

<b>Web Services Generator Tool Software Requirements Specification</b>		<b>TEM 008</b>
<b>Prepared By</b>	Nagaraja Mekala	<b>Documentation Type : SRS</b>
<b>Checked By</b>	Saurabh Bagchi/Ayushman Ghosh/Girish P	<b>Document Ref :SDLC_TEM_SRS</b>
<b>Approved By</b>	Censof	<b>Version No.: 1.0</b>
<b>Date of Release: 01/10/2018</b>		<b>No. Of Pages (Including cover page): 77</b>

< UNCONTROLLED COPY if printed (unless and otherwise specified)>



**"This document is confidential to Mindteck. The information contained herein is not to be revealed or disseminated outside the company to any other party without the prior expressed consent and written permission of Mindteck (India) Limited"**

### Revision History

<b>Change Description</b>	<b>Section Numbers Changed</b>	<b>Version No.</b>	<b>Date of Revision</b>	<b>Revised By</b>	<b>Approving Authority</b>
Baseline Draft	All	0.1	16-11-2017	Saurabh/Nagaraja	Cenof
Baseline Draft	Functional Requirements	0.2	20-11-2017	Saurabh/Nagaraja	Cenof
Baseline Draft	Functional Requirements	0.3	23-11-2017	Saurabh/Nagaraja	Cenof
Baseline Draft	Functional Requirements	0.4	28-11-2017	Saurabh/Nagaraja	Cenof
Baseline Draft	Functional Requirements	0.5	28-11-2017	Saurabh/Ayushman	Cenof
Baseline Draft	Functional Requirements	0.6	08-12-2017	Saurabh/Ayushman	Cenof
Baseline Final Version	Every Section	0.7	22-12-2017	Saurabh/Nagaraja/Ayushman	Cenof
Baseline Final Version	Sections related to Phase - II	0.8	05-01-2018	Saurabh/Nagaraja/	Cenof
Final Version 0.1	Sections related as per the feedback from Cenof	0.9	06-02-2018	Saurabh/Nagaraja/	Cenof
Final Version 0.2	Sections related to Reading the parameters from other WSDL and WADL files to generate the SOAP and REST services – User Interfaces added	0.10	09-03-2018	Saurabh	Cenof
Final Version	Sections updated: 1. Creating client to call SOAP web services 2. UI integration with back end client code for SOAP web services	0.11	11-03-2018	Saurabh	Cenof

<b>Change Description</b>	<b>Section Numbers Changed</b>	<b>Version No.</b>	<b>Date of Revision</b>	<b>Revised By</b>	<b>Approving Authority</b>
	3. Creating client to call REST web services 4. UI integration with back end client code for REST web services 5. Reading the parameters from other WSDL and WADL files to generate the SOAP and REST services				
Final Extended	Section Updated: Business Rules (Method Chaining)	0.12	14-06-2018	Saurabh Bagchi	Censof
Final Extended 2	1. Section Updated: Business Rules (Method Chaining) 2. User Management 3. Deployment Automation 4. Updated Class Diagram - Design	0.13	30-07-2018	Ashwin M and Saurabh Bagchi	Censof
Final Extended 3	User Edit Added	0.14	09-08-2018	Ashwin M and Saurabh Bagchi	Censof
Final 4 with corrections as per feedback of Censof	As per feedback of Censof	0.15	12-09-2018	Ashwin M and Saurabh Bagchi	Censof
Final 5 – Phase-I with corrections as per feedback of Censof	As per feedback of Censof	0.16	18-09-2018	Ashwin M and Saurabh Bagchi	Censof

Change Description	Section Numbers Changed	Version No.	Date of Revision	Revised By	Approving Authority
Final Phase-I with corrections as per feedback of Censof	As per feedback of Censof	1.0	01-10-2018	Ashwin M and Saurabh Bagchi	Censof

## Contents

<b>1. INTRODUCTION</b>	<b>6</b>
1.1 PURPOSE	6
1.2 SCOPE	6
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS	6
THE ABBREVIATIONS USED IN THIS DOCUMENT ARE NOTED BELOW: A) WSGD TOOL – WEB SERVICE GENERATION AND DEPLOYMENT TOOL.	
1.4 REFERENCES	6
<b>2. GENERAL DESCRIPTION</b>	<b>8</b>
2.1 OVERVIEW	8
2.2 PRODUCT PERSPECTIVE	8
2.3 PRODUCT FUNCTIONS	9
2.4 REQUIREMENTS CHANGE MANAGEMENT STRATEGY	13
2.5 BIDIRECTIONAL TRACEABILITY OF REQUIREMENTS	13
2.6 OPERATIONAL CONCEPTS AND SCENARIOS	14
2.7 ACCEPTANCE CRITERIA	14
2.8 USER CHARACTERISTICS	15
2.9 GENERAL CONSTRAINTS	18
2.10 ASSUMPTIONS AND DEPENDENCIES	18
<b>3. SPECIFIC REQUIREMENTS</b>	<b>19</b>
3.1 EXTERNAL INTERFACE REQUIREMENTS	19
3.1.1 <i>User Interfaces</i>	19
3.1.1.1 <i>Login Screen</i>	19
3.1.2 <i>Hardware Interfaces</i>	50
3.1.3 <i>Software Interfaces</i>	50
3.2 FUNCTIONAL REQUIREMENTS	50
<b>A. APPENDICES</b>	<b>72</b>
A.1 APPENDIX 1	72
A.2 APPENDIX 2	76

## 1. Introduction

### 1.1 Purpose

Purpose of this document is to capture the Software requirements of the **WebServiceGenerator and Deployment Tool** in detail. The specifications and the technical requirements mentioned in the document are derived from the technical discussions, documents and email communications exchanged between Mindteck and Censof, and the proposal submitted to Censof by Mindteck.

### 1.2 Scope

The scope of the project includes the following:

- 1) Develop a tool to generate WebService based on predefined Templates. This is for a system where many WebServices needs to be written and this tool will eliminate the tedious work of writing the WebServices, WSDL& WADL code.
- 2) The WSGD Tool shall be a wizard based tool to support Web Services, and Multiple Databases. This tool shall have the following features:
- 3) Creating a web service, Compiling the source code, Editing the properties files, Creating the war file, Downloading the war file, Deploying the war file in the configured server, Uploading the template files, Adding, editing and deleting the users.
- 4) Develop a Test Automation framework. The TestNG framework shall be used.
- 5) UI for testing the WebService by end-user/client.
- 6) Code quality analysis report shall be generated for the generated source code using SonarQube.
- 7) Generated source code shall be committed to version control SVN. UI shall be provided to Deploy the generated source code.
- 8) For CI+CD, Jenkins shall be used.

