OpenESB standalone edition Version 3.0

OpenESB set up in a multiple environments context. Application configurations and variables

Doc: 770-006: OE setup for multiple environments

Document identifier:

Pymma document: 770-006 OE setup for multiple environments

Location:

pymma.com

Editor:

Pymma Services: contact@pymma.com

Abstract:

This document explains how to use application configuration and variable to set OpenESB ready for multiple environments

Status:

Beta,

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

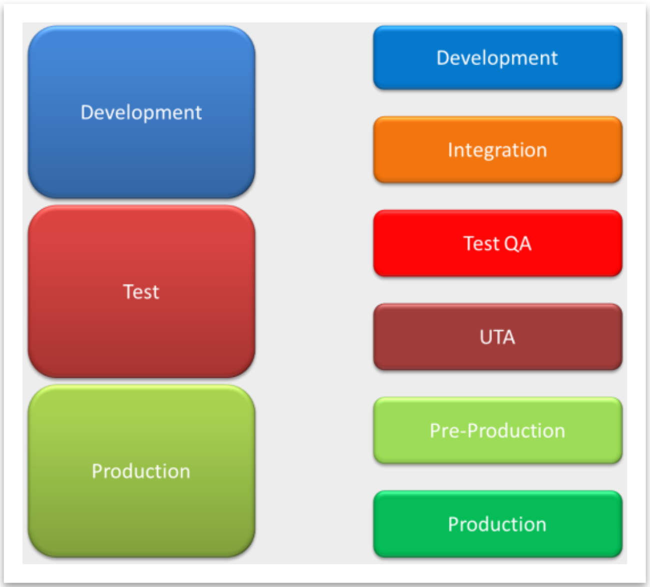
Disclaimer

Trademark Notice

Contents

1 Introduction

From the development to the production OpenESB applications go through many environments. Regarding their quality requirements and budgets, companies deploy many environments to control and validate their softwares.



IT Environments

At the development time, other environment parameters are not known just yet. By default, developers set up their applications with development environment parameters. Moving to another environment forces the applications to be rebuilt to match new environment parameters.

OpenESB offers 2 features to externalise environment parameters from the applications. They are named:

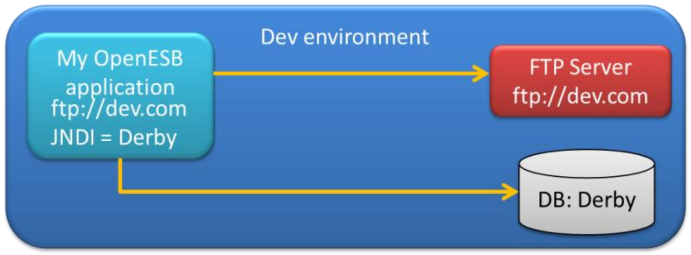
- Application configuration
- Application variable

The aim of this document is to show how to set up application configurations and variables, as well as to externalise environment parameters from OpenESB applications.

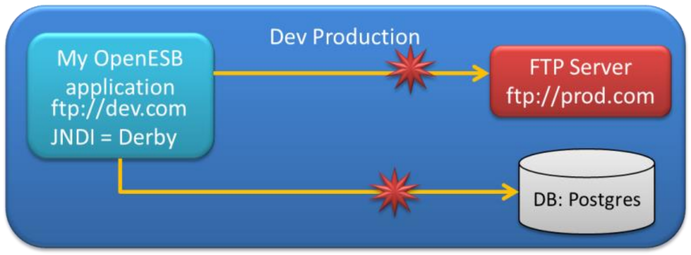
2 Application configuration

Application Configuration features allow you to externalize connectivity parameters from an OpenESB application without changing or rebuilding the application and deploy it to a different environment. So, an application developed and tested in an environment A can be deployed in an environment B by redefining the application configuration without rebuilding the application.

Let's take an example:



An OpenESB application has been developed in an environment with a Derby DB and an FTP server at the address ftp://dev.com. After passing the tests, the application is deployed in a production environment.


Dev Production
My OpenESB application ftp://dev.com JNDI = Derby
FTP Server ftp://prod.com
DB: Postgres

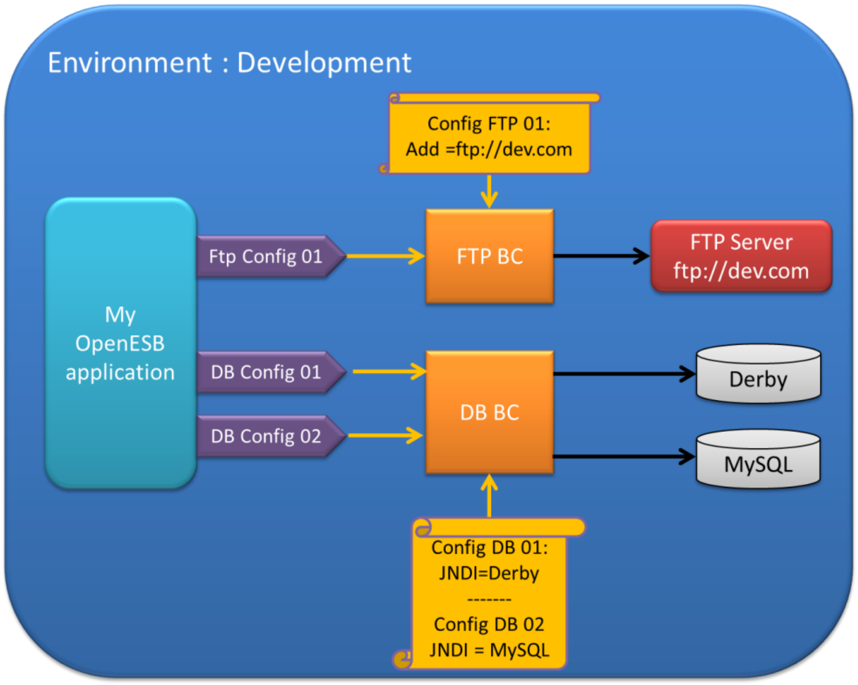
There, the database in production is a PostgreSQL DB and the FTP address is ftp://prod.com. Without parameters' externalisation, the application must be rebuilt to match the parameters of the new environment.

