



OpenESB Standalone Edition V3.0

Web admin console

Document identifier:

Pymma document: 770-003

Location:

www.pymma.com

Editor:

Pymma Services: contact@pymma.com

Abstract:

This document explains how to manage OpenESB Standalone Edition with the web console. It provides a guide to install component and shared libraries, deploy services assemblies and monitor your instance.

Status:

This Document is in a beta state

ABOUT PYMMA CONSULTING

Pymma Services is a technical architect bureau founded in 1999 and headquartered in London, United Kingdom. It provides expertise in service oriented integration systems design and implementation. Leader of OpenESB project, Pymma is recognised as one of the main actors in the integration landscape. It deeply invests in open source projects such as Drools rules engine. Pymma is a European company based in London with regional offices in France, Belgium and Canada. (contact@pymma.com or visit our website on www.pymma.com)

Copyright

Copyright © 2014, Pymma Services LTD. All rights reserved. No part of this publication may be copied or distributed, transmitted, transcribed, stored in a retrieval system, or translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, manual, optical, chemical or otherwise; or disclosed to third parties without the express written permission of Pymma Services LTD, Inc.

Disclaimer

Information in this document is subject to change without notice and does not represent a commitment on the part of Pymma Services LTD. This manual is provided “as is and Pymma is not responsible for and expressly disclaims all warranties of any kind with respect to third-party content, products, and services. Pymma will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services

Trademark Notice

Pymma is a registered of Pymma Engineering LTD. Java is registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

1	Introduction.....	7
1.1	OpenESB entities.....	7
1.1.1	Shared Library.....	7
1.1.2	Components.....	7
1.1.3	Service unit.....	8
1.1.4	Service assembly.....	8
2	OpenESB life cycles.....	9
2.1	Shared library life cycle.....	9
2.2	Component life cycle.....	9
2.3	Service assembly life cycle.....	10
2.4	Service Unit lifecycle.....	10
3	Web admin console overview.....	11
3.1	Console configuration.....	11
3.2	Browser.....	11
3.3	Log to the console.....	12
3.3.1	Create a new user.....	12
3.3.2	Change a password.....	14
3.4	Console overview.....	14
4	Dashboard.....	15
5	Instances.....	17
5.1	General.....	17
5.2	Logger.....	18
5.2.1	Set up log level.....	18
5.3	NMR Monitoring.....	19
5.4	System Monitoring.....	20
6	Shared Libraries.....	21
6.1	Install a shared library.....	21
6.2	Uninstall a shared library.....	22
6.3	Shared library details.....	22
7	Components.....	24
7.1	Install a new component.....	24
7.2	Uninstall a component.....	25
7.3	Component detail.....	26
7.3.1	General.....	26
7.3.2	Configuration.....	27
7.3.3	Application configuration.....	28
7.3.4	Application variables.....	30
7.3.5	Descriptor.....	32
7.3.6	Loggers.....	32

7.4	Shared libraries	33
7.5	Monitoring	33
8	Services assemblies.....	35
8.1	Service unit overview	35
8.2	Service assembly overview.....	36
8.2.1	Deploy a new service assembly	37
8.3	Undeploy a service assembly.....	38
8.4	Service assembly detail.....	39
8.4.1	General.....	39
8.4.2	Service Unit overview.....	40
8.4.3	Service assembly descriptor	41
8.4.4	Service assembly monitoring	41
9	Endpoints	43
10	What's next	44
11	Help and support.....	45
11.1	From the community.....	45
11.2	From Pymma.....	45

1 Introduction

OpenESB Standalone edition embedded a new, light and smart web admin console. Very fast, it allows administrator to manage an OpenESB instance and install, deploy, start, stop components or services assemblies. In this document, we will provide a short summary of OpenESB components' life cycles. This knowledge is essential to manage OpenESB. Likewise, we recommend you to read JBI specifications and understand how components or service assemblies interact with the bus. <http://download.oracle.com/otndocs/jcp/jbi-1.0-fr-eval-oth-JSpec/>

1.1 OpenESB entities

OpenESB defines 5 entities to manage. They are:

1. Shared library
2. Service Engine
3. Binding component
4. Service unit
5. Service assemblies

In the next paragraphs, we will explain in brief what these entities in OpenESB are and what their roles are. If you know these concepts, jump to the next chapter.

1.1.1 Shared Library

Shared libraries are Java libraries that can be shared by many components. By default, OpenESB components use 3 shared libraries.

- wsdlsl.jar
- wsdlxlib.jar
- encoderlib.jar

The two first libraries are helpful to process WSDL documents. The last one encoderlib.jar allows components to support encoding features. It is up to the OpenESB administrator to decide which shared library must be installed regarding the component he wants to install later. We advise you to install wsdlxlib.jar and encoderlib.jar before installing the components.

1.1.2 Components

OpenESB supports two kinds of components: Service Engines and Binding Components. OpenESB only distinguishes between these two kinds through a flag; the model and API are otherwise identical. However, by convention, Service Engines and Binding Components implement different functionalities in OpenESB.

1.1.2.1 Binding component

Binding components are used to send and receive messages via particular protocols and transports. They serve to isolate the OpenESB environment from the particular protocol by providing normalisation and denormalisation from and to the protocol-specific format, allowing the OpenESB environment to deal only with normalized messages.

1.1.2.2 Service engines

Service Engines are the business logic drivers of the OpenESB. Engines can orchestrate service consumption and provision, in the course of, for example, executing long-lived business processes. Other engines can provide simple services, such as data transformation. Yet other engines may provide sophisticated routing or EDI services such as

message collation / de-collation facilities. Service Engines can create new services by aggregating other services. SEs can serve as service providers, service consumers, or both.

1.1.3 Service unit

A “Service Unit” or SU is a single deployment package, destined for a single component. The contents of a SU are opaque to JBI (other than a single descriptor file), but are transparent to the component to which it is being deployed, as well as the design-time tooling that produced the artefacts contained by the SU. The service unit must contain a single JBI-defined descriptor file that defines the static services produced and consumed by the service unit.

1.1.4 Service assembly

Often multiple deployments are required to create a new service or consumer application within an OpenESB environment. To support this directly, OpenESB provides a composite deployment capability, where deployments meant for different components can be grouped into a Service Assembly, or SA. Such an assembly includes a composite service deployment descriptor, detailing to which component each Service Unit contained in the SA is to be deployed. Note that this service assembly concept is sometimes termed “composite service description”, or CSD. A service assembly represents a composite service. Because of this interrelationship, OpenESB provides management functions to control the life cycles of the individual units of a service assembly collectively, rather than individually.

2 OpenESB life cycles

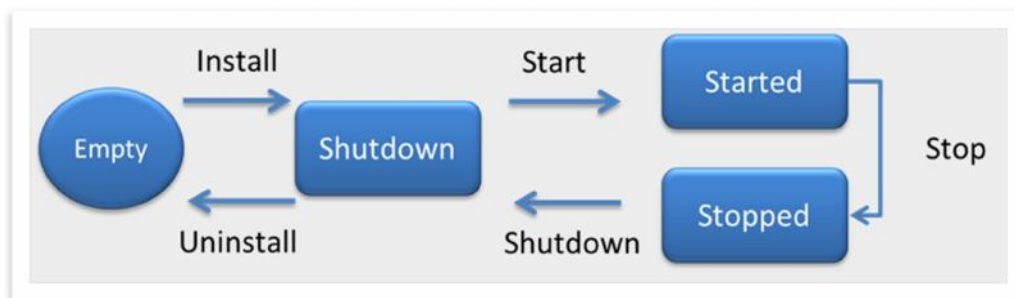
OpenESB web admin console does not offer the same life cycles than the ones defined in JBI specifications. We make it simpler and more accurate to production tasks.

2.1 Shared library life cycle



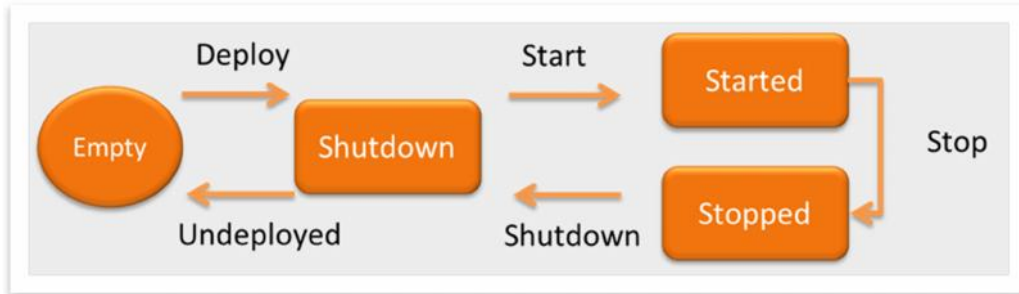
A shared library can have 2 states: Empty (or not installed) and Installed. When a shared library is installed, it becomes available for the components to be deployed.

2.2 Component life cycle



A component can have 4 states: Empty (or not installed), Shutdown (Installed but not started), Started and Stopped. Stopped stated means that a component does not accept any new message but run for the message already in process. OpenESB web console does not offer a Shutdown to Stopped transition.

2.3 Service assembly life cycle



A service assembly can have 4 states: Empty (or not deployed), Shutdown (deployed but not started), Started and Stopped. OpenESB web console does not propose a Shutdown to Stopped transition.

2.4 Service Unit lifecycle

OpenESB web admin console does not offer a way to manage Service Unit lifecycle as detailed in JBI specifications since Service Unit and Service Assembly are synchronized.

Service unit state	Service Assembly State
Empty	Empty
Shutdown	Shutdown
Started	Started
Stopped	Stopped

3 Web admin console overview

3.1 Console configuration

By default, OpenESB web admin console is listening on the port 4848. To access to the console check if the OpenESB instance you want to manage is running. Open a browser and type the address with the following pattern: `http://${host}/${port}/webui` where `${host}` is the host where OpenESB instance is running and `${port}` the listening port for the console. By default the address to access to the console is [http://localhost/4848/webui](http://localhost:4848/webui). Default port can be changed to set up a more accurate port. To do it, modify the content of `${OESE-HOME}\OE-Instance\config\openesb.yaml`.