### 1.3 Definitions, Acronyms, and Abbreviations

The abbreviations used in this document are noted below:

NO.	ACRONYMS / TERMS	DEFINITIONS
1	SRS	Software Requirement Specifications
2	OSGI	Open Service Gateway Initiative
3	XML	Extensible Mark-up Language
4	TestNG	TEST Next Generation
5	WSGD	Web Service Generation and Deployment Tool
6	WSDL	Web Service Definition Language

7	WADL	Web Application Definition Language
8	SVN	SubVersion
9	REST	Representational State Transfer
10	SOAP	Simple Object Access Protocol

## 2. General Description

### 2.1 Overview

The overview of the project is to develop a tool which will generate source code, compile source code and build the OSGI bundle file for SOAP and Restful web services to deploy in the JBoss Fuse and Fabric server. The WSGD Tool shall generate WebService based on predefined Templates. The WSGD Tool shall be a wizard based tool to support Web Services and Multiple Databases. This tool shall have the following features:

Creating a web service, Compiling the source code, Editing the properties files, Creating the war file, Downloading the war file, Deploying the war file in the configured server, Uploading the template files, Adding and deleting the users. There shall be a Test Automation framework and UI for testing the WebService by end-user.

Code quality analysis report shall be generated for the generated source code using Sonar Cube.

Generated source code shall be committed to version control SVN.

For CI+CD, Jenkins shall be used.

### 2.2 Product Perspective

The tool web service generate and deployment tool shall allow the user to create web service using data source as database and data source as web service that can be deployed in JBoss fuse server.

The following are the main features that are included in service generate and deployment.

- Creating a web service.
- Compiling the source code.
- Creating the OSGI bundle file.
- Downloading the OSGI bundle file.
- Deploying the OSGI bundle file in the configured server.
- Uploading the template files.
- Adding, editing and deleting the users will be performed by Admin user. The user shall be authorized to login and use the application. User shall be authenticated before login. The general user shall only view the user details.
- Logged In Admin User shall not be allowed to edit and delete his own profile but can edit and delete the other users/admins profile.
- There shall be a Test Automation framework and UI for testing the Web Service by end-user.
- Code quality analysis report shall be generated for the generated source code using Sonar Cube.
- Jenkins configuration for build and deployment.

## 2.3 Product Functions

The main functionality of this product is to develop a tool which will generate source code, compile source code and build the OSGI bundle file for SOAP and Restful web services to deploy in the JBoss Fuse and Fabric server. The WSGD Tool shall generate WebService based on predefined Templates. The WSGD Tool shall be a wizard based tool to support Web Services and Multiple Databases.

This tool shall have the following features:

- Creating a web service, Compiling the source code, Editing the properties files, Creating the war file, Downloading the war file, Deploying the war file in the configured server, Uploading the template files, Adding and deleting the users, Adding, updating deleting the Dependency list. There shall be a Test Automation framework and UI for testing the WebService by end-user.

Along with the tool, there is also the need for:

- Developing a Test Automation framework
- Develop UI for testing the web service by end user/client
- The tool shall allow the user to specify the following
  - Name of the Webservice
  - Name of the Method
  - Input request parameters like Rows, Filters, Measure, Group by
  - Output parameters and their data types

**The Business Rule (Method Chaining) may have the following features:**

1. The WSGDT shall be able to create Web Services (REST, SOAP) where the Data Source Category may be Database (Postgres (Version PostgreSQL 10.3), MySQL(Version 5.7.21), MSSQL(Version SQL Server 2017 14.0), Oracle (Version Oracle 12 Release 2: 12.2.0.1)) or any other third-party Web Services.
2. The Business logic flow may consume the web methods of the Web Services created by Step 1 by the WSGDT, chain wise sequentially.
3. There shall be no limitation to the method chaining call.
4. The user shall be able to define the Request Parameters from the UI.
5. The very first Web Method which is part of a Web Service created by the WSGDT shall use the Request Parameters as input parameters for the Web Method.
6. From the Second Web Method (from other Web Services created by the WSGDT having either database as data source category or any other third-party web service as data source category) onwards the WSGDT may use either the Response

- parameter of the just previous Web Method and/or Request Parameters defined by the user in the Request Parameters in the as input parameters for the Web Method.
7. The Step 6 may be repeated multiple number of times with other desired Web Methods of the Web Services created by the WSGDT until the destination Web Method of the Web Service is reached by the user to achieve the desired output.
  8. The WSGD Tool shall support Mapping Request / Response parameters for web service method chaining assuming Single thread implementation, sequential call of multiple web services with Unique key dependencies only.
  9. The WSGD Tool will support Implementation of the Aggregate Results / Business Logic that is when Webservice response is a List (Example multiple values from Web service input to next Web service and so on) considering only unique key field with one to one mapping of fields.
  10. If any Web Method is returning a Data Table containing multiple Fields say F1, F2, F3...Fn, then the WSGDT shall be able to concatenate any of the combination of Fields to pass it on to the subsequent Web Method of another Web Service to perform some action over there.
  11. The WSGD Tool may be able to support Generation of Model Classes (Request and Response) Using WADL file for REST Web Services.
  12. The WSGD Tool may be able to support Generation of Model Classes (Request and Response) Using WSDL file for SOAP Web Services.
  13. The WSGD Tool shall be able to Generate Client to call the Web Service(s).
  14. The WSGD Tool shall Deploy & Configure REST Services to JBOSS Fuse.
  15. The WSGD Tool shall Deploy & Configure SOAP Services to JBOSS Fuse

To explain the scenario, the following example may be considered:

### **Business Logic Flow Example**

#### **Objective:**

Let us assume a scenario where:

- There is a database table, say "tbl\_Schools", which give the details of all Schools of a State (Province)
- There is a 3<sup>rd</sup> party Webservice which gives the list of students for a School

Also assume that we need to generate a Web Service which has the objective to Get Students of State based in StateID and Gender.

To meet the objective the tool will create a project which will have three Methods as follows:

1. **GetSchoolsForState (StateID).**

Description: This method will have Database table “tbl\_Schools” as the Data source and will return the schools of a state

2. **GetStudentsForSchool (SchoolID, Gender).**

Description: This method will have the 3<sup>rd</sup> Party WebService as the Data source and will return the Students for a school.

3. **GetStudentsForState (StateID, Gender) – This is the proposed new type of Method**

Description: This method will return the Students for State using StateID and Gender.

This method will actually call internally first call the method GetSchoolsForState (StateID)  
Then for each SchoolID it will call the method GetStudentsForSchool (SchoolID, Gender)  
Then it will collate all the output to get the desired output.