2.1 Application configurations general overview

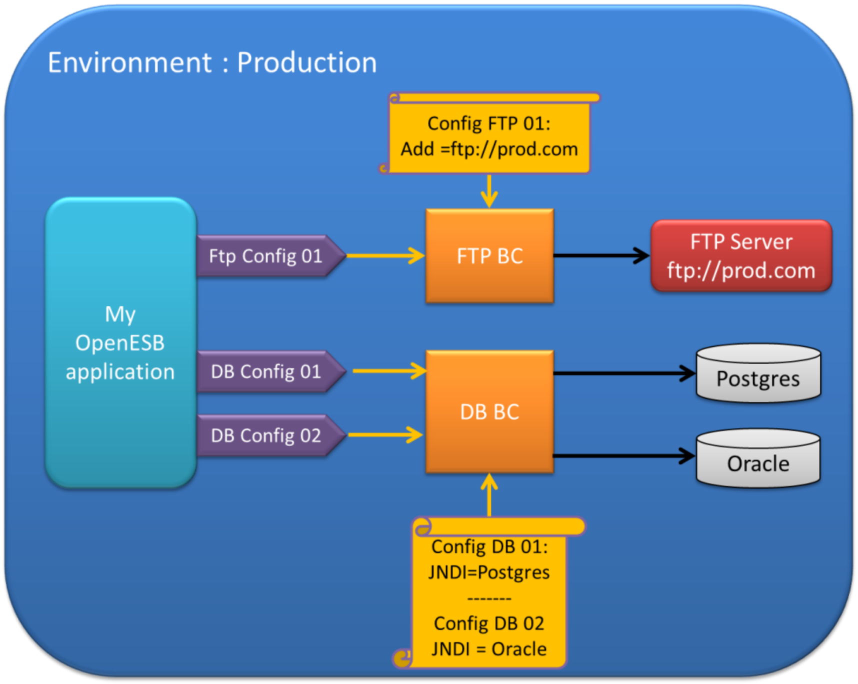
Two steps are required to set up the application configuration.

1. OpenESB administration level: Create the application configuration at the component level
2. OpenESB development level: Link an endpoint with a configuration.

Before starting, let's have a look on the diagrams below.

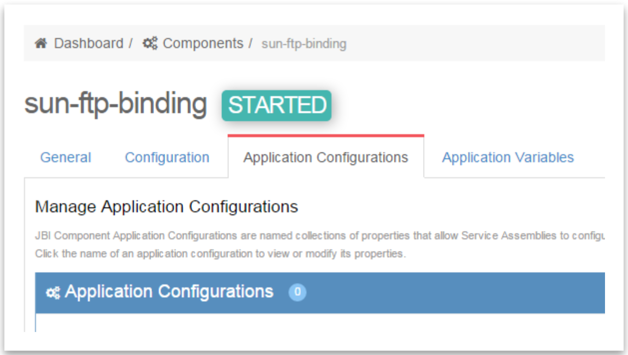


An application is developed in a development environment, with one FTP server (ftp://dev.com) and two databases Derby and MySQL. It must be transferred in the following environment with a FTP server with the address (ftp://prod.com) and two databases PostgreSQL and Oracle.



2.1.1 Set up application configurations in the development environment

OpenESB administrator must create one application configuration for FTP BC and two application configurations for Database BC (one per Database). To do it he uses the web admin console and adds three new configurations.


Dashboard / Components / sun-ftp-binding

sun-ftp-binding STARTED

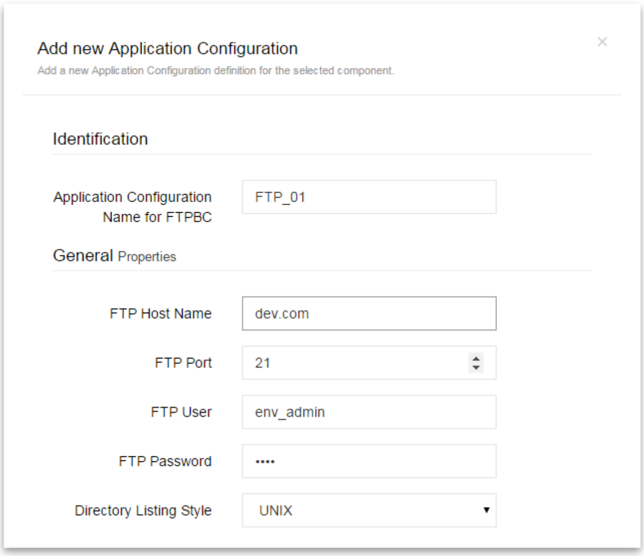
General | Configuration | Application Configurations | Application Variables

Manage Application Configurations

JBI Component Application Configurations are named collections of properties that allow Service Assemblies to configu
Click the name of an application configuration to view or modify its properties.

Application Configurations 0

OpenESB web admin console


Add new Application Configuration

Add a new Application Configuration definition for the selected component.

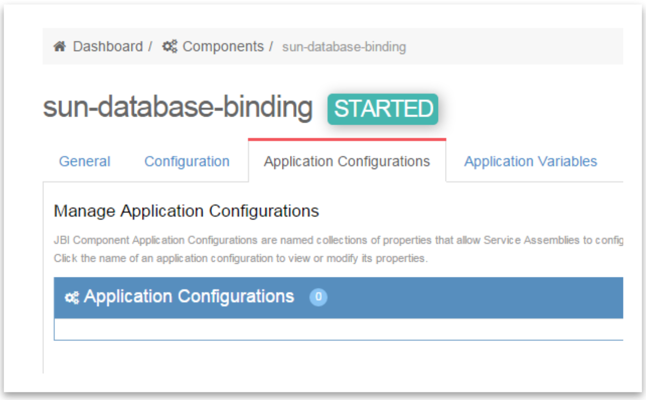
Identification

Application Configuration Name for FTPBC	FTP_01

General Properties

FTP Host Name	dev.com
FTP Port	21
FTP User	env_admin
FTP Password	••••
Directory Listing Style	UNIX

Application configuration TFP_01 in development


Dashboard / Components / sun-database-binding

sun-database-binding STARTED

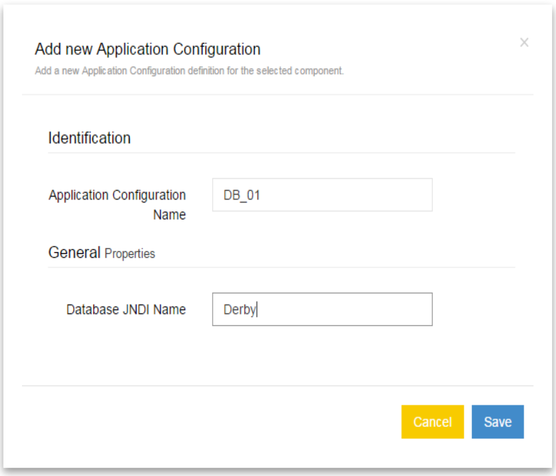
General | Configuration | Application Configurations | Application Variables

Manage Application Configurations

JBI Component Application Configurations are named collections of properties that allow Service Assemblies to config
Click the name of an application configuration to view or modify its properties.