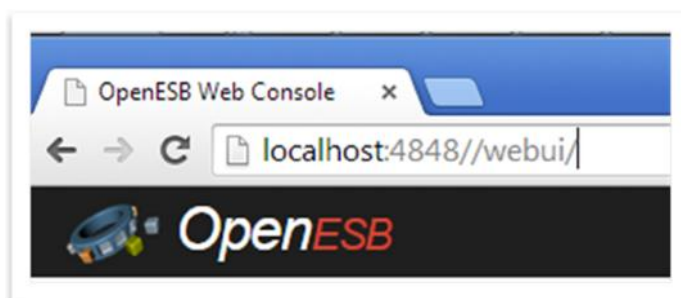
```
##### HTTP #####
# Set a custom port to listen for HTTP traffic:
# http.port: 4848

# Disable HTTP completely:
# http.enabled: false

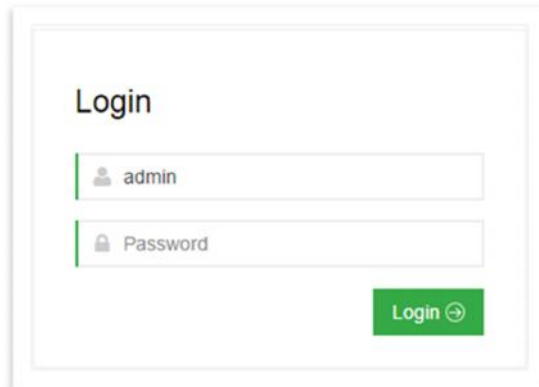
# HTTP Binding
# http.binding: localhost
```

3.2 Browser

OpenESB web admin console has been tested with the latest versions of Firefox, Chrome and Safari. Exhaustive tests have not been made with Internet Explorer.



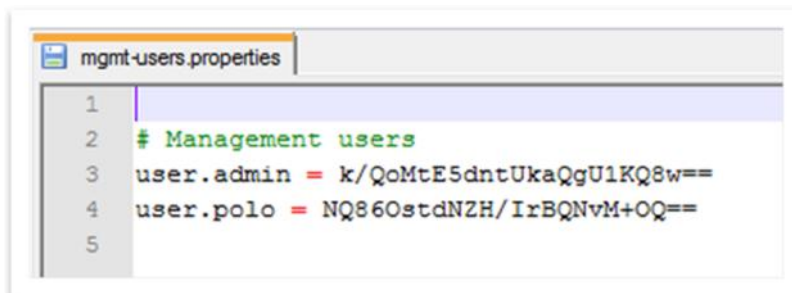
3.3 Log to the console



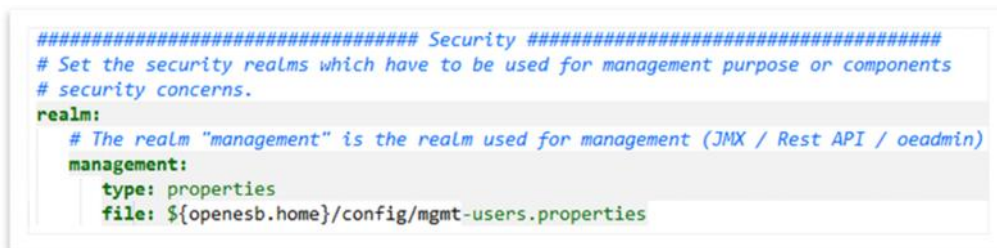
OpenESB web console access is protected by a login / password screen. By default login = admin and password = admin. You can change the admin password or add new users. There is not user hierarchy with more or less rights in the web admin console. You simply have the right to access the console or not.

3.3.1 Create a new user

OpenESB web admin console users are stored in the following file `${OESE-HOME}\OE-Instance\config\mgmt-users.properties`.



Login and password are stored in users.properties file. Passwords are encrypted for obvious security purposes. So, the administrator can provide access to the console to new users by adding new properties in this file. You can use any other properties files to store the users and their password. The properties file used by OpenESB is defined in the `${OESE-HOME}\OE-Instance\config\openesb.yaml`.



Let's add a new user "pymma" with the password "Services". In the mgmt.-users.properties file, add the user pymma:

```
mgmt-users.properties
1 # Management users
2 user.admin = k/QoMtE5dntUkaQgU1KQ8w==
3 user.polo = NQ86OstdNZH/IrBQNVm+OQ==
4 user.pymma = ???
5
```

To encrypt the password, OpenESB proposes you a simple tool to it. This tool is the java class PasswordManagement found in jar file `${OESE-HOME}\OE-Instance\lib\openesb-standalone-container-xxxxx`, where xxxxx is the version or the build number.

Open a console and execute the following command where my_password is the password you want to encrypt:

```
java -cp lib/openesb-standalone-container-xxxxx.jar
net.openesb.standalone.security.utils.PasswordManagement my_password
```

```
F:\OpenESB-SE-3.0\OE-Instance\lib>java -cp openesb-standalone-container-xxxxx.jar
Generate encrypted password for <Services>
Encrypted password is: 0kAZjg0YHG9WtUr5lezGKw==
F:\OpenESB-SE-3.0\OE-Instance\lib>_
```

Copy and paste the password to the properties files.

```
mgmt-users.properties
1 # Management users
2 user.admin = k/QoMtE5dntUkaQgU1KQ8w==
3 user.polo = NQ86OstdNZH/IrBQNVm+OQ==
4 user.pymma = 0kAZjg0YHG9WtUr5lezGKw==
5
```

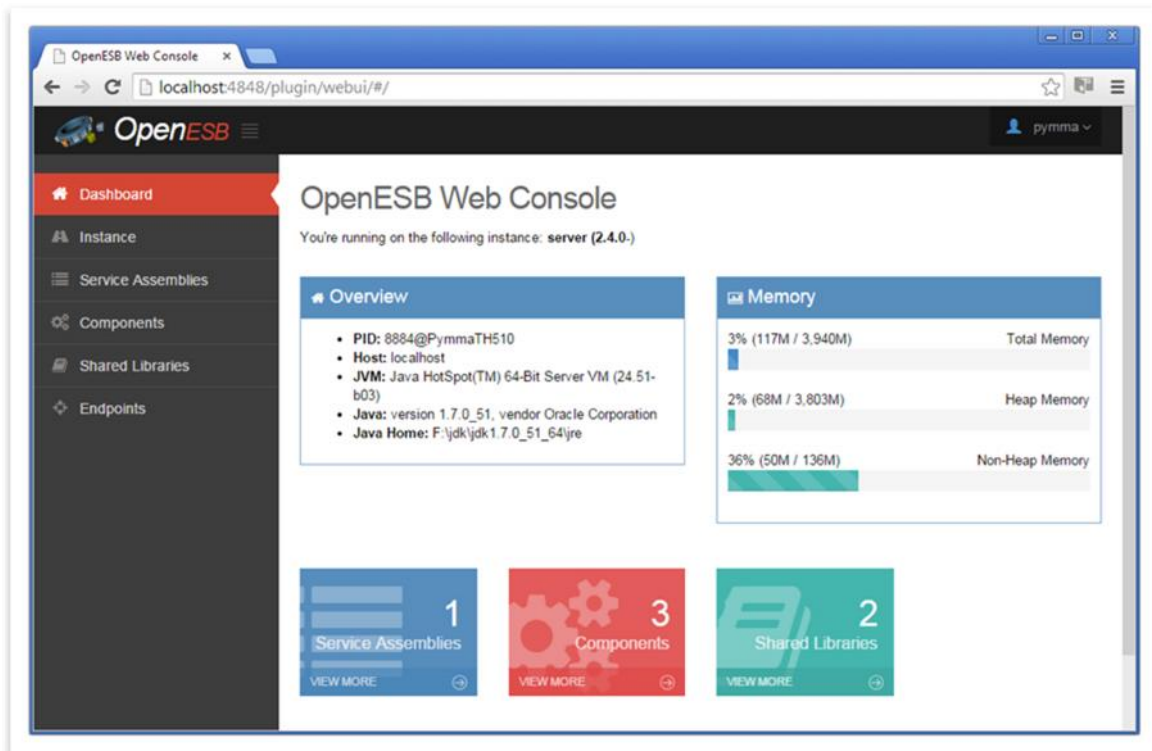
Save the file, reset OpenESB instance to take into account the new user. OpenESB offers an option to avoid an OpenESB reset. To do it, add the options reload and interval in the file `${OESE-HOME}\OE-Instance\config\openesb.yaml` as display below.

```
##### Security #####
# Set the security realms which have to be used for management purpose or components
# security concerns.
realm:
  # The realm "management" is the realm used for management (JMX / Rest API / oadmin)
  management:
    type: properties
    file: ${openesb.home}/config/mgmt-users.properties
    reload: true
    interval: 30 # seconds
```

3.3.2 Change a password

To change an existing password, replace the old password with the new one in the user management file (by default: `${OESE-HOME}\OE-Instance\config\mgmt-users.properties`).