The User will specify while creating this Method the mapping of the nested Method Parameters as follows:

For first call:

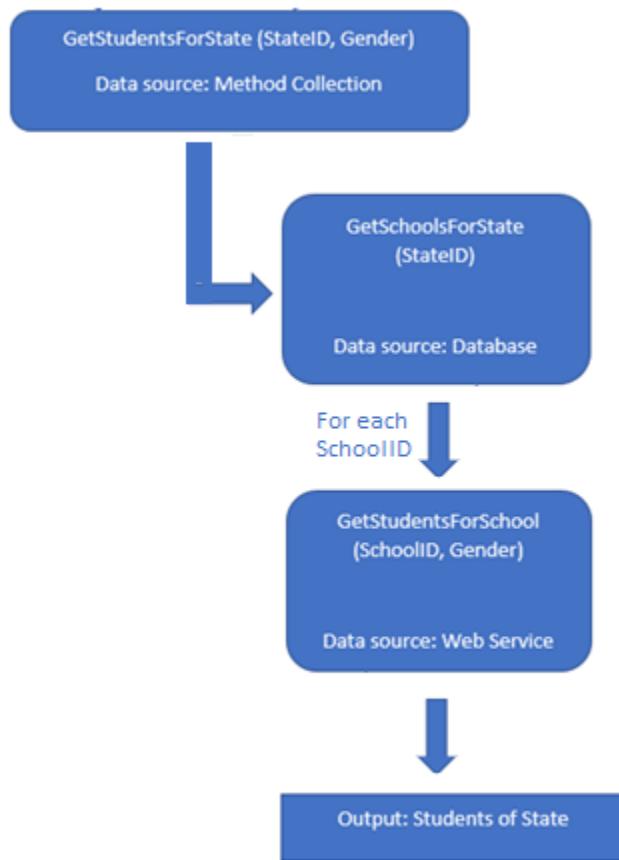
GetSchoolsForState(**StateID**) <= GetStudentsForState (**StateID**, Gender)

For second call:

GetStudentsForSchool (**SchoolID**, Gender) <= GetSchoolsForState(StateID)'s output parameter  
**SchoolID**

GetStudentsForSchool (**SchoolID**, **Gender**) <= GetStudentsForState (**StateID**, **Gender**)

### Diagrammatic Representation:



**The assumptions for the implementation of the Business Rule may be considered as follows:**

- i) Any form of cyclic and/or self-referencing of any data models or fields shall be excluded for the current implementation.
- Once all inputs are available, the tool shall be able to compile the Java code, create a deployable Webservice OSGI Bundle file and deploy the Webservice in the JBoss Fuse server.
- The destination where the Webservice needed to be created and deployed may be configurable.
- The system shall give the user an UI to enter above mentioned details and then give an option to generate and deploy the Webservices in the JBOSS server.
- Each time the Webservice is edited and regenerated, a backup copy of the earlier version may be backed up in case the user wants to go back to the immediate prior version.
- The system shall use the JBoss Fuse as the server to host the application and it may support ESB also.

- The Generator may create both SOAP and REST web services and both may support XML and JSON response format.
- For SOAP, the system shall use the CXF framework.
- For REST, the system shall use the CXF(JAX-RS) framework.
- Automation Testing Framework shall be developed using TestNG.
- Jenkins shall be used for build and deployment.
- Template shall have the header parameter information and list of validation error code and error description.
- Since this is a Government-related high-security project, adequate measures for data security shall be built in so that the data is secure. To provide the security agency\_app\_auth\_private\_key and agency\_app\_auth\_key shall be validated.
- The tool shall support multiple databases like Postgres, Oracle, MySQL and MSSql only.
- The tool shall generate the DAO classes to communicate with above mentioned database layers too and shall implement connection pooling.
- Sonar integration shall also be a part of the tool.
- Reading the parameters from other WSDL and WADL files to generate the SOAP and REST services shall also be part of the tool.
- CI/CD and Version Control. SVN shall be used for Version Controlling.
- Jenkins shall be used for CI/CD.

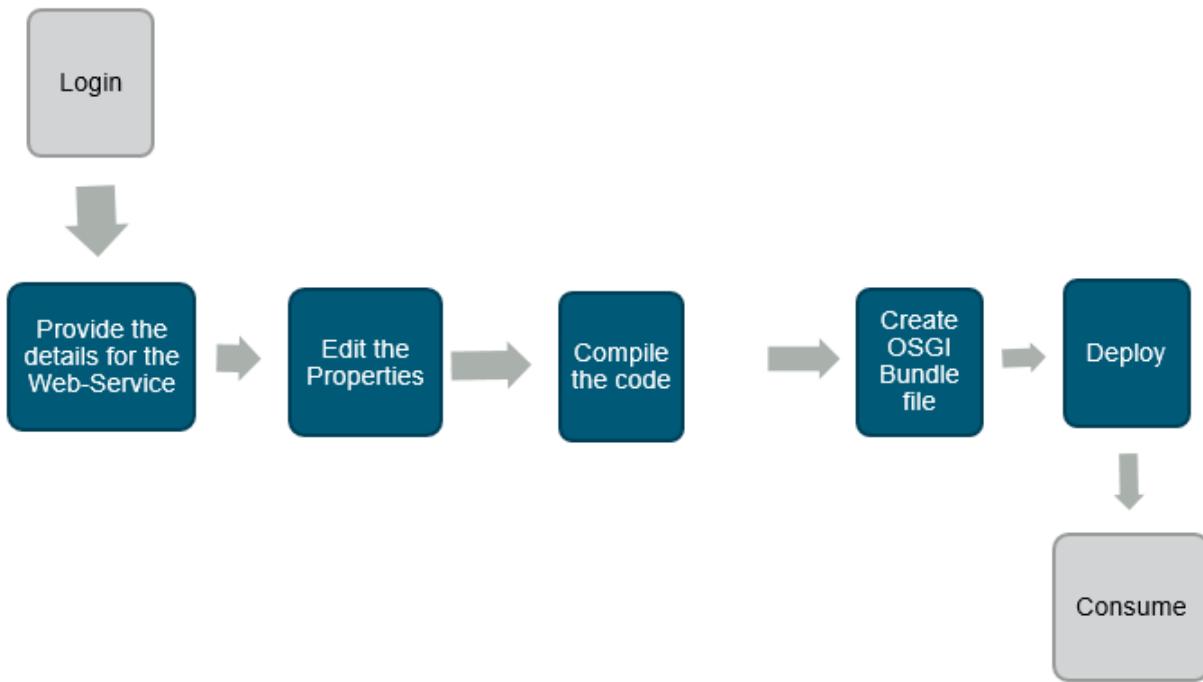
## 2.4 Requirements Change Management Strategy

Mindteck Standard CR process shall be followed. The CR process document shall be shared separately.

## 2.5 Bidirectional Traceability of Requirements

Mindteck shall prepare a RTM document for traceability requirements. Mindteck Standard RTM shall be followed. The RTM document shall be shared separately. For Traceability Template please refer to Appendix 2.