Application Configurations 0

OpenESB web admin console


Add new Application Configuration

Add a new Application Configuration definition for the selected component.

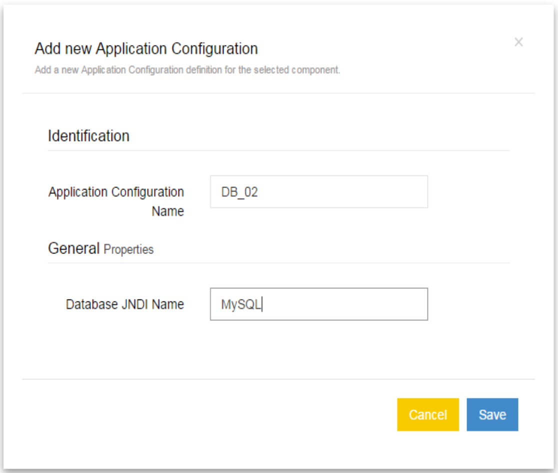
Identification

Application Configuration Name	DB_01

General Properties

Database JNDI Name	Derby

Cancel Save


Add new Application Configuration

Add a new Application Configuration definition for the selected component.

Identification

Application Configuration Name	DB_02

General Properties

Database JNDI Name	MySQL

Cancel Save

Application configurations for databases in development

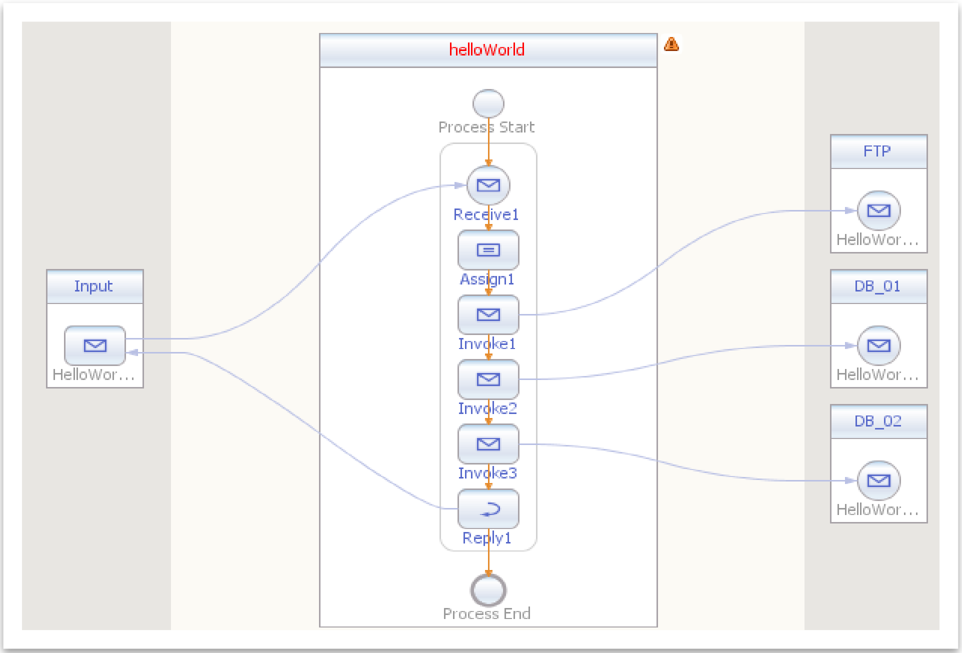
Database JNDI Names must be defined in the OpenESB instance context.xml. For more details, have a look at our document **770-005 OE JNDI Support**.

APPLICATION CONFIGURATIONS MUST BE DOCUMENTED AND COMMUNICATED TO THE DEVELOPERS.

2.1.2 Setting up the application configurations at development level

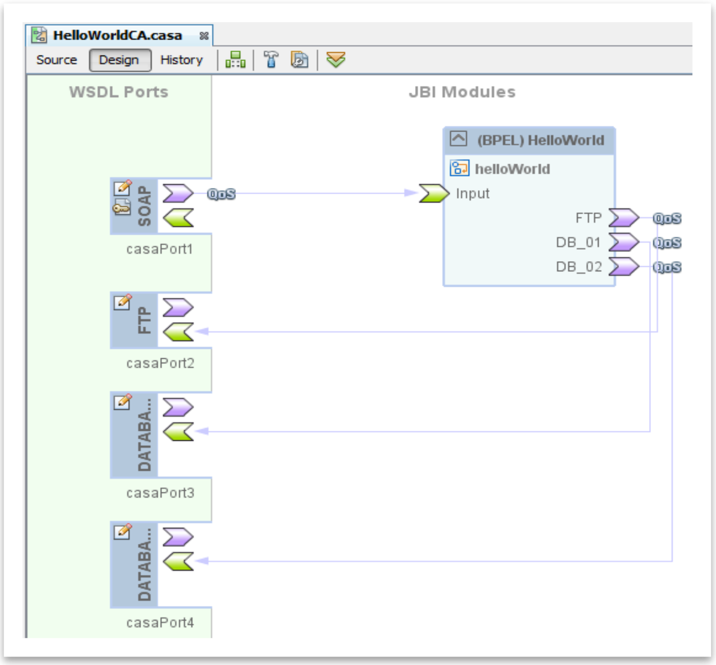
During the development, in the casa editor, you associate the composite application endpoints with binding components.

Let's take an example, a BPEL defines 4 endpoints.



1. Input endpoint
2. FTP Endpoint
3. DB_01 Endpoint
4. DB_02 endpoint

With the CASA editor, the developer associates endpoints with binding components.

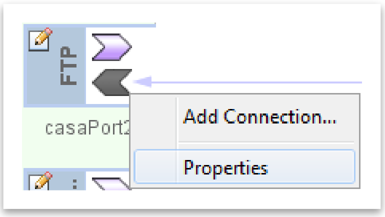


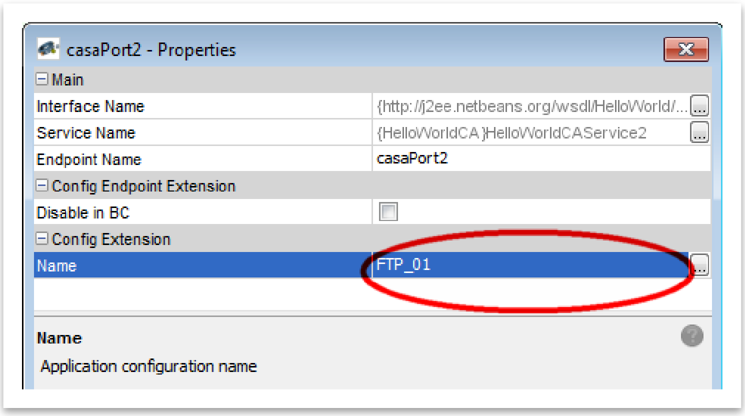
CASA Editor

1. Input endpoint is linked with SOAP BC
2. FTP endpoint is linked with FTP BC,
3. DB_01 endpoint is linked with Database BC,
4. DB_02 endpoint is linked with Database BC,

In this example, we set up application configurations for FTP, DB_01 and DB_02 endpoints.