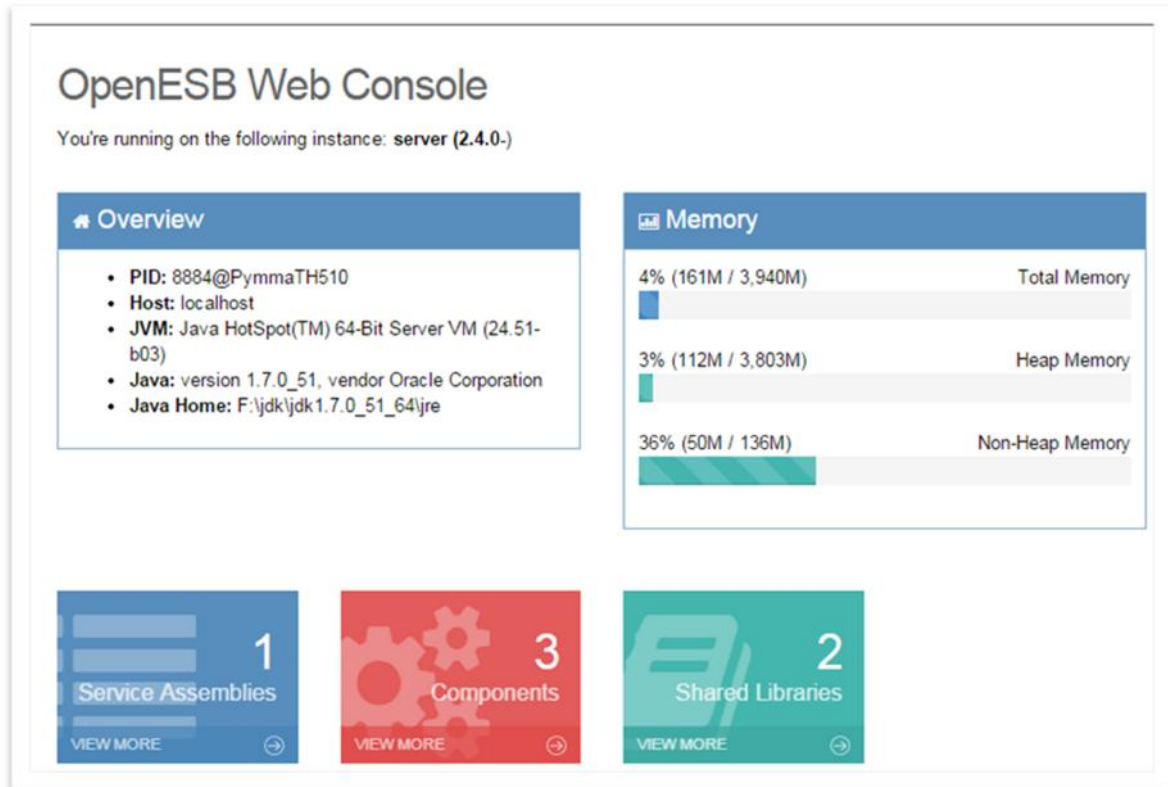
3.4 Console overview



The console is divided in two parts, the main menu on the left and the details on the right. On the top right user name is displayed, click on it to log out.

During the console design, we tried to keep it simple, straightforward and fast. You can use the web admin console to monitor OpenESB instances, install components and deploy services assemblies.

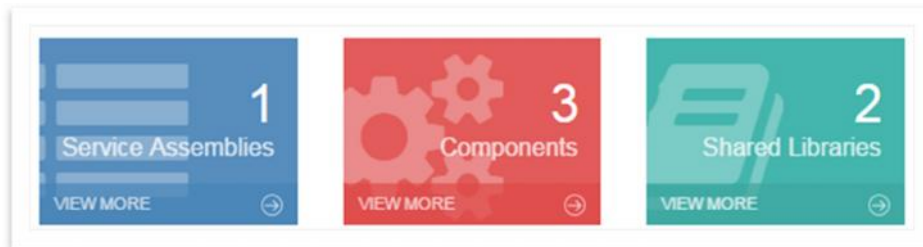
4 Dashboard



Overview		
Key	Value	Comment
Instance	Server (2.4.0)	OpenESB V3.0 runs with an instance version 2.4. The versions are not synchronised.
PID	8884@PymmaTH510	Instance unique ID. The machine where the instance runs is PymmaTH510
JVM	Java HotSpot(TM) 64-Bit Server VM (24.51-b03)	JVM version used to run the instance
Java	version 1.7.0_51, vendor Oracle Corporation	Java version used to run the instance
Java_Home	F:\jdk\jdk1.7.0_51_64\jre	Java Home defined in OE instance environment

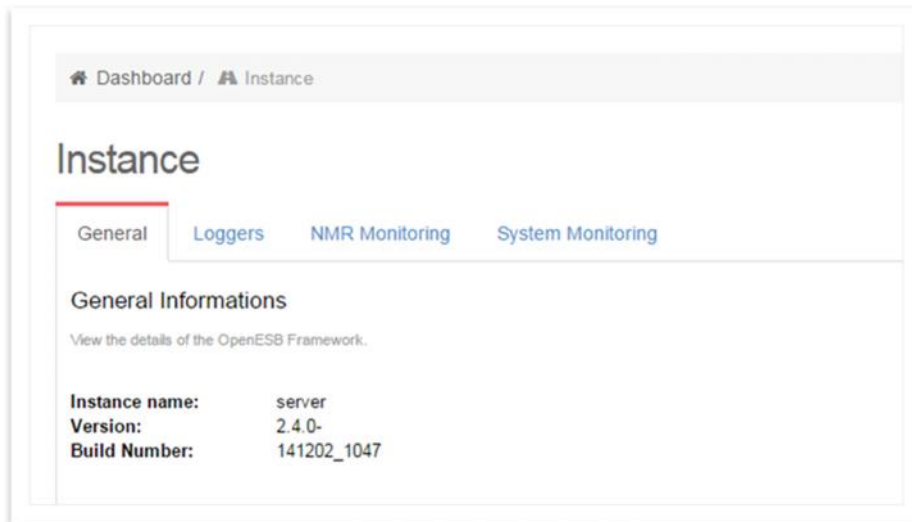
Memory	
Key	Explanation
Total Memory	Memory used by OE instance
Heap memory	Memory used to store Java objects used by OE instance
Non Heap Memory	Memory used to store loaded classes and other meta-data. JVM code

itself, JVM internal structures, loaded profiler agent code and data, etc.



These 3 pictures give you an overview of the installed elements in OE instances.

5 Instances



Instance screen provides more details of the instance itself.

5.1 General

General Informations		
Key	Value	Comment
Instance name	Server	“Server” is the instance default name. This name can be set up in the file <code>\${OESE-HOME}\OE-Instance\config\openesb.yaml</code> . You can give a more significant name to the instance such as “production”, “Finance”...
Version	2.4.0-	Instance version. Please note that OpenESB and Instance versions are not synchronised
Build Number	141202_1047	Build number for the instance

5.2 Logger

Name	Level									
com.sun.jbi.messaging	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management.LoggingService	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management.InstallationService	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management.DeploymentService	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management.ConfigurationService	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management.AdminService	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.framework	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT

OE instance offers a complete set of logger to monitor, trace and debug instance behaviour. A logger is associated with the classes which have the same package than the Logger name.

Each logger can be set individually.

- ALL: indicates that all messages should be logged.
- FINEST: Maximum verbosity
- FINER: Moderate verbosity
- FINE: Minimal verbosity
- CONFIG: Messages related to server configuration
- INFO: Messages related to server configuration or server status, excluding errors
- WARNING: Warnings, including exceptions
- SEVERE: Events that interfere with normal program execution
- DEFAULT: revert to the parent logger level or INFO if none
- OFF: is a special level that can be used to turn off logging

During development and test time, log level can be set from INFO to FINEST. Sometimes, FINER and FINEST are too verbose and can confuse log analysis.

For performance test and production, log level can be set from SEVERE to INFO. We recommend you to set up your loggers to INFO since higher levels can miss significant messages.

5.2.1 Set up log level

To set up a log level, just click on the selected level. You don't need to reset the instance to take into account a new log setting.

Instance

General **Loggers** NMR Monitoring System Monitoring

Manage Instance Loggers

Manage JBI Instance Loggers. Specify log levels for the individual loggers.

Name	Level									
com.sun.jbi.messaging	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management.LoggingService	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management.InstallationService	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management.DeploymentService	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management.ConfigurationService	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management.AdminService	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.management	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi.framework	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT
com.sun.jbi	ALL	FINEST	FINER	FINE	CONFIG	INFO	WARNING	SEVERE	OFF	DEFAULT

5.3 NMR Monitoring

NMR is the acronym of Normalized Message Router. The NMR can be seen as a bus where messages pass through (For more information on the NMR, have a look on JBI specifications). Many statistics are generated from the NMR and reported on this screen. Initial time for the statistics is the time OpenESB starts up. Statistics are reset when the instance is reset.

Instance

General Loggers **NMR Monitoring** System Monitoring

Monitor

Description	Value
Total number of message exchange accept operations	0
Total number of message exchange accept timeouts	0
Current rate of active exchanges per hour	0
Number of message exchanges currently active	0
Mean time to close a message exchange	0
Message exchange success rate	100
Total number of failed message exchanges	0
Total number of faulted message exchanges	0
Total number of InOnly message exchanges processed	0
Total number of InOptionalOut message exchanges processed	0
Total number of InOut message exchanges processed	0
Total number of RobustInOnly message exchanges processed	0
Total number of message exchange send operations	0
Total number of message exchange sendSync operations	0
Total number of transacted message exchanges	0

5.4 System Monitoring

Instance

General Loggers NMR Monitoring **System Monitoring**

System Monitoring

View the details of the system.

► Memory

Total Memory (324M / 3.940M)

8%

Heap Memory (274M / 3.803M)

7%

Non-Heap Memory (51M / 136M)

37%

► Threads (Total: 60)

Runnable (19)

32%

Timed waiting (12)

20%

Waiting (29)

48%

Blocked (0)

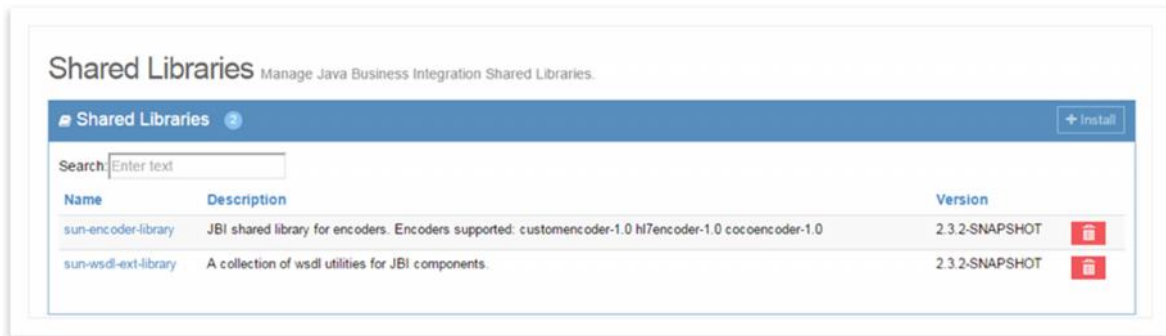
► Garbage collections

Mark Sweep count	1
Mark Sweep time	173ms
Scavenge count	7
Scavenge time	379ms

System monitoring displays additional statistic on the OE instance. These elements (Memory, Thread and Garbage Collecting) can be useful for optimising your instance and improving its scalability.