## 2.6 Operational Concepts and Scenarios



## 2.7 Acceptance Criteria

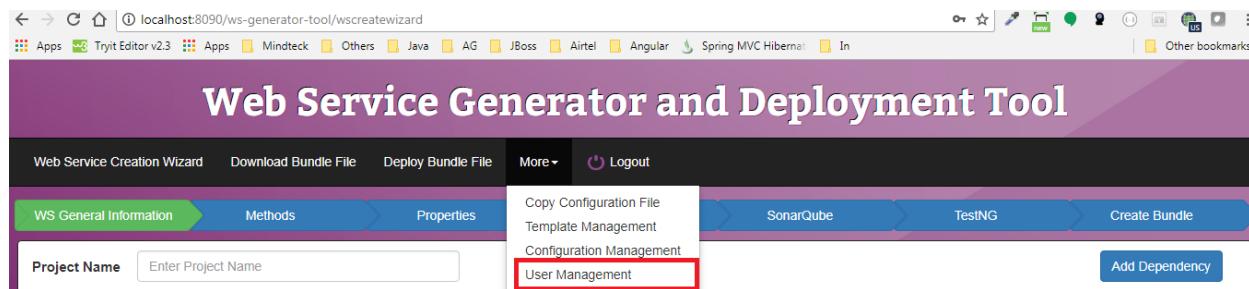
Mindteck shall prepare a system test case document for testing the Web Services Generator Tool. These test cases shall be executed for every release against the features that are developed. The results of the test cases shall be shared with Cenof for acceptance. The test strategy shall be agreed upon between Mindteck and Cenof. The Test case document will be shared separately. For Test case template please refer to Appendix 2.

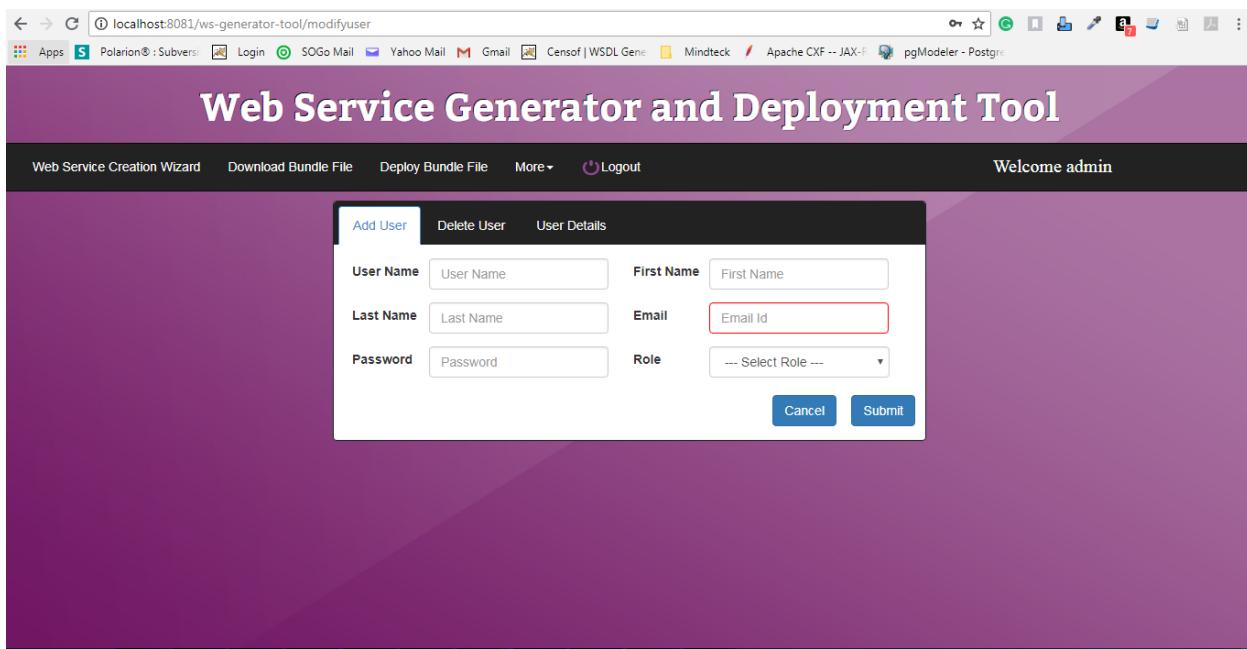
Exit Criteria	Requirement	Actual	Conformance
Test Case execution in UAT	100% UAT test cases executed.	100% Test Cases have been executed	
100% test coverage (as proven by the traceability provided in the test cases document).	100% test coverage	100% test coverage is taken care and RTM is provided in Test case Document	

Exit Criteria	Requirement	Actual	Conformance
UAT test cases pass rate.	90% UAT test cases pass rate.	100% UAT test case pass rate is there	
Impact area test cases	Impact Area test cases to be executed as part of regression testing due to fixing a defect	Impact area test cases have been traced and executed on the build	
Critical defects.	Zero (0) critical defects.	All the critical issues have been resolved	
Major defects.	Zero (0) Major defects.	All Major issues have been resolved	

## 2.8 User Characteristics

This Application shall be used by users only created from this Tool.