Right click on the green arrow of FTP WSDL port, select properties and set up the configuration name with the value “FTP_01”



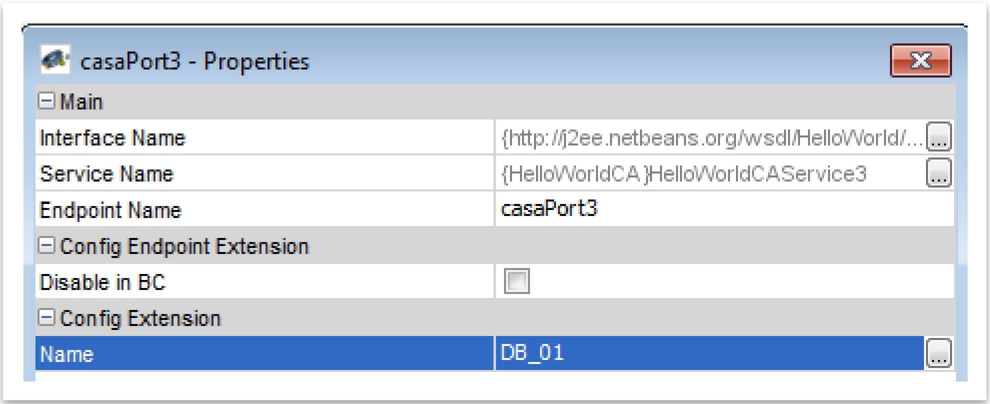


casaPort2 - Properties

Main	
Interface Name	{http://j2ee.netbeans.org/wsdl/HelloWorld/...
Service Name	{HelloWorldCA}HelloWorldCAService2
Endpoint Name	casaPort2
Config Endpoint Extension	
Disable in BC	☐
Config Extension	
Name	FTP_01

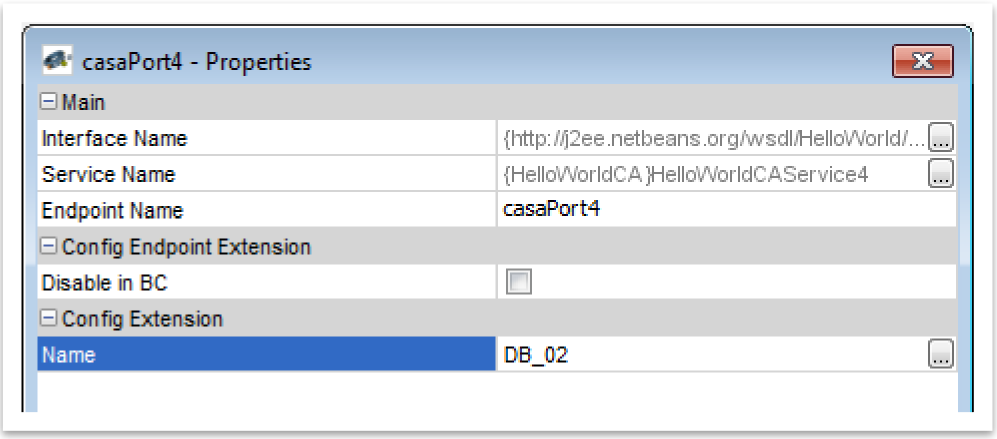
Name
Application configuration name

Do the process again for DB_01 and DB_02.



casaPort3 - Properties

Main	
Interface Name	{http://j2ee.netbeans.org/wsdl/HelloWorld/...
Service Name	{HelloWorldCA}HelloWorldCAService3
Endpoint Name	casaPort3
Config Endpoint Extension	
Disable in BC	☐
Config Extension	
Name	DB_01



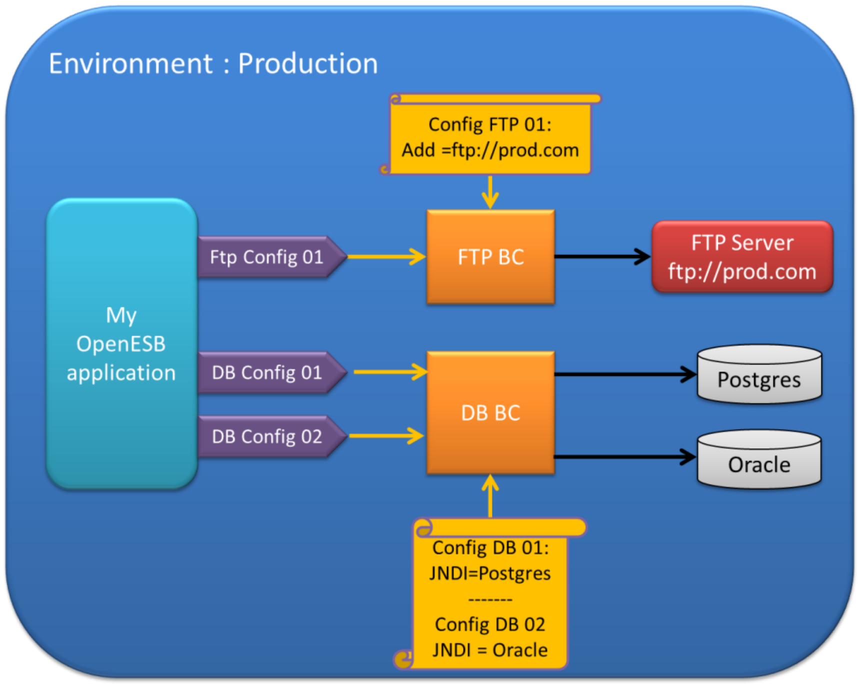
casaPort4 - Properties

Main	
Interface Name	{http://j2ee.netbeans.org/wsdl/HelloWorld/...
Service Name	{HelloWorldCA}HelloWorldCAService4
Endpoint Name	casaPort4
Config Endpoint Extension	
Disable in BC	☐
Config Extension	
Name	DB_02

Now the application setting is complete and you can run the application in the development environment.