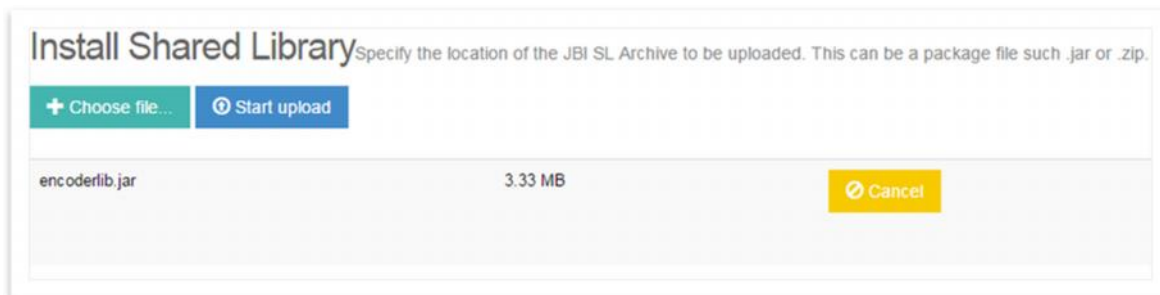
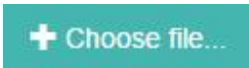
6 Shared Libraries

Shared libraries are Java libraries that can be shared by many components. The difference between a Java library and a shared library comes from a jbi.xml file added in the shared library jar file. For more details on jbi.xml, have a look at JBI specifications.



6.1 Install a shared library


To install a new shared library, click on the install button  then chose the file you want to install

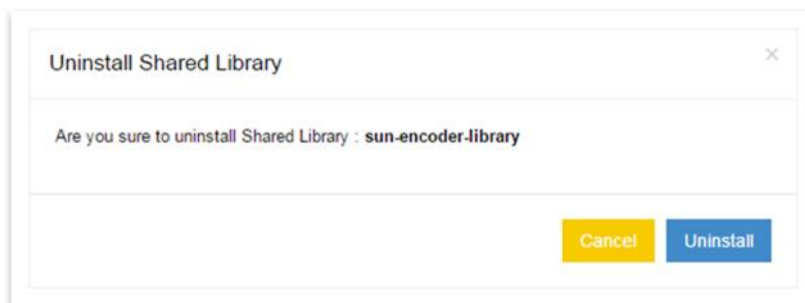


Once the file is chosen, click on  or  if you want to cancel the installation.

6.2 Uninstall a shared library

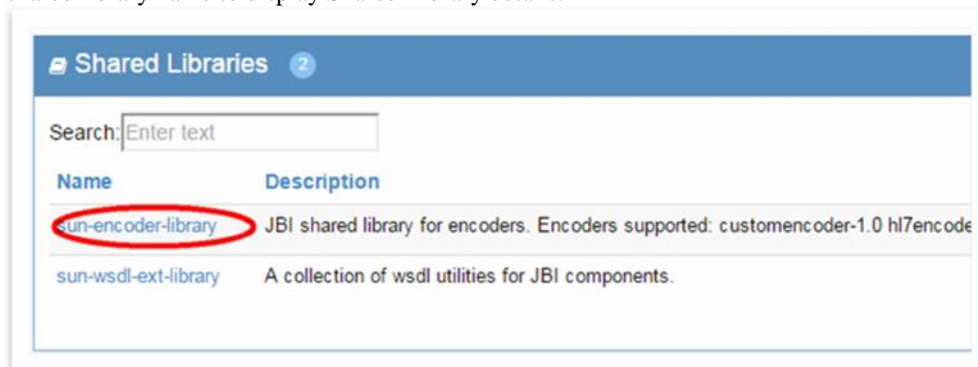


To uninstall a shared library, click on the associated bin icon  and confirm it.

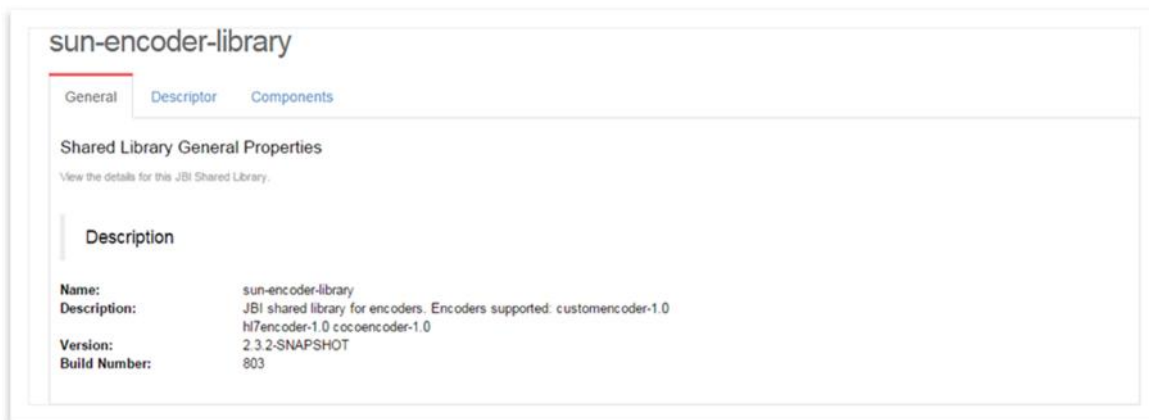


6.3 Shared library details

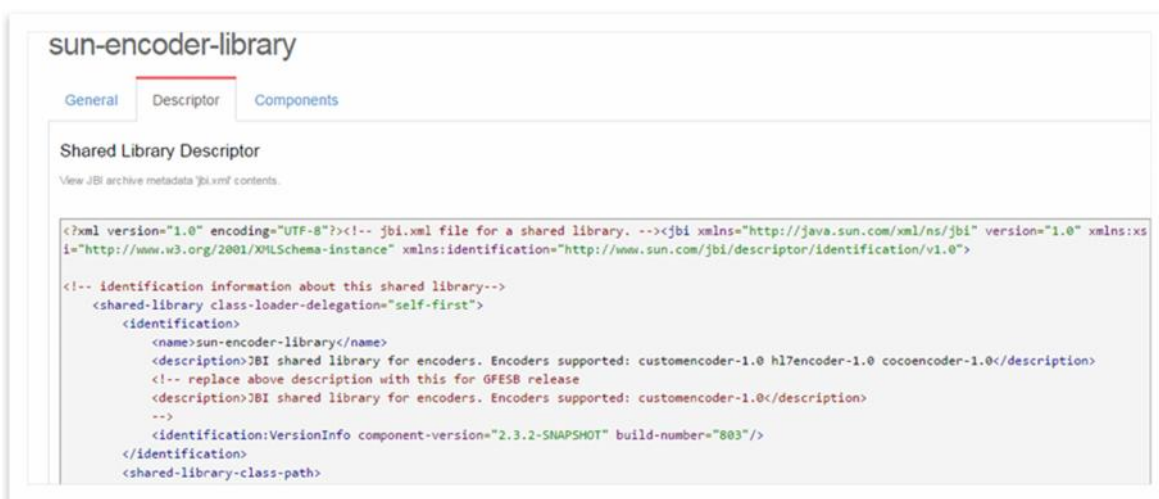
Click on a shared library name to display Shared Library details.



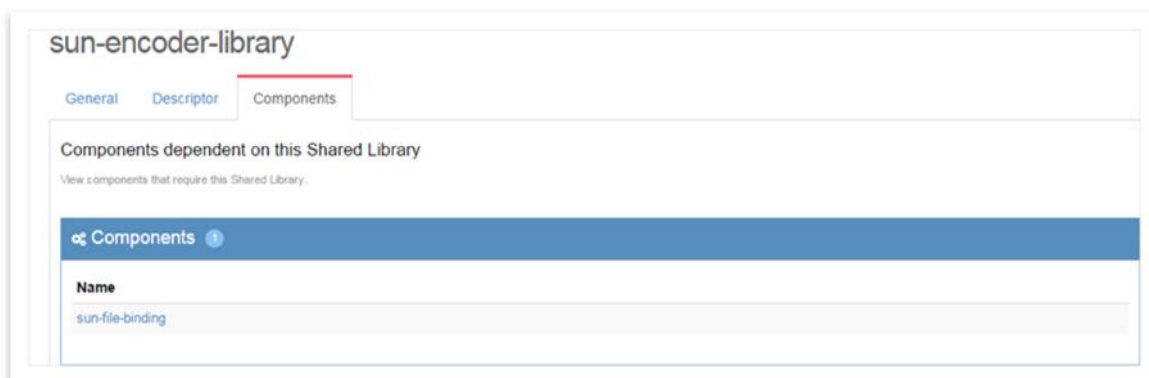
A detailed screen opens with three tabs: General, Descriptor and Components



The General screen provides General elements on the shared library such as Name or Version.



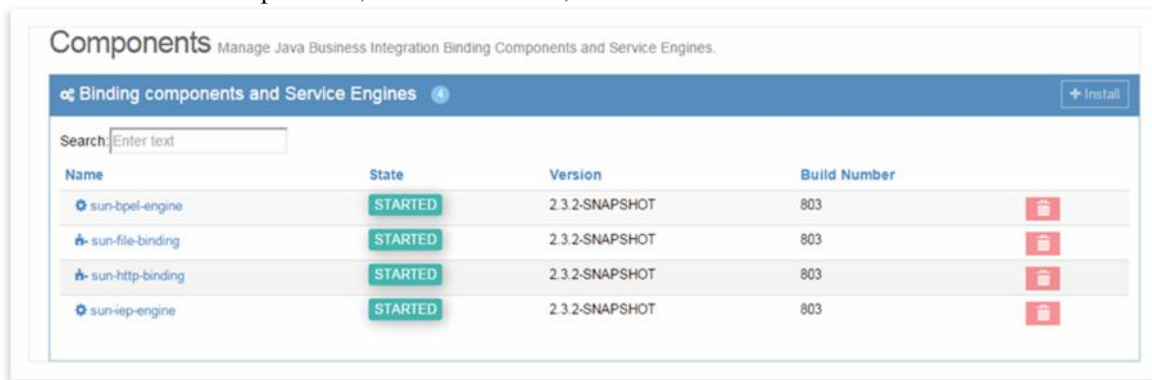
The Descriptor screen displays the content of the descriptor jbi.xml associated with the shared library.



The Components screen lists the components which use the shared library.

