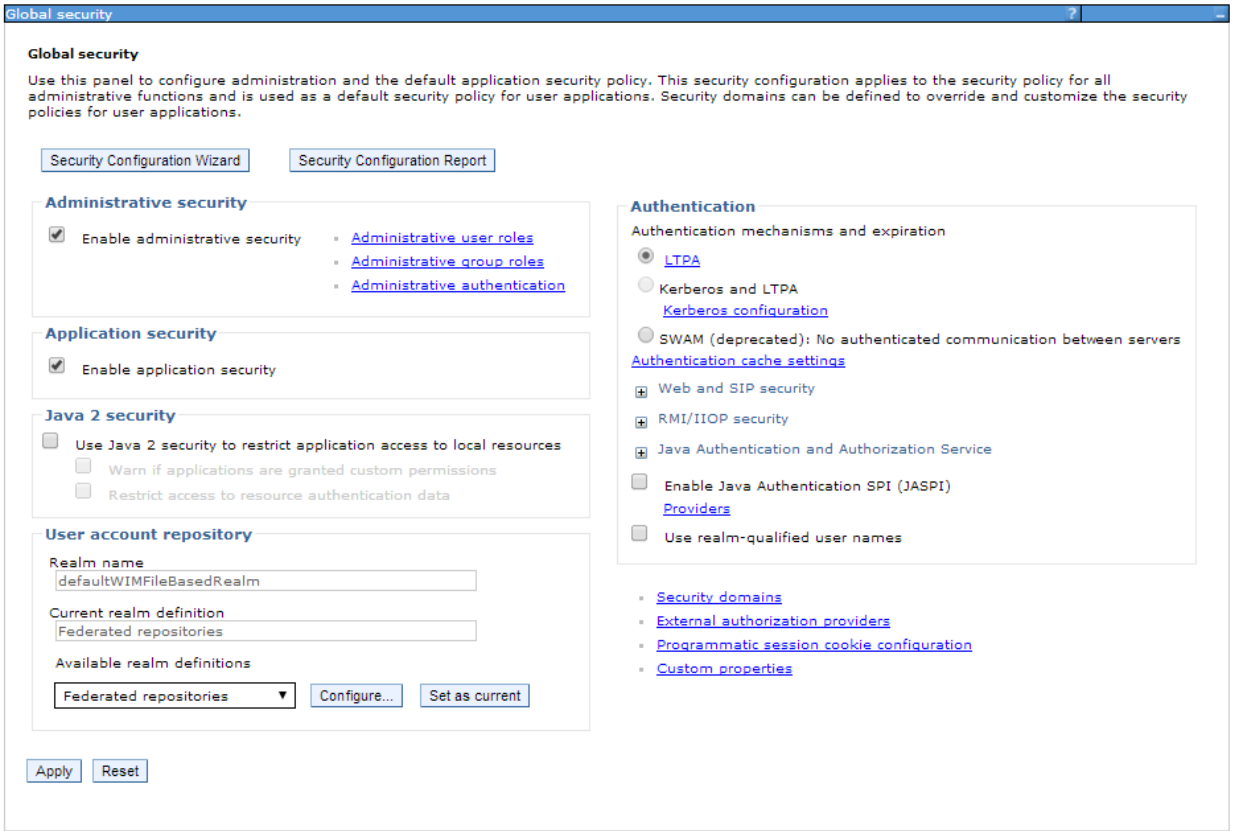
Webapp

No specific change applies to webapp.

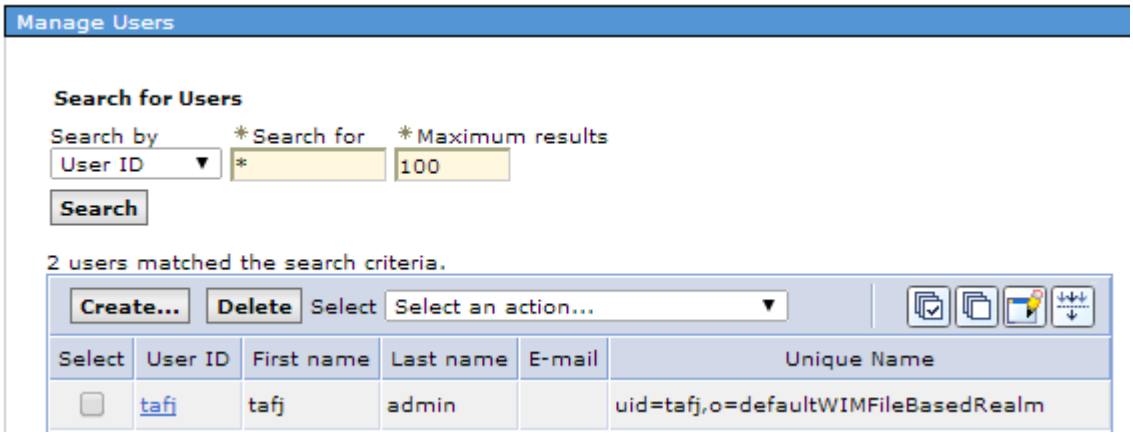
WAS 8.5 deployment

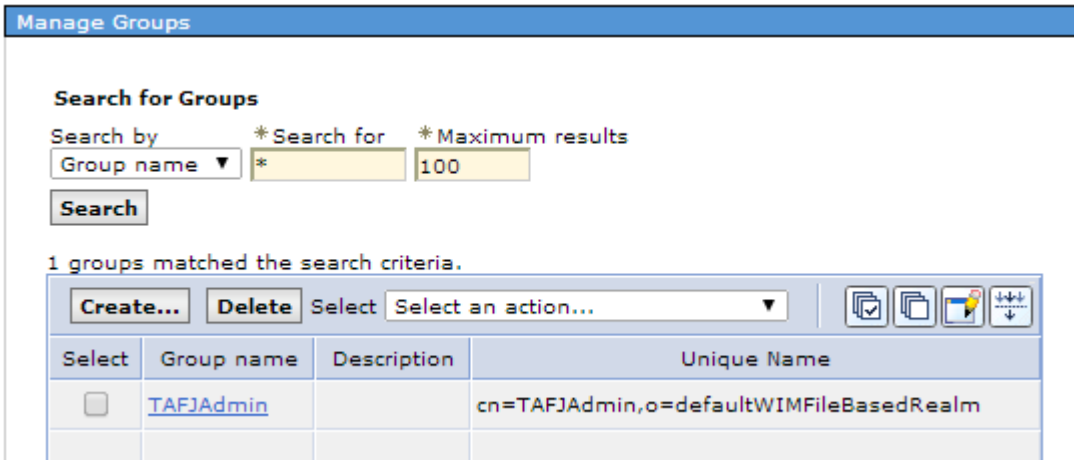
The Websphere profile must be secured to perform the following configuration.

Enable administrative security and application security.



Create user and group.





Map TAFJAdmin role defined at webapp level to TAFJAdmin group.