2.1.3 Set up application configurations in the production environment

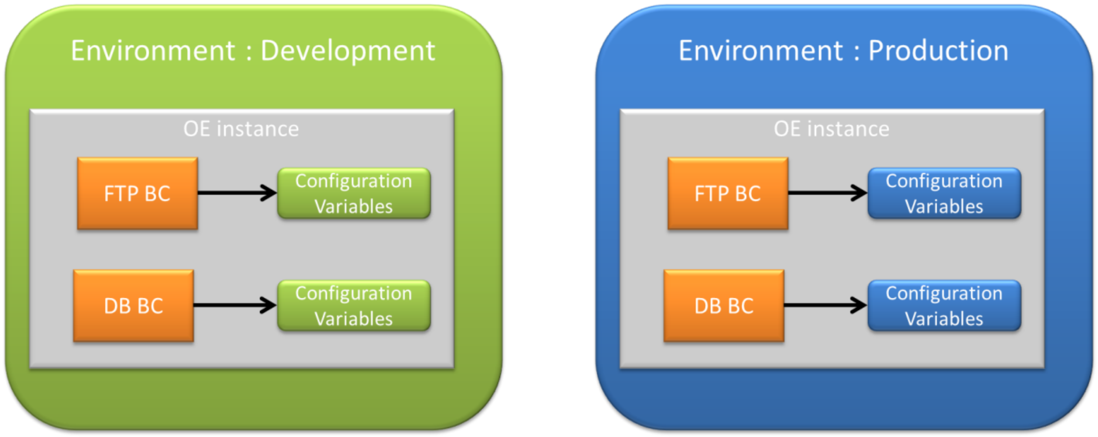
The production environment setup is similar and must be configured as follows:



FTP server address changes to ftp://prod.com and DB JNDI names have changed. Derby and MySQL databases have been replaced by PostgreSQL and Oracle.

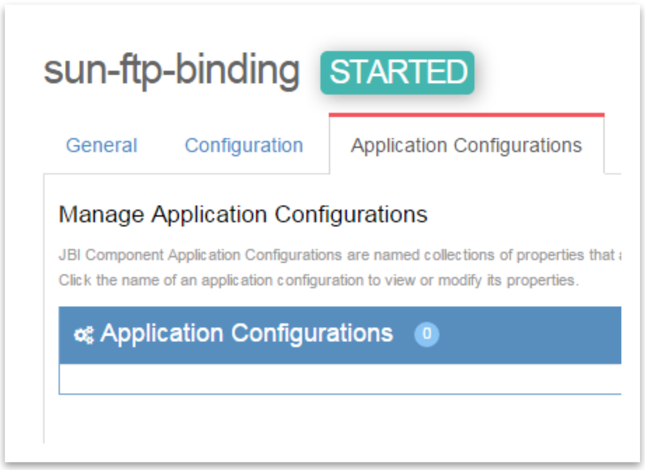
Configure FTP and DB binding components in the same way you did for the development environment. Create application configurations at the component level in the production environment.

Notice: To be sure you understand us, we precise to OpenESB beginners that in our use case, we are talking about 2 environments with an OpenESB instance independent of each other. So OpenESB component configurations in “Environment” are independent from Production’s ones (see picture below).



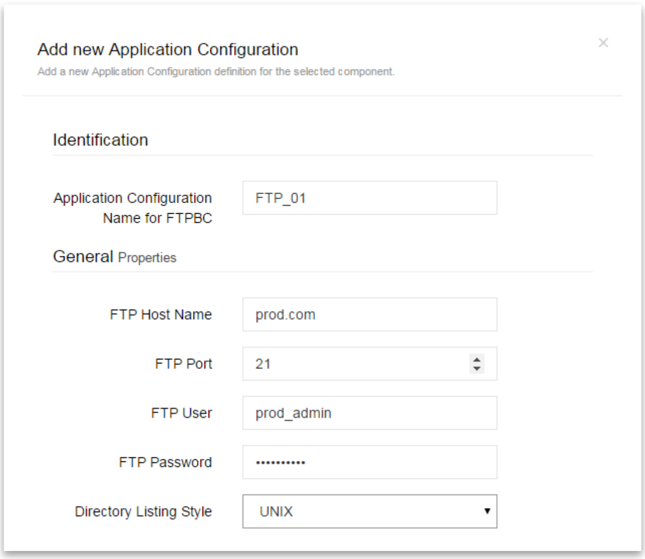
The administrator of the production environment creates new configuration variables at the component level.

Let's set up an FTP BC application configuration in “Production” and name it with the same name as in “Environment”: “FTP_01”



Manage Application Configurations

JBI Component Application Configurations are named collections of properties that
Click the name of an application configuration to view or modify its properties.



Add new Application Configuration

Add a new Application Configuration definition for the selected component.

Identification

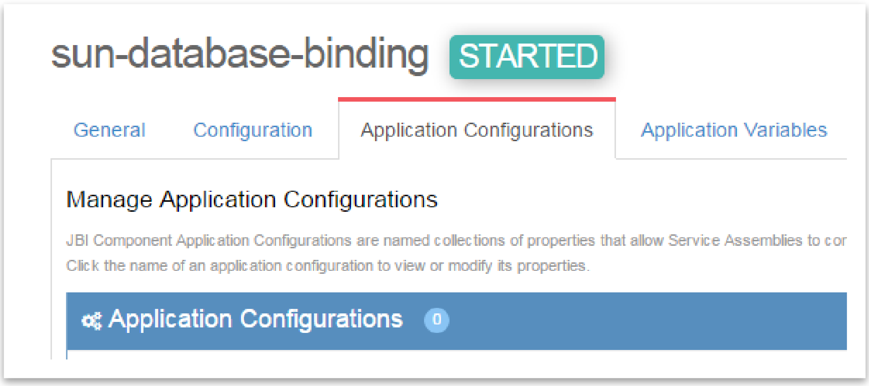
Application Configuration Name for FTPBC	FTP_01

General Properties

FTP Host Name	prod.com
FTP Port	21
FTP User	prod_admin
FTP Password	••••••••••
Directory Listing Style	UNIX

Application configuration TFP_01 for production

Let's set up two Database BC application configurations in "Production" and name them with the same names as in "Environment: "DB_01" and "DB_02".