7 Components

OpenESB supports two kinds of components, Service Engines and Binding Components. OpenESB only distinguishes between the two kinds through a flag; the model and API are otherwise identical. However, by convention, Service Engines and Binding Components implement different functionality in OpenESB. Binding components are used to send and receive messages via particular protocols and transports. They serve to isolate the OpenESB environment from the particular protocol by providing normalisation and denormalisation from and to the protocol-specific format, allowing the OpenESB environment to deal only with normalized messages. Service Engines are the business logic drivers of the OpenESB. Engines can orchestrate service consumption and provision, in the course of, for example, executing long-lived business processes. Other engines can provide simple services, such as data transformation. Yet other engines may provide sophisticated routing or EDI services such as message collation / de-collation facilities. Service Engines can create new services by aggregating other services. SEs can serve as service providers, service consumers, or both.



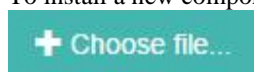
The Component main screen lists OpenESB components installed in an OE instance. Component state, Version and Build number are provided as well.

States can be:

Component states	
State	Description
Shutdown	The component is installed but does not consume any message for processing
Stop	The component is installed and does not consume any new message but process the current messages already consumed.
Start	The component is installed. It consumes new messages and processes them.
Unknown	This state indicates the console is not able to determine component state.

7.1 Install a new component

To install a new component, click on the install button  then chose the file you want to install



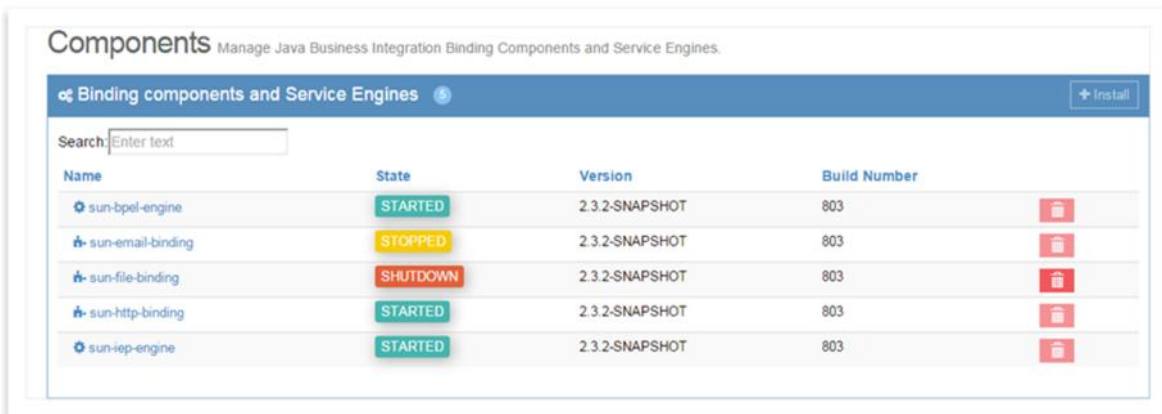



Once the file chosen, click on  or  if you want to cancel the installation.

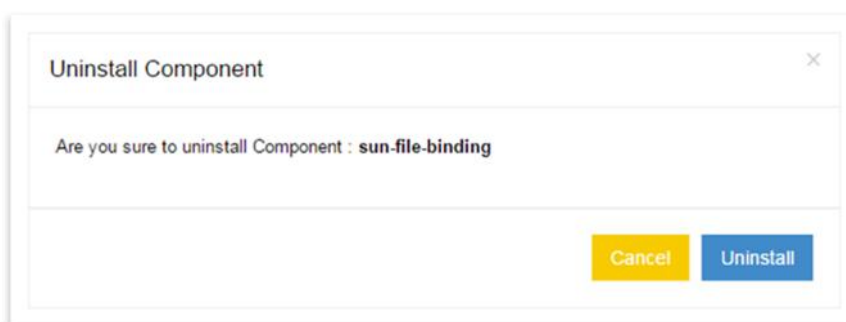
There is no difference between a Service engine and a Binding component installation.

7.2 Uninstall a component

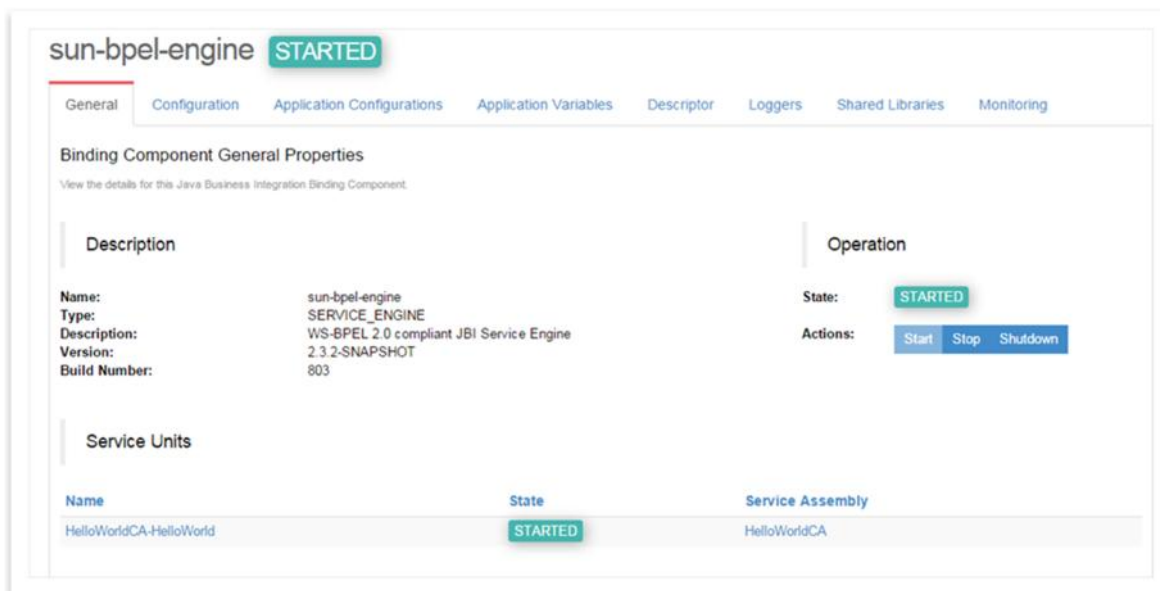
You can uninstall a component in the state shutdown only.



To uninstall a shutdown component, just click on the associated bin icon  and confirm it.



7.3 Component detail



Component detail screen displays 8 tabs

Tab	Comment
General	Provides general information on the component
Configuration	Configures dynamically the component. Available when the component is started.
Application configuration	Component Application Configurations are named collections of properties that allow Service Assemblies to configure a target component. There is no application configuration for Service Engine component.
Application Variable	Application variables allow you to define a list of variable names and values along with their type. The application variable name can then be used as a token in a WSDL extensibility element attribute for the component
Descriptor	Displays 'jbi.xml' contents associated with the component.
Logger	Manages component loggers. Each logger can be Set individually.
Shared libraries	Lists the shared libraries used by the component
Monitoring	Provides monitoring metrics such as active endpoints and message number.

7.3.1 General

General tab displays general information on the component such as: Name, Type, Version, Build number... This tab can be used to manage component state. (See Component life cycle chapter above).

Service Units		
Name	State	Service Assembly
ControlCustomerCA-sun-http-binding	STARTED	ControlCustomerCA
HelloWorldCA-sun-http-binding	STARTED	HelloWorldCA

General tab lists the Service Units associated with the component. For OpenESB new joiners, a Service Unit can be seen as a component configuration. An HTTP Service unit defines a port, an endpoint and interface for the HTTP component.

When a component state changes, the state of the Services Units linked with the component changes as well.

7.3.2 Configuration

Each component has its own configuration. Some can be very simple (Ex: file) but some more tricky to optimise (FTP, BPEL). It depends on component's complexity.

The screenshot shows the configuration page for the 'sun-file-binding' component. The component is in a 'STARTED' state. The 'Configuration' tab is active, showing a 'General Manage component' section. Two input fields are present: 'Number of Outbound Processor Threads' and 'Number of Inbound Worker Threads', both with the value '5'. A 'Save' button is located at the bottom of the configuration area.

The screenshot shows the configuration interface for the 'sun-ftp-binding' component. At the top, the component name 'sun-ftp-binding' is displayed next to a green 'STARTED' status indicator. Below this, there are navigation tabs: 'General', 'Configuration' (which is selected), 'Application Configurations', 'Application Variables', and 'De...'. The main content area is titled 'Configuration' and includes a note: 'Refer to the component's documentation for its runtime configuration properties.' Underneath, it says 'General Manage component'. The configuration parameters are as follows:

Number of Outbound Threads	10
Time out for Invoke	1000000
Use Passive FTP	<input checked="" type="radio"/> True <input type="radio"/> False
Use Proxy	<input type="radio"/> True <input checked="" type="radio"/> False
Proxy URL	socks5://localhost:1080
Proxy User ID	
Proxy User Password	
Connection Pool Minimum Size	2

Please refer to the component configuration guide for more details on the component. The configuration setup can be done only when the component has been started.

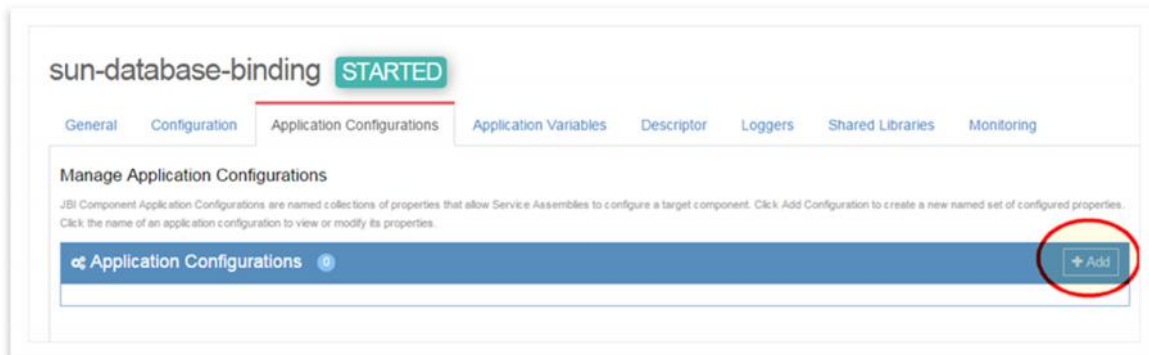
7.3.3 Application configuration

Application Configurations allow you to configure the external connectivity parameters for a JBI application and, without changing or rebuilding the application, deploy the same application to a different system. For example, if you have an application that is running in a test environment, you can deploy it to a production environment using new connectivity parameters without rebuilding the application.