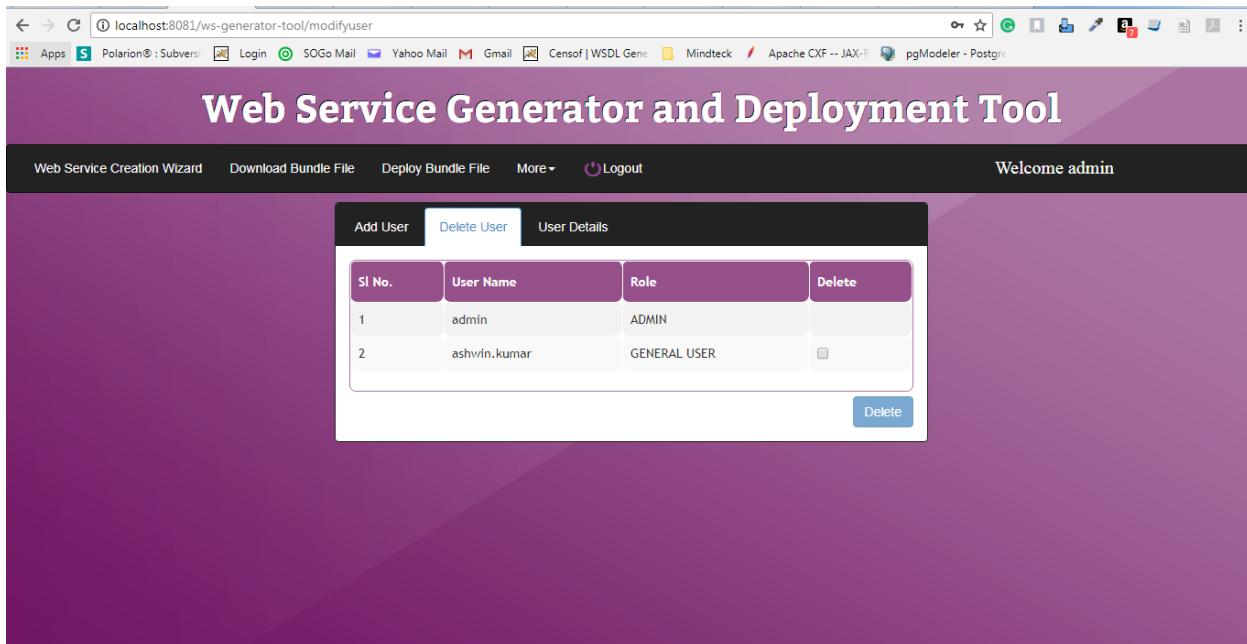


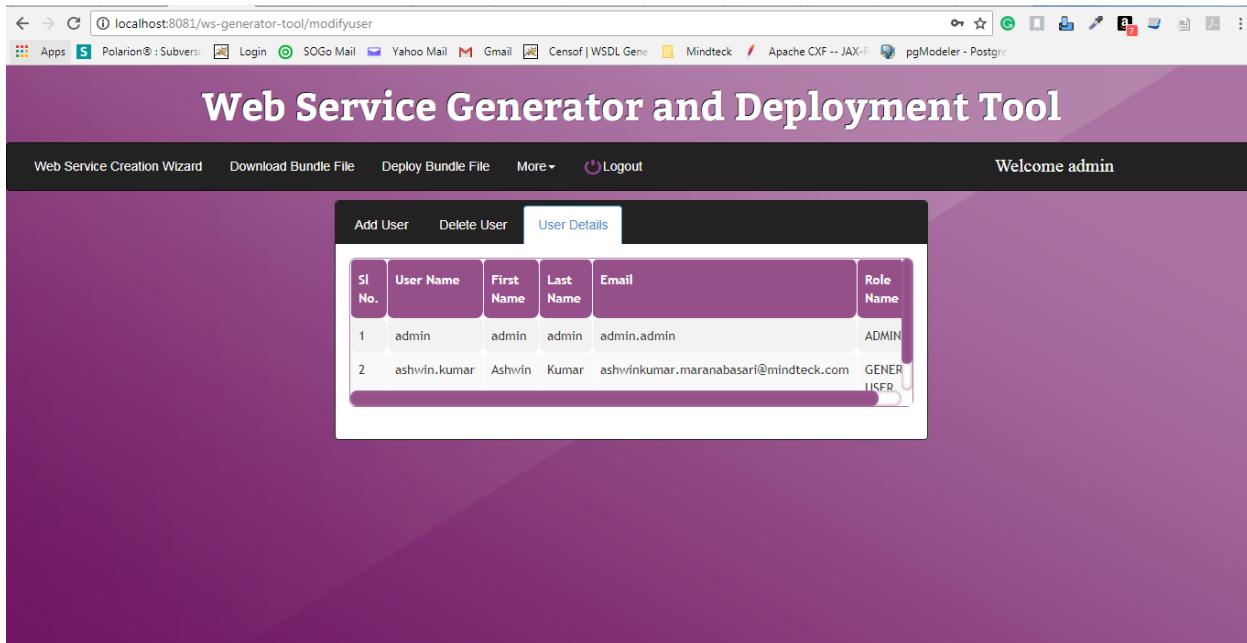


User roles and their responsibilities.

There are two roles

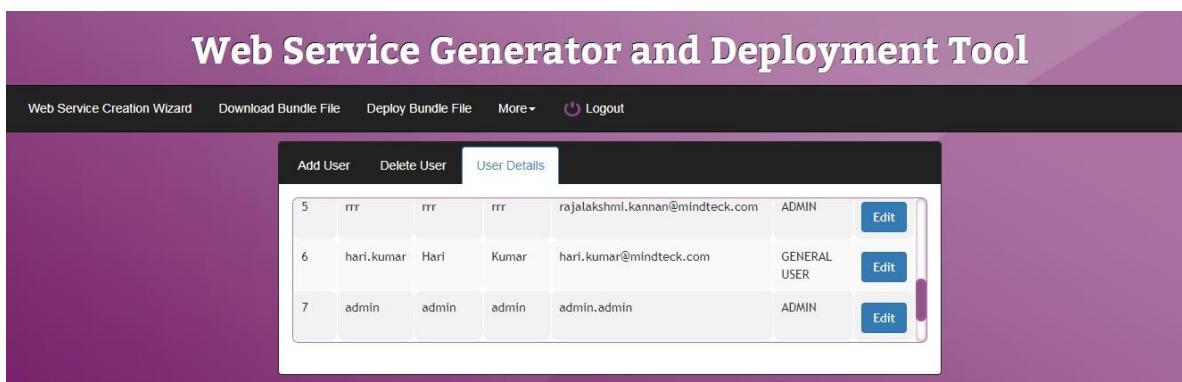
- Admin
  - Admin user can add new user, delete the existing user and can edit the existing user.
- General User
  - General User can only view the users who can use the tool.
  - General User don't have privileges to add, edit and delete the user





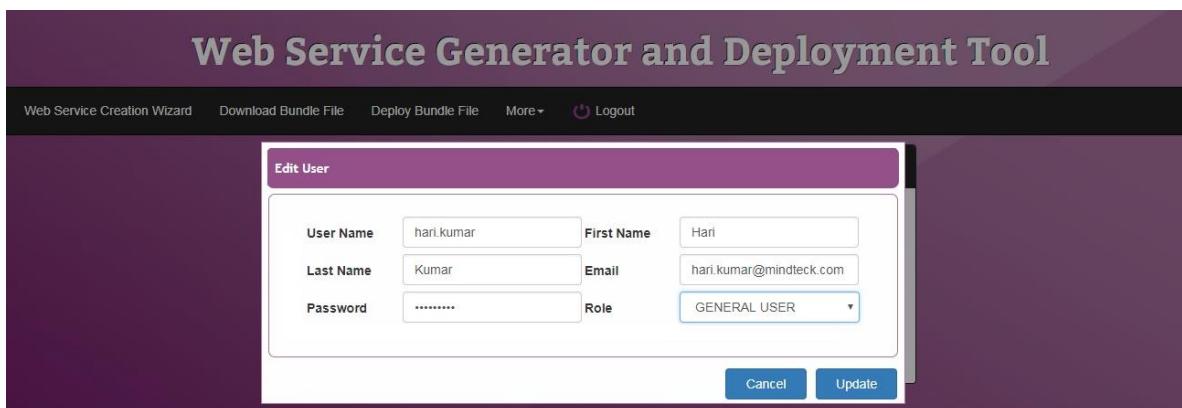
The screenshot shows the 'User Details' section of the application. It displays a table with columns: Sl No., User Name, First Name, Last Name, Email, and Role Name. There are two entries:

Sl No.	User Name	First Name	Last Name	Email	Role Name
1	admin	admin	admin	admin.admin	ADMIN
2	ashwin.kumar	Ashwin	Kumar	ashwinkumar.maranabasari@mindteck.com	GENERAL USER



The screenshot shows the 'User Details' section of the application. It displays a table with columns: Sl No., User Name, First Name, Last Name, Email, and Role Name. There are three entries:

Sl No.	User Name	First Name	Last Name	Email	Role Name
5	rrr	rrr	rrr	rajalakshmi.kannan@mindteck.com	ADMIN
6	hari.kumar	Hari	Kumar	hari.kumar@mindteck.com	GENERAL USER
7	admin	admin	admin	admin.admin	ADMIN



The screenshot shows the 'Edit User' dialog box. It contains fields for User Name (hari.kumar), First Name (Hari), Last Name (Kumar), Email (hari.kumar@mindteck.com), Password (\*\*\*\*\*), and Role (GENERAL USER). At the bottom are 'Cancel' and 'Update' buttons.

## 2.9 General Constraints

For ease of understanding of any Business Logics, it is assumed that Censof would make available all documentation, APIs or resources of the existing system to Mindteck.

The documents & business logic will include items like:

1. Sample Database design/structure document from where we can get the data.
2. Information to communicate with messaging queue like queue name, IP, port number, payload information (Parameters to send as part of the payload)
3. Discuss and finalize the structure of the template so that the template should have a logic to fetch the data and it may contain the ESB information or Database information. We need to finalize detailed information regarding template structure. e.g. how to identify template is having ESB information or database information.
4. Sample APIs from where data needs to be read.
5. Timely responses from Censof for any clarifications raised by Mindteck.

## 2.10 Assumptions and Dependencies

1. For ease of understanding of any Business Logics, it is assumed that Censof would make available all documentation, APIs or resources of the existing system to Mindteck.  
The documents will include items like:
  - a. Sample Database design/structure document from where we can get the data
  - b. ESB document for the payload information
  - c. Detailed structure for the template
  - d. Sample APIs from where data needs to be read
2. The detailed dependencies for the project which may affect project Schedule will be mutually discussed and their timeline availability agreed upon at the start of each Sprint.
3. Censof shall provide DB dump for development and may not provide any VPN access to the main servers.

### 3. Specific Requirements

The requirement of the project is to develop below features:

**Web Services Generator Tool:** This application is used to generate Restful and SOAP web services dynamically based on the inputs provided by user and based on the uploaded template. This application shall also compile the generated source code and build the OSGI bundle file. Using this tool, the user can also select the list of OSGI bundle files to deploy in the selected JBoss server.

**Web Application for testing SOAP and Restful services:** This web application is used to test the deployed Restful and SOAP web services. This web application is mainly used by the end-user.

**TestNG framework:** Develop an automation framework using TestNG to test the Restful and SOAP web service. There shall be a screen for TestNG. In this screen user shall be able to enter the test data for all the methods. TestNG framework shall generate the scripts and call the generated source code by using the entered test data. This framework shall show the results by running the dynamically generated test scripts.

#### 3.1 External Interface Requirements

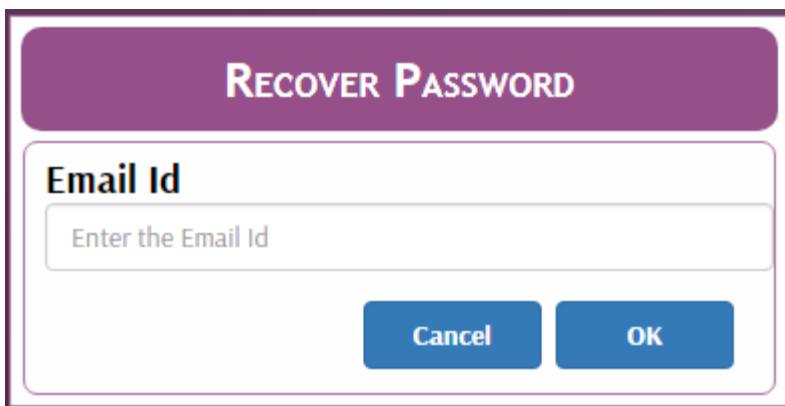
##### 3.1.1 User Interfaces

###### 3.1.1.1 Login Screen



The user shall be allowed to retry five times to login the system if wrong password is supplied.

There shall be an option for Forgot Password to recover the password. The user shall be allowed to change the password by using the link that may be sent via e-mail.



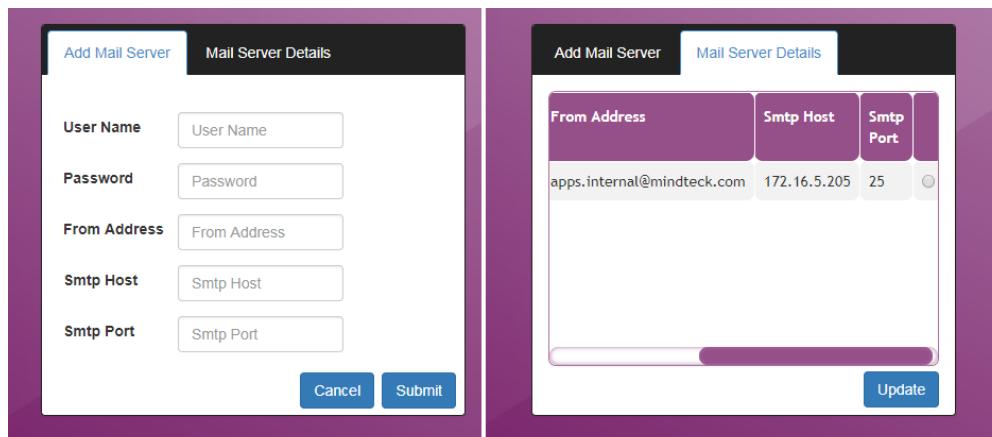
**RECOVER PASSWORD**

**Email Id**

Enter the Email Id

**Cancel** **OK**

**Mail Server Settings:** User may add multiple mail servers using the mail server feature and can activate which one is required in Mail server Details screen.