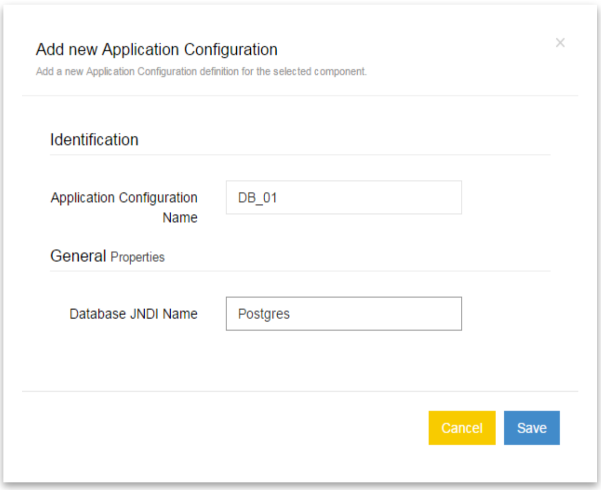


sun-database-binding STARTED

Manage Application Configurations

JBI Component Application Configurations are named collections of properties that allow Service Assemblies to cor

Click the name of an application configuration to view or modify its properties.


Add new Application Configuration

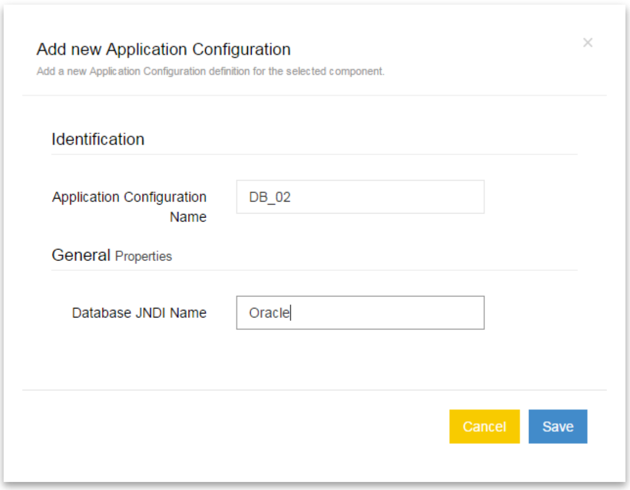
Add a new Application Configuration definition for the selected component.

Identification

Application Configuration Name: DB_01

General Properties

Database JNDI Name: Postgres


Add new Application Configuration

Add a new Application Configuration definition for the selected component.

Identification

Application Configuration Name: DB_02

General Properties

Database JNDI Name: Oracle

Application configurations for databases in production

New application configurations are setup with the values corresponding with the production environment. FTP address becomes ftp://prod.com and Database JNDI names: Postgres and Oracle.

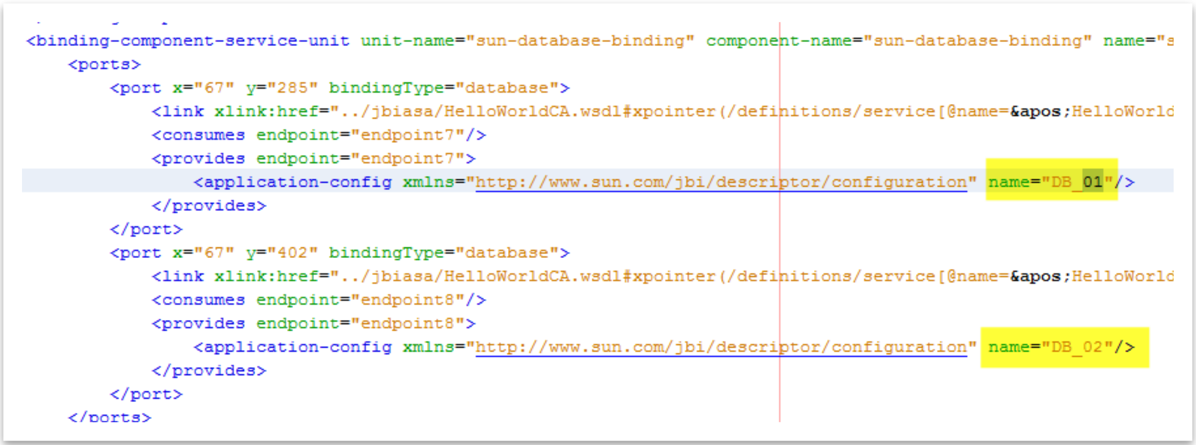
No change must be applied at the application level.

Application configuration feature allows you to externalize environment parameters from the application and then deploy it on many environments without any modification.

2.1.4 Linked between applications and components

How OpenESB creates the link between information provided in the CASA Editor and the application configurations defined at the component level.

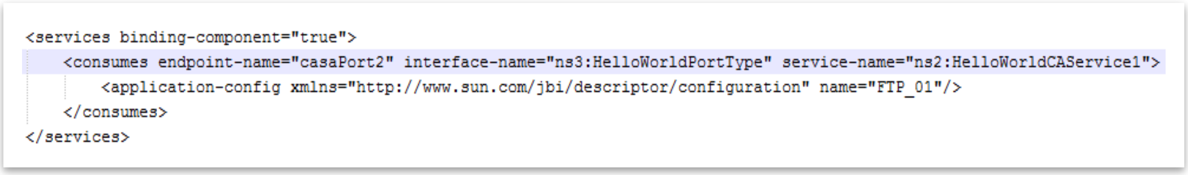
The answer is simple. In the CASA editor, the configuration you assign to an endpoint is stored in the CASA file.



```
<binding-component-service-unit unit-name="sun-database-binding" component-name="sun-database-binding" name="s
    <ports>
        <port x="67" y="285" bindingType="database">
            <link xlink:href="../jbiasa/HelloWorldCA.wsdl#xpointer(/definitions/service[@name=&apos;HelloWorld
            <consumes endpoint="endpoint7"/>
            <provides endpoint="endpoint7">
                <application-config xmlns="http://www.sun.com/jbi/descriptor/configuration" name="DB_01"/>
            </provides>
        </port>
        <port x="67" y="402" bindingType="database">
            <link xlink:href="../jbiasa/HelloWorldCA.wsdl#xpointer(/definitions/service[@name=&apos;HelloWorld
            <consumes endpoint="endpoint8"/>
            <provides endpoint="endpoint8">
                <application-config xmlns="http://www.sun.com/jbi/descriptor/configuration" name="DB_02"/>
            </provides>
        </port>
    </ports>
```

Composite Application WSDL Extract

When OpenESB builds the composite application to generate a service assembly, it adds the application configuration in the JBI.XML file.



```
<services binding-component="true">
    <consumes endpoint-name="casaPort2" interface-name="ns3:HelloWorldPortType" service-name="ns2:HelloWorldCAService1">
        <application-config xmlns="http://www.sun.com/jbi/descriptor/configuration" name="FTP_01"/>
    </consumes>
</services>
```

Service assembly jbi.xml extract

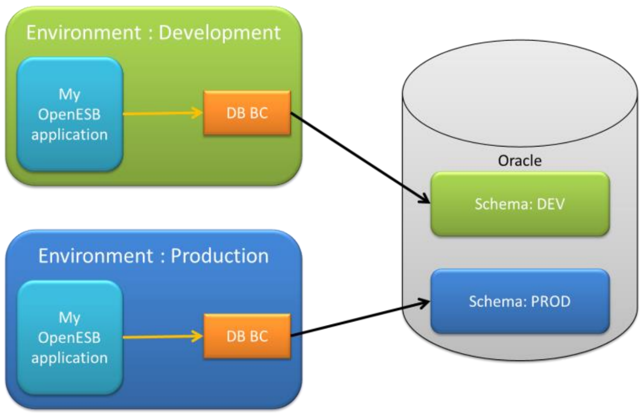
When the component loads the service assembly it generates an association between the endpoint and the configuration. End of the mystery!