The connectivity parameters for OpenESB Binding Component are normally defined in the WSDL service extensibility elements. When you create and apply application configurations for these parameters, the values defined for the application configuration override the values defined in the WSDL elements. You apply the configurations to the Composite Application by entering the application configuration name in the Config Extension Name property for the appropriate endpoint in the Service Assembly.

For more detail, please read our document: **770-006: OpenESB Multiple environments**

7.3.3.1 Create an application configuration



Select the tab Application Configurations and click on the Add button. Each binding component has its own configuration and the form to fill to create a configuration is different for each component.

A screenshot of a modal window titled 'Add new Application Configuration'. The subtitle reads 'Add a new Application Configuration definition for the selected component.' The form is divided into two sections. The first section, 'Identification', contains a text input field labeled 'Application Configuration Name'. The second section, 'General Properties', contains a text input field labeled 'Database JNDI Name'. At the bottom right of the form, there are two buttons: a yellow 'Cancel' button and a blue 'Save' button.

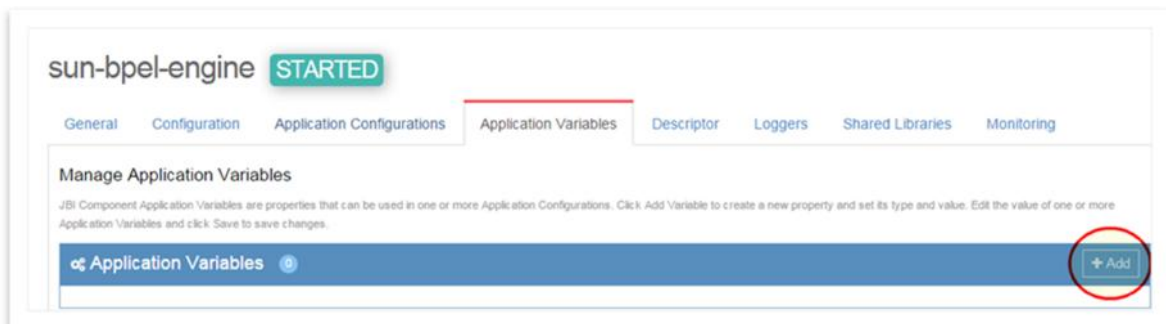
Database BC Application Configuration form

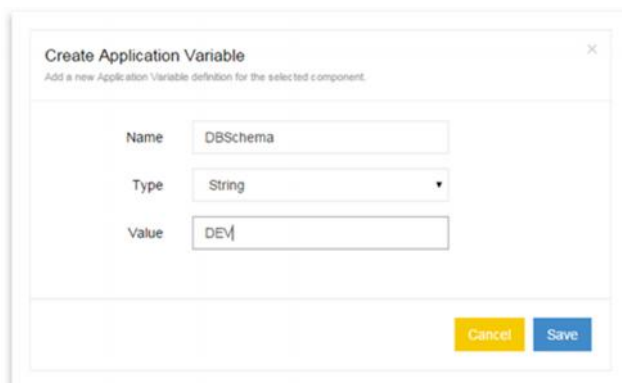
FTP BC Application Configuration form

7.3.4 Application variables

Application variables allow you to define a list of variable names and values along with their type. The application variable name can then be used as a token in a WSDL extensibility element attribute for the component. For example, you could define a string variable named **ServerName** with a value of **MyHost.com**. To reference this in the WSDL document, you would enter **\${ServerName}**. When you deploy an application that uses application variables, any variable that is referenced in the application's WSDL document is loaded automatically. For more detail, please have a look on our document: 770-006: OpenESB Multiple environments

7.3.4.1 Create Application variable





Create Application Variable X

Add a new Application Variable definition for the selected component.

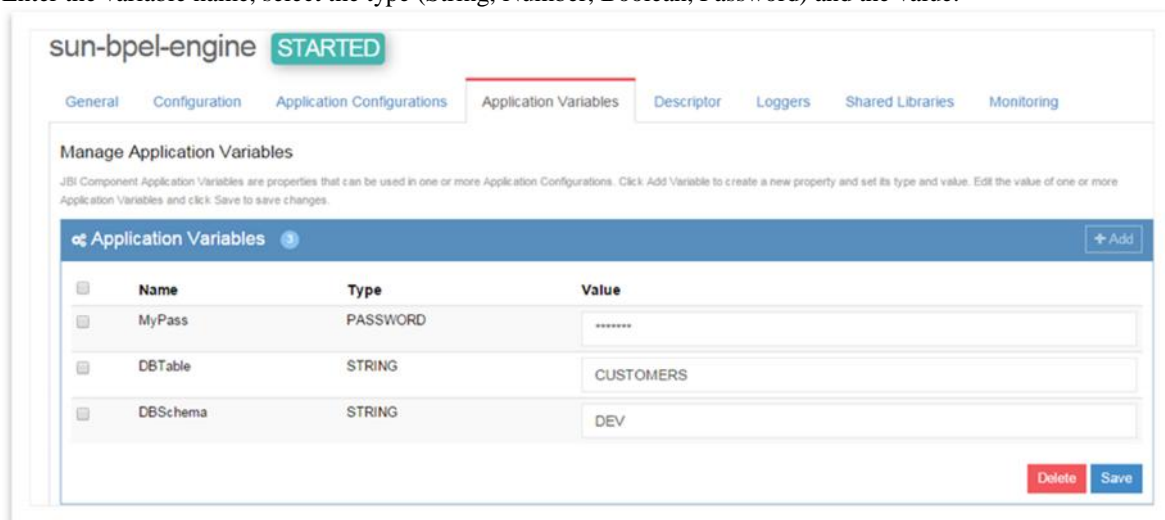
Name:

Type:

Value:

Cancel Save

Enter the variable name, select the type (String, Number, Boolean, Password) and the value.



sun-bpel-engine STARTED

General Configuration Application Configurations **Application Variables** Descriptor Loggers Shared Libraries Monitoring

Manage Application Variables

JBI Component Application Variables are properties that can be used in one or more Application Configurations. Click Add Variable to create a new property and set its type and value. Edit the value of one or more Application Variables and click Save to save changes.

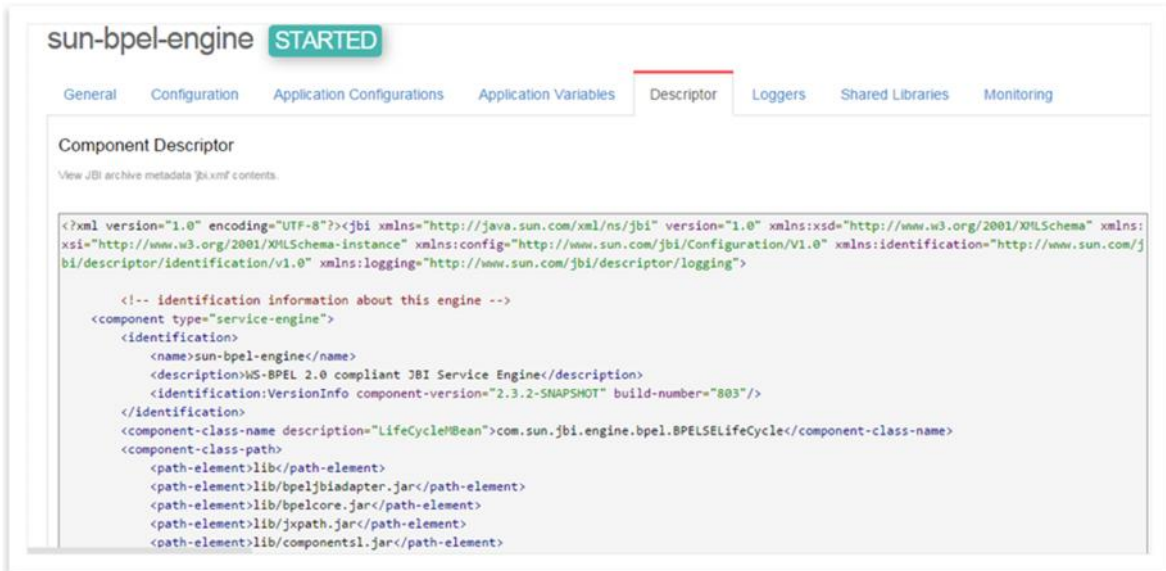
☰ Application Variables + Add

Name	Type	Value
MyPass	PASSWORD	*****
DBTable	STRING	CUSTOMERS
DBSchema	STRING	DEV

Delete Save

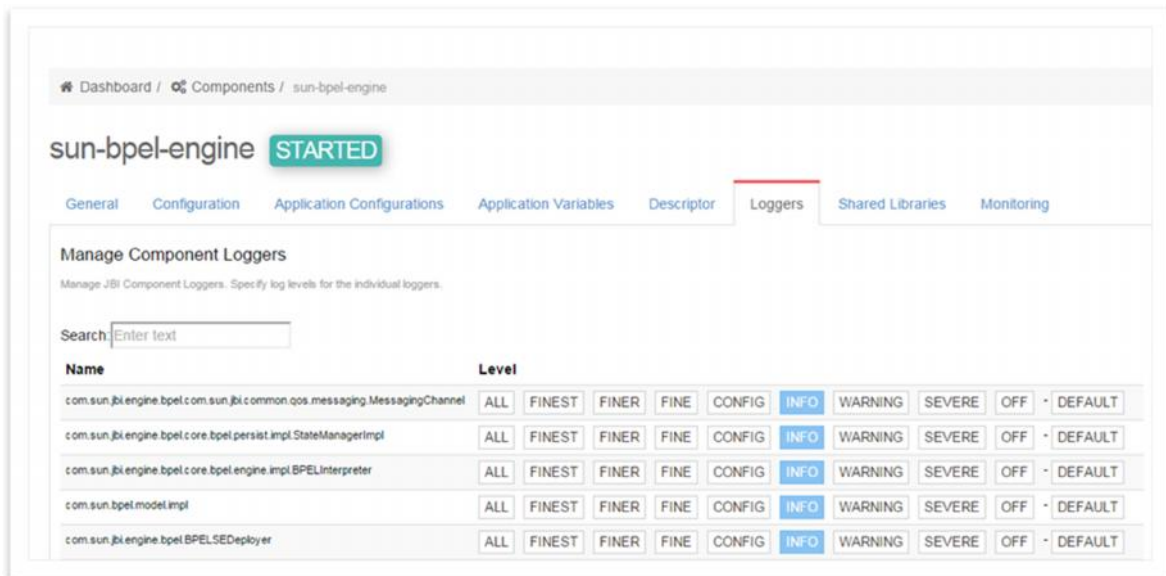
Values can be changed directly in the list.