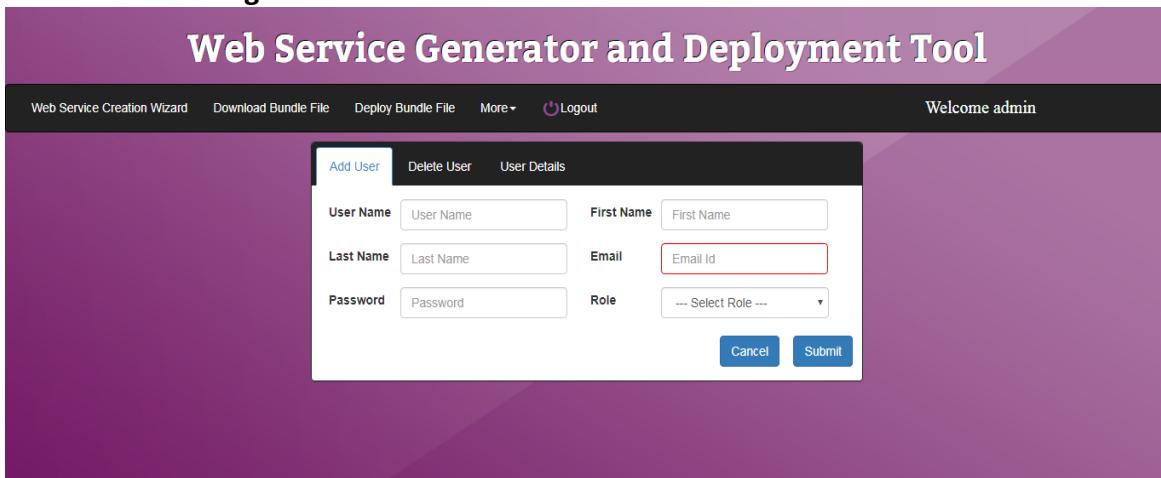


The image shows two side-by-side screens for managing mail servers.

**Left Screen (Add Mail Server):** This screen is titled "Mail Server Details". It contains fields for "User Name", "Password", "From Address", "Smtp Host", and "Smtp Port". Below these fields are "Cancel" and "Submit" buttons.

**Right Screen (Mail Server Details):** This screen also has a "Mail Server Details" title. It displays a table with three columns: "From Address" (apps.internal@mindteck.com), "Smtp Host" (172.16.5.205), and "Smtp Port" (25). There is a "Update" button at the bottom right of this screen.

### 3.1.1.2 User Management- Create User screen



This screenshot shows the "Web Service Generator and Deployment Tool" interface with a purple header bar containing the title "Web Service Generator and Deployment Tool".

The main content area has a dark purple background and features a "User Details" form. The form includes tabs for "Add User", "Delete User", and "User Details". The "User Details" tab is active, showing fields for "User Name", "First Name", "Last Name", "Email", "Password", and "Role". The "Email" field is highlighted with a red border. At the bottom of the form are "Cancel" and "Submit" buttons.

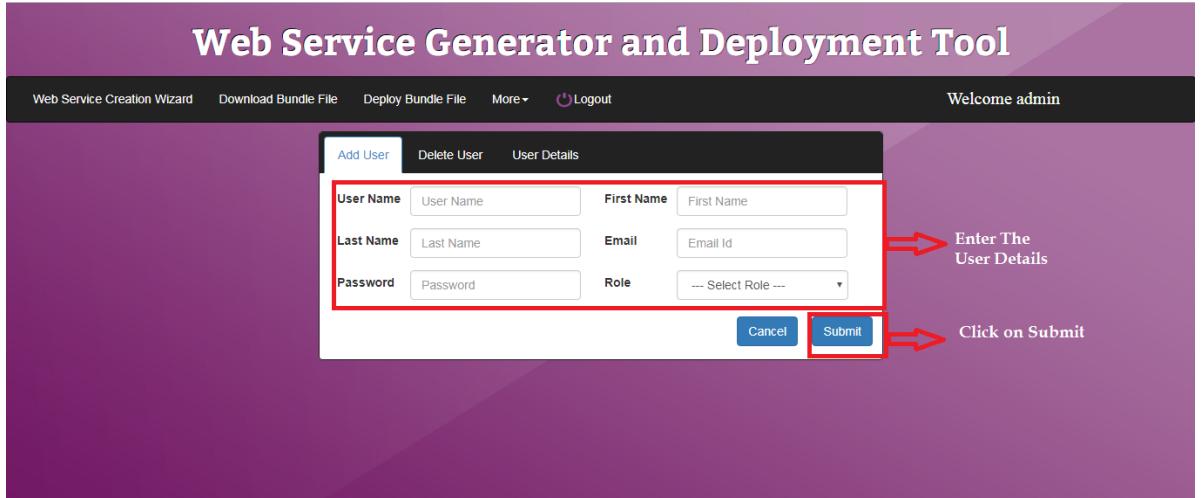
Admin User will be able add the users using the above screen with user details.

General user will be able to view only user.

The User Management screen shall have the following fields:

1. Name
2. Password – The password shall follow sha#2 algorithm. Password shall be strong and shall have the following characteristics
  - US-ASCII character encoding
  - At least eight characters
  - At least one uppercase character
  - At least one lowercase character
  - At least one number
3. Email-Id – Password shall be sent to the email id of the user when the admin user creates the user. The User shall get a mail for verification and activation. The verification and activation shall not be clicking link based. The user created shall get a default password to his/her mail id. After login the user shall get a change password screen to change the password. (*To be implemented after Phase-II completion*)

**3.1.1.3 Add User:** User can be added from the user management add user page as shown below



The screenshot shows the 'Web Service Generator and Deployment Tool' interface. At the top, there's a navigation bar with links like 'Web Service Creation Wizard', 'Download Bundle File', 'Deploy Bundle File', 'More...', and 'Logout'. On the right, it says 'Welcome admin'. Below the navigation is a sub-menu with 'Add User', 'Delete User', and 'User Details'. The main area contains a form with the following fields:
 

User Name	User Name	First Name	First Name
Last Name	Last Name	Email	Email Id
Password	Password	Role	--- Select Role ---

 The 'User Name', 'Last Name', 'Password', and 'Role' fields are highlighted with a red box. Two red arrows point to specific elements: one to the 'Submit' button with the text 'Click on Submit' and another to the 'Role' dropdown with the text 'Enter The User Details'.