3 Application variables

Application variables allow you to define a list of variable names and values with their types. The application variable name can then be used as a token in a WSDL extensibility element attribute for a Binding 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.

3.1 Example

Let's take a simple example with an Oracle Database that contains 2 schemas. The first named DEV is used for development and test purposes. The second is named PROD and used in production.



One schema per environment in the same database

In the first case (in development), when the developer wants to query the database and gets a list of customers living in London, he sends the following request:

Select * from DEV.CUSTOMERS where TOWN='LONDON'

In the second case (in production) the request becomes:

Select * from PROD.CUSTOMERS where TOWN='LONDON'

Without any additional features, OpenESB applications must be modified to take into account the production schema.

3.2 Define a variable at component level

In our example, we cannot rely on application configuration defined by Database BC to solve this issue since it only offers to set up JNDI names and doesn't take into account different database schema names.

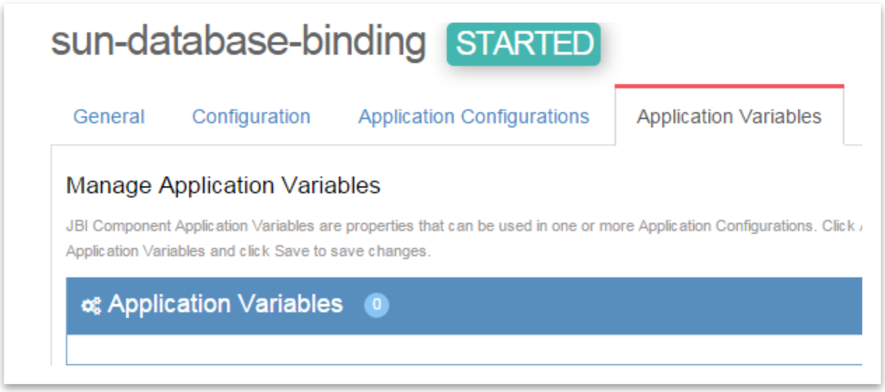
Application variable have been defined to solve this issue and externalize other environment parameters such as database schema.

To remove dependencies between OpenESB application and database schema names, we create an application variable with the environment schema name as value. A type must be defined for each application variable: OpenESB proposes 4 types for an application variable.

- String
- Number
- Boolean
- Password

A complex type cannot be used to define an application variable.

Use OpenESB wed admin console to set up application variables.

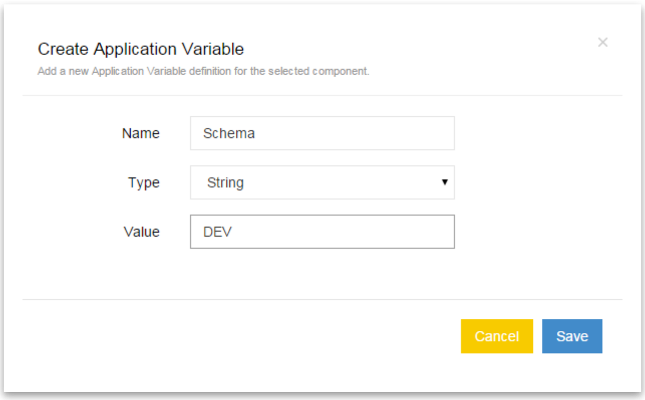


Manage Application Variables

JBI Component Application Variables are properties that can be used in one or more Application Configurations. Click
Application Variables and click Save to save changes.

3.2.1 Development environment

In the development environment, add an application variable to the Database Binding Component. Name it "schema", select the type String and give it the value "DEV".


Create Application Variable

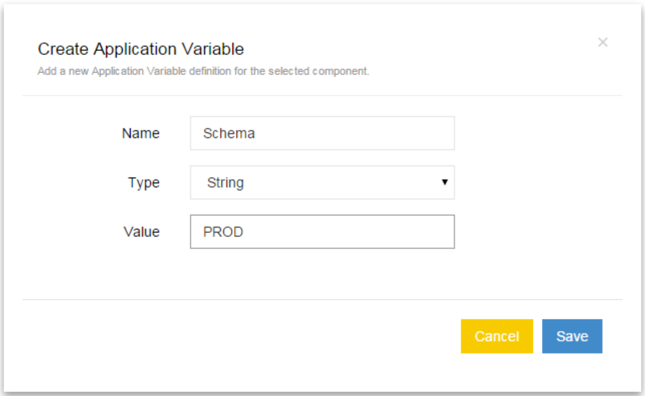
Add a new Application Variable definition for the selected component.

Name	Schema
Type	String
Value	DEV

Application variable Schema for development environment

3.2.2 Production environment

In the production environment, add an application variable to the Database Binding Component. . Name it “schema”, select the type String and give it the value “PROD”.


Create Application Variable

Add a new Application Variable definition for the selected component.

Name	Schema
Type	String
Value	PROD

Application variable Schema for production environment

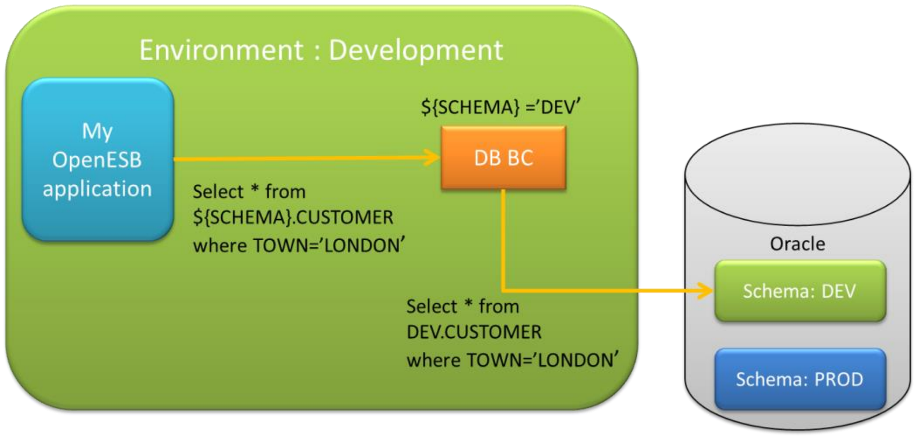
3.3 Application variables during design time

During design time, application variables are used in the concrete part of the WSDL linked with the components.