7.3.5 Descriptor



Descriptor tab displays the component descriptor jbi.xml. It cannot be modified from the console.

7.3.6 Loggers



The Logger tab is used to set up component loggers. They are numerous and provide complete information on component behaviour.

Each logger can be set individually.

- ALL: indicates that all messages should be logged.

- **FINEST:** Maximum verbosity
- **FINER:** Moderate verbosity
- **FINE:** Minimal verbosity
- **CONFIG:** Messages related to server configuration
- **INFO:** Messages related to server configuration or server status, excluding errors
- **WARNING:** Warnings, including exceptions
- **SEVERE:** Events that interfere with normal program execution
- **DEFAULT:** revert to the parent logger level or INFO if none
- **OFF:** is a special level that can be used to turn off logging

During development and test time, log level can be set from INFO to FINEST. FINER and FINEST are verbose and can confuse log analysis.

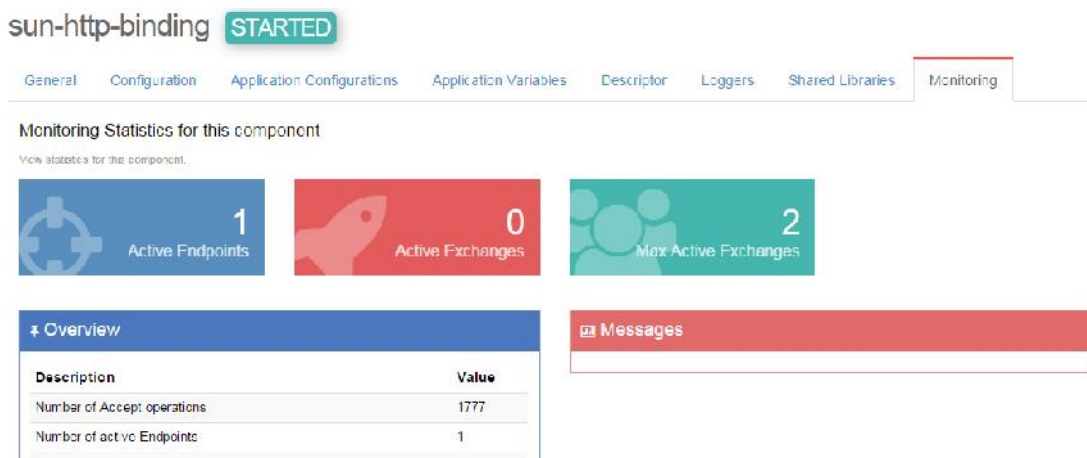
For performance test and production, log level can be set from SEVERE to INFO. We recommend you to set up the logger to INFO since higher level can miss significant messages.

7.4 Shared libraries



Shared libraries tab lists the shared library used by the component.

7.5 Monitoring



Monitoring provides metrics on the component. Message part of the screen will be used by the next versions.

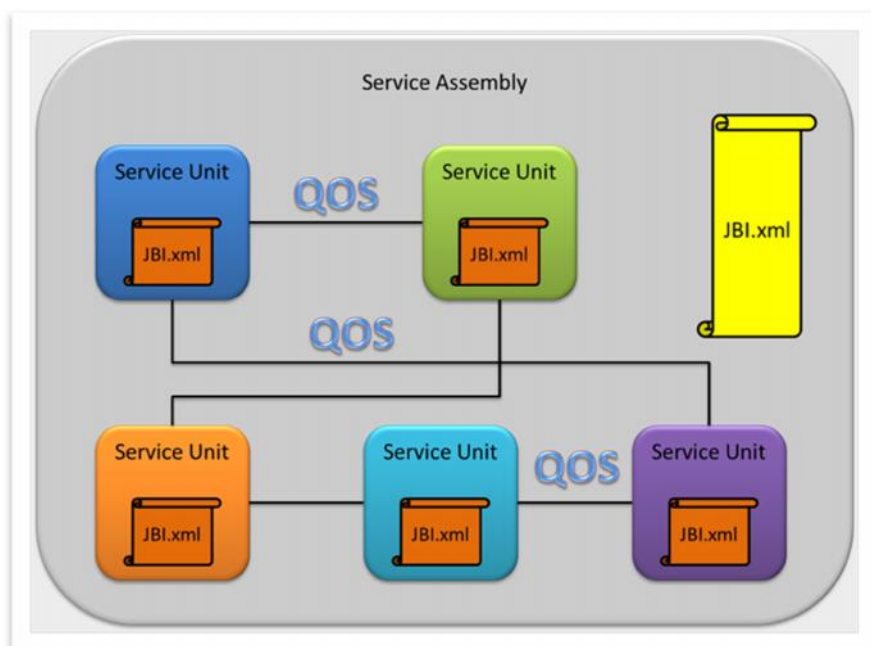
Description	Comments
Number of Accept operations	In each OpenESB component an endless loop check id a message is available for processing. This loop is named Accept Operation. It can be used to check if the component is alive.
Number of active Endpoints	Active point associated with the component
Number of active MessageExchanges	Message currently in process
Number of waits that didn't find any work	
Number of active MessageExchanges Max	Number of messages max processed parallel by the component
Number of queued MessageExchanges Max	Number of messages max waiting for component processing
Number of queued MessageExchanges	Number of messages waiting for component processing
Number of DONE requests received	Number of messages completely processed since the component started.
Number of ERROR requests received	Error means non happy path not defined by the contract of service. Ex: Java exception
Number of faults received	Fault means non happy path not defined by the contract of service.
Number of replies received	No comment
Number of requests received	No comment
Number of Send operations	No comment
Number of DONE requests sent	Number of requests successfully received by the addressee.
Number of ERROR requests sent	Error means non happy path not defined by the contract of service. Ex: Java exception
Number of faults sent	Fault means non happy path not defined by the contract of service.
Number of replies sent	No comment
Number of requests sent	No comment
Number of SendSync operations	Number of requests send in a synchronous mode

8 Services assemblies

A service assembly is the unit of deployment for OpenESB. You install components and shared libraries but you deploy service assemblies.

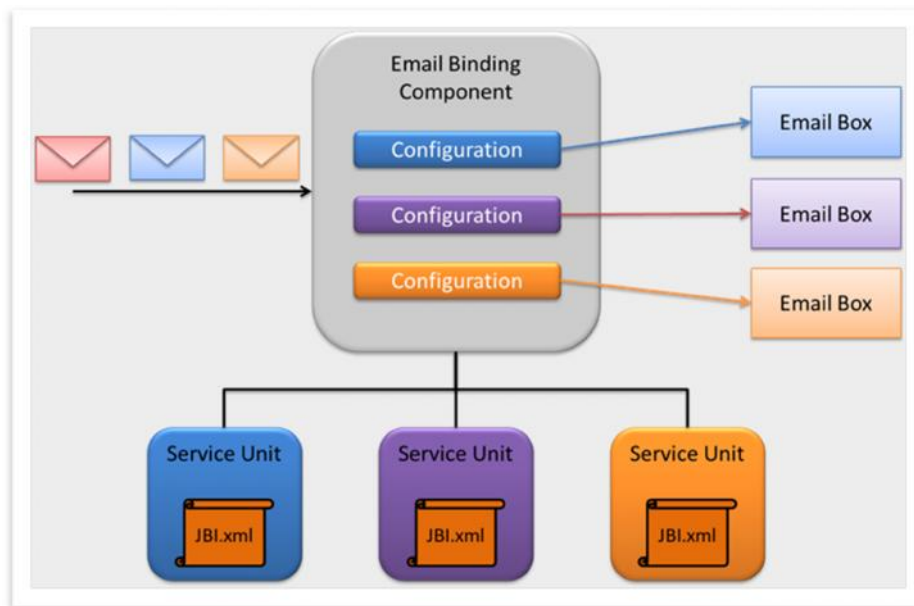
8.1 Service unit overview

Service assembly architecture is detailed below



A service assembly is the element to be deployed on OpenESB; it contains one or more service units. A service unit can be seen as a configuration for a component. The configuration is also named an endpoint. When the service unit is deployed, the component reads the service unit jbi.xml and creates a configuration by using jbi.xml parameters. So a component supports many endpoints or many configurations at the same time.

Let's take an example with Email Binding Component.



Email BC and Service Units configuration

During design time when creating Service Units, we associate them with a component type (Ex: BPEL or Email BC). Then during deployment, the component reads service unit configurations (jbi.xml), creates internal configurations and declares this configuration or endpoint to OpenESB.

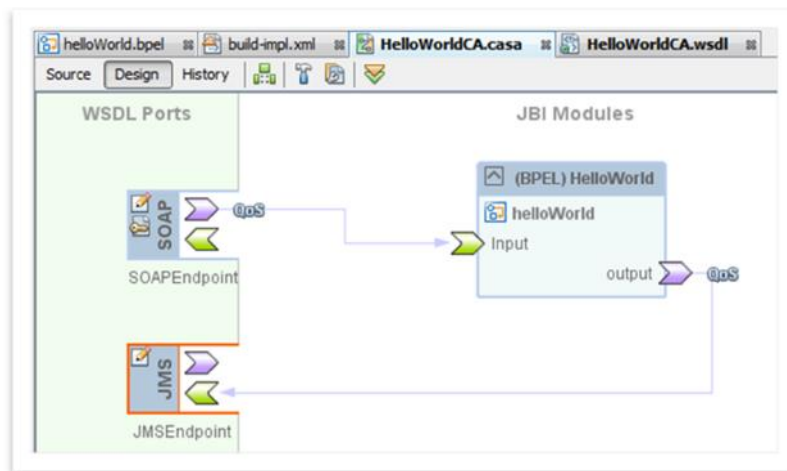
In the example above:

1. We deploy three service units linked at design time with Email BC. Each SU (service Unit) defines a connection to an email box.
2. Email BC creates 3 configurations and creates 3 connections to the email boxes and declare 3 endpoints to OpenESB.
3. A partner wants to send a message to an email box. In the message, a destination endpoint is added in the message header. Note that the partner does not know that Email BC deals with these endpoint but he just knows the endpoint that provides the email box services.
4. OpenESB knows the relationship between the endpoint and Email BC and routes the message to Email BC.
5. When Email BC receives a message, it introspects jbi.xml to find a relationship between the endpoint and the email box.
6. Then Email BC sends the message to the email box corresponding to the endpoint.