**3.1.1.4 Edit User Details:** Admin User can edit the User Information

### Web Service Generator and Deployment Tool

Logout

	Add User	Delete User	User Details
5	rrr	rrr	rrr rajalakshmi.kannan@mindteck.com ADMIN Edit
6	hari.kumar	Hari	Kumar hari.kumar@mindteck.com GENERAL USER Edit
7	admin	admin	admin admin.admin ADMIN Edit

**Click On "Edit"**

### Web Service Generator and Deployment Tool

Logout

**Edit User**

User Name	hari.kumar	First Name	Hari
Last Name	Kumar	Email	hari.kumar@mindteck.com
Password	*****	Role	GENERAL USER

**Cancel** **Update** **Click ON "Update"**

**Edit the User Information**

#### 3.1.1.5 Delete User Screen

### Web Service Generator and Deployment Tool

Welcome admin

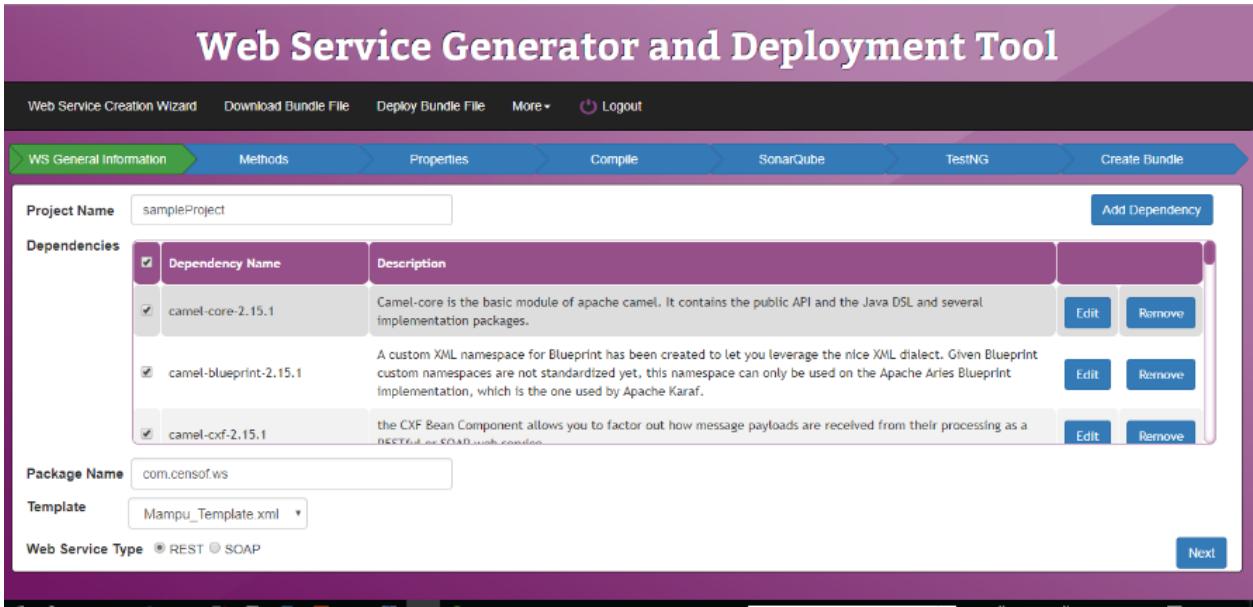
Logout

	Add User	Delete User	User Details
Sl No.	User Name	Role	Delete
1	admin	ADMIN	<input type="checkbox"/>
2	ashwin.kumar	GENERAL USER	<input checked="" type="checkbox"/>

**Delete**

The Admin shall be able to delete user using this screen. Admin may be allowed to check multiple check boxes and shall be allowed to delete the selected users.

#### 3.1.1.4 Web Service General Information Screen



The screenshot shows the 'Web Service Generator and Deployment Tool' interface. At the top, there's a navigation bar with links: 'Web Service Creation Wizard', 'Download Bundle File', 'Deploy Bundle File', 'More...', and 'Logout'. Below the navigation bar, a breadcrumb navigation shows the current path: 'WS General Information' → 'Methods' → 'Properties' → 'Compile' → 'SonarQube' → 'Testing' → 'Create Bundle'. The main content area is titled 'WS General Information'. It has fields for 'Project Name' (set to 'sampleProject'), 'Dependencies' (listing 'camel-core-2.15.1', 'camel-blueprint-2.15.1', and 'camel-cxf-2.15.1' with descriptions and edit/remove buttons), 'Package Name' (set to 'com.censof.ws'), 'Template' (set to 'Mampu\_Template.xml'), and 'Web Service Type' (set to REST). A 'Next' button is visible at the bottom right.

1. The Webservice General Information screen shall contain the Project Name (input box) and Dependencies.
2. The user may be able to add dependencies.
3. The user shall be able to edit particular dependencies by selecting that dependency by checking the check box one at a time.
4. The user shall be able to delete multiple dependencies by selecting multiple dependencies from the list.

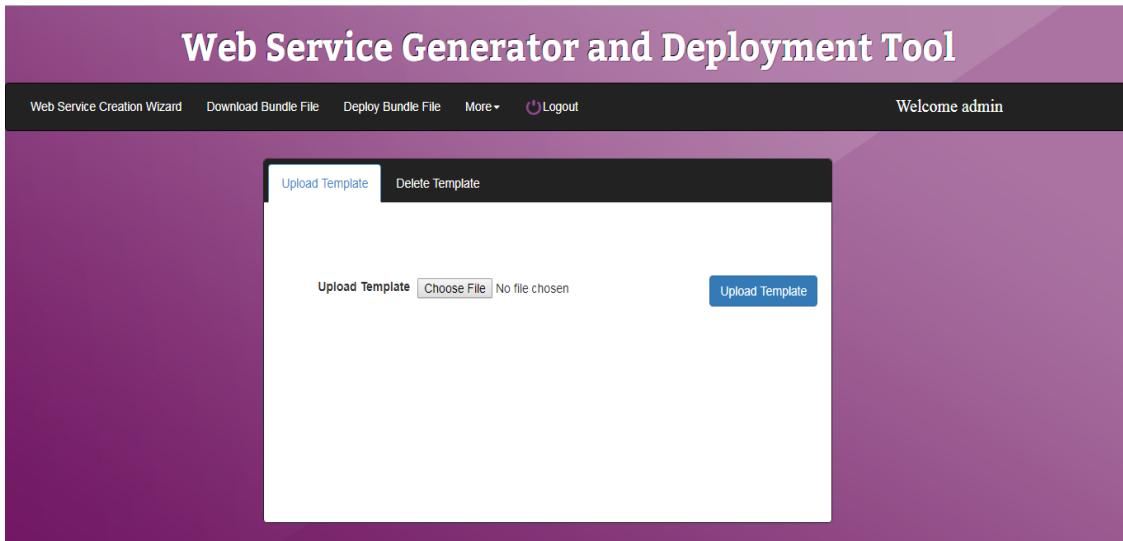
### **3.1.1.5 Template Management**

User may be able to click on Template Management under More menu to see the template management screen.

In this screen there shall be two tabs with name Upload Template and Delete Template.