In the WSDL document our SQL request must be written as follows:

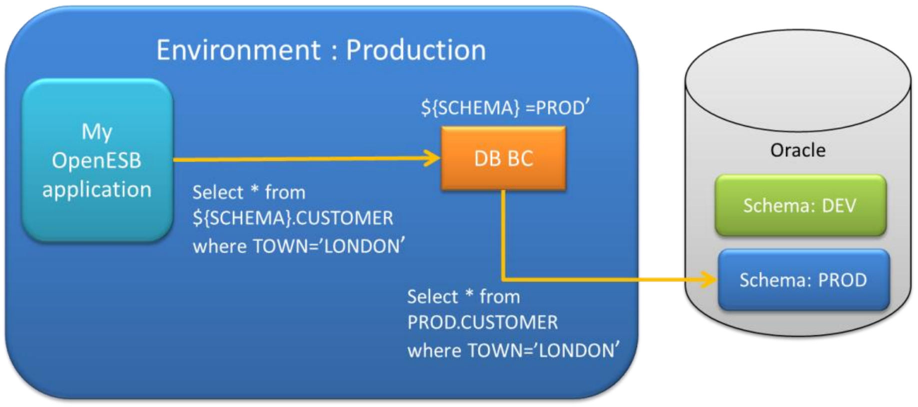
Select * from ${SCHEMA}.CUSTOMER where TOWN='LONDON'

3.4 What happens at the running time


Environment : Development
My OpenESB application
${SCHEMA} ='DEV'
DB BC
Select * from ${SCHEMA}.CUSTOMER where TOWN='LONDON'
Select * from DEV.CUSTOMER where TOWN='LONDON'
Oracle
Schema: DEV
Schema: PROD

Application variable process in development environment

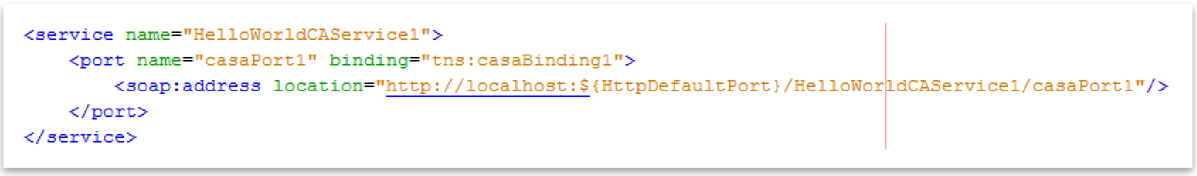
When an endpoint is deployed with a WSDL containing tokens (ex: **${SCHEMA}**), the component replaces the token with the value of the variable having the same name. The value is defined by the environment administrator.


Environment : Production
My OpenESB application
${SCHEMA} =PROD'
DB BC
Select * from ${SCHEMA}.CUSTOMER where TOWN='LONDON'
Select * from PROD.CUSTOMER where TOWN='LONDON'
Oracle
Schema: DEV
Schema: PROD

Application variable process in production environment

When the environment changes, variable values and endpoint definitions change as well, without additional development or rebuild.

Any parameter used in a concrete WSDL can be replaced by an application variable. The most used is the http default port setup in HTTP BC configuration.



```
<service name="HelloWorldCAService1">
    <port name="casaPort1" binding="tns:casaBinding1">
        <soap:address location="http://localhost:${HttpDefaultPort}/HelloWorldCAService1/casaPort1"/>
    </port>
</service>
```

HttpDefaultPort is a HTTP BC predefined (You don't have to define it) application variable.

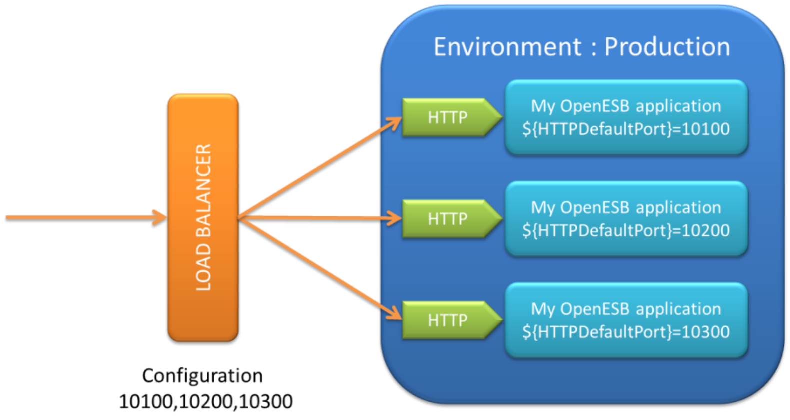
3.5 Predefined HTTP application variables (token)

HTTP BC predefines 2 application variables

1. "${HttpDefaultPort}" for the HTTP port
2. "${HttpsDefaultPort}" for the HTTPS port

These token names are used in lieu of a real port number in the endpoint URL (soap:address) to allow the application client to direct HTTP requests to the default port. The value of the token is then resolved by the HTTP Binding Component, based on the configured default values when an application is deployed.

These application variables can be set up for load balancing and high availability purposes (See our document: **770-008: OE multiple instances environment**).


Environment : Production
LOAD BALANCER
HTTP
My OpenESB application ${HTTPDefaultPort}=10100
HTTP
My OpenESB application ${HTTPDefaultPort}=10200
HTTP
My OpenESB application ${HTTPDefaultPort}=10300
Configuration 10100,10200,10300

Load balancing configuration with application variables

4 Application configuration or variable?

Next question! What is the best for your project? Is it preferable to use an application configuration or an application variable? To answer this questions one can confirm that application configuration features offer a standard and fix set of variables grouped in a configuration. Application configurations are simple to define and match a minimal requirement for portability between environments.

On the other hand, application variable features are more flexible and can be used to parametrize any element in the concrete WSDL. (ex: database schema).

It is recommended to use an application configuration for very standard configurations such as JNDI Name. For any other use cases, an application variable feature will be more efficient for matching your requirements.

5 Help and support

5.1 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

5.2 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.