This mechanism is well described in JBI specifications.

8.2 Service assembly overview

Generally, an OpenESB application is made up with many service units.



The example above displays a simple service assembly with 3 services units.

1. SOAP SU
2. BPEL SU
3. JMS SU

OpenESB beginners frequently ask the question: Why do we have to deploy a service assembly and not the BPEL SU, the JMS SU and the SOAP SU separately. The reason is easy to understand. If you had to deploy a Service Unit separately, you would have to define its connection with the other SUs and the external world as well.

A service assembly is more than a set of service units. A service assembly manages the relationship (connections) between service units and between the service units and the outside world. This is simply the reason the unit of deployment is the Service Assembly.

Connections definitions are stored in the `jbi.xml` of the service assembly.

```

<connections>
  <connection>
    <consumer endpoint-name="casaPort1" service-name="ns1:CompositeApp3Service1"/>
    <provider endpoint-name="newWSDLPortTypeRole_myRole" service-name="ns2:PartnerLink1"/>
  </connection>
  <connection>
    <consumer endpoint-name="newWSDLPortTypeRole_partnerRole" service-name="ns2:PartnerLink2"/>
    <provider endpoint-name="casaPort2" service-name="ns1:CompositeApp3Service2"/>
  </connection>
</connections>



```

Connection set in JBI.xml

8.2.1 Deploy a new service assembly

Select Service Assemblies in the main menu.



Click on the button  then chose the file you want to deploy. 




Once you choose the file click on  or  if you want to cancel the deployment.

8.3 Undeploy a service assembly

You can undeploy a Service Assembly in a state shutdown only.



To undeploy a shutdown SA, just click on the associated bin icon  and confirm it.

8.4 Service assembly detail

8.4.1 General

The screenshot shows the 'HelloWorldCA' service assembly in a 'STARTED' state. The 'General' tab is active, displaying the service assembly's name, description, and state. The state is 'STARTED', indicated by a green button. Below the description, there are two service units listed in a table. The 'Operation' section shows 'Start', 'Stop', and 'Shutdown' buttons.

Name	State	Description	Target
HelloWorldCA>HelloWorld	STARTED	Represents this Service Unit	sun-bpel-engine
HelloWorldCA>sun-http-binding	STARTED	Represents this Service Unit	sun-http-binding

The General tab is used to manage the Service Assembly life cycle. To start, stop or shutdown the Service Assembly, just click on the relevant button.

The screenshot shows the 'HelloWorldCA' service assembly in a 'SHUTDOWN' state. The 'General' tab is active, displaying the service assembly's name, description, and state. The state is 'SHUTDOWN', indicated by a red button. Below the description, there are two service units listed in a table. The 'Operation' section shows 'Start', 'Stop', and 'Shutdown' buttons.

Name	State	Description	Target
HelloWorldCA>sun-http-binding	SHUTDOWN	Represents this Service Unit	sun-http-binding
HelloWorldCA>HelloWorld	SHUTDOWN	Represents this Service Unit	sun-bpel-engine

Name	State	Description	Target
HelloWorldCA-sun-http-binding	STOPPED	Represents this Service Unit	sun-http-binding
HelloWorldCA>HelloWorld	STOPPED	Represents this Service Unit	sun-bpel-engine

Note the service unit states are identical to the SA state. Nevertheless, it can happen in some cases that the console is unable to ascertain the state of a service unit. It is a sign that something wrong happened during SU deployment.

Name	State	Description	Target
ControlCustomerCA-sun-http-binding	STARTED	Represents this Service Unit	sun-http-binding
ControlCustomerCA-sun-file-binding	STARTED	Represents this Service Unit	sun-file-binding
ControlCustomerCA>ControlCustomer	UNKNOWN	This Represents the IEP Service Unit	sun-iep-engine

In the Service Unit list, note the target name is corresponding to the component that will deploy the service unit.

8.4.2 Service Unit overview

Name	State	Description	Target
HelloWorldCA>HelloWorld	STARTED	Represents this Service Unit	sun-bpel-engine
HelloWorldCA-sun-http-binding	STARTED	Represents this Service Unit	sun-http-binding



The descriptor tab displays the jbi.xml of the Service Unit.

8.4.3 Service assembly descriptor



The tab service assembly displays the file jbi.xml associated with the service assembly.

8.4.4 Service assembly monitoring

HelloWorldCA **STARTED**

General Descriptor **Monitoring**

Service Assembly Statistics

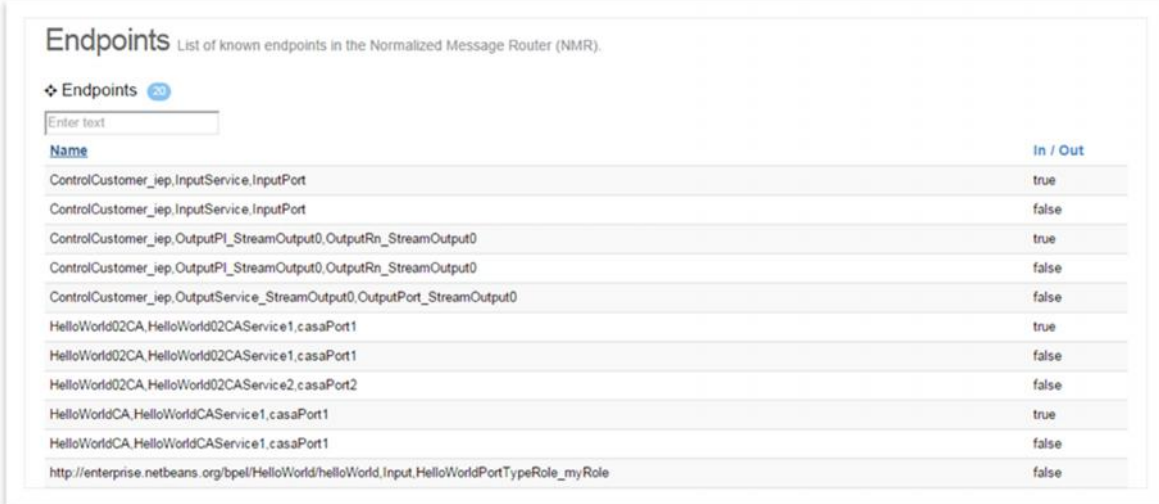
View the statistics for this Service Assembly.

Last startup time:	2015-01-02T11:31:23+00:00
Uptime:	0d 0h 18m 26s
Startup time:	0s 298ms

Monitoring the service assembly provides time statistic on the service assembly.

9 Endpoints

Provides the endpoint list registered in OpenESB.



The screenshot shows the 'Endpoints' page in the OpenESB web admin console. The page title is 'Endpoints' with a subtitle 'List of known endpoints in the Normalized Message Router (NMR)'. Below the title is a search bar labeled 'Enter text'. The main content is a table with two columns: 'Name' and 'In / Out'. The table lists various endpoints, including those for 'ControlCustomer_iep', 'HelloWorld02CA', and 'HelloWorldCA'. The 'In / Out' column indicates whether each endpoint is inbound (true) or outbound (false).

Name	In / Out
ControlCustomer_iep.InputService.InputPort	true
ControlCustomer_iep.InputService.InputPort	false
ControlCustomer_iep.OutputPI_StreamOutput0.OutputRn_StreamOutput0	true
ControlCustomer_iep.OutputPI_StreamOutput0.OutputRn_StreamOutput0	false
ControlCustomer_iep.OutputService_StreamOutput0.OutputPort_StreamOutput0	false
HelloWorld02CA>HelloWorld02CAService1.casaPort1	true
HelloWorld02CA>HelloWorld02CAService1.casaPort1	false
HelloWorld02CA>HelloWorld02CAService2.casaPort2	false
HelloWorldCA>HelloWorldCAService1.casaPort1	true
HelloWorldCA>HelloWorldCAService1.casaPort1	false
http://enterprise.netbeans.org/bpel/HelloWorld/helloWorld.Input>HelloWorldPortTypeRole_myRole	false

In/ Output indicates if the endpoint is an inbound or outbound endpoint.

Comment: Some endpoints names finish by the word “redeliveryLoopback”, for example:

<http://www.sun.com/jbi/qos/redelivery,sun-http-binding,redeliveryLoopback>.

These endpoints are generated by OpenESB for “Quality of Services” purposes.

10 What's next

For a better understanding of component and Service assembly, please have a look at JBI specifications.

For Application configurations and application variables, please have a look at our document 770-006 OE Multiple environment.

11 Help and support

11.1 From the community

You can find all our OpenESB documentations on the OpenESB official web site:
www.open-esb.net.

If you have any questions or would like to share your feedback, use the OpenESB forum at:
<http://openesb-community-forum.794670.n2.nabble.com>

Feel free to notify us with a bug or suggest how to improve our services on :
<https://openesb.atlassian.net/secure/Dashboard.jspa>

11.2 From Pymma

Pymma is deeply involved in the community and offers services and consulting on OpenESB. Pymma has professional services that can assist you from the development of your SOA design, implementation and ongoing management. All of our skills and background are based on our extensive first-hand experience and industry-leading methods.

Pymma releases an OpenESB Enterprise Edition with many additional enterprise features and a professional support.

In addition to OpenESB development, Pymma designed a new Service-Oriented development process named Rebecca to help business, architect and development team during the design and the implementation of their service oriented projects with OpenESB or any other service oriented development tool.

Feel free to contact us by email at contact@pymma.com for any further information on our OpenESB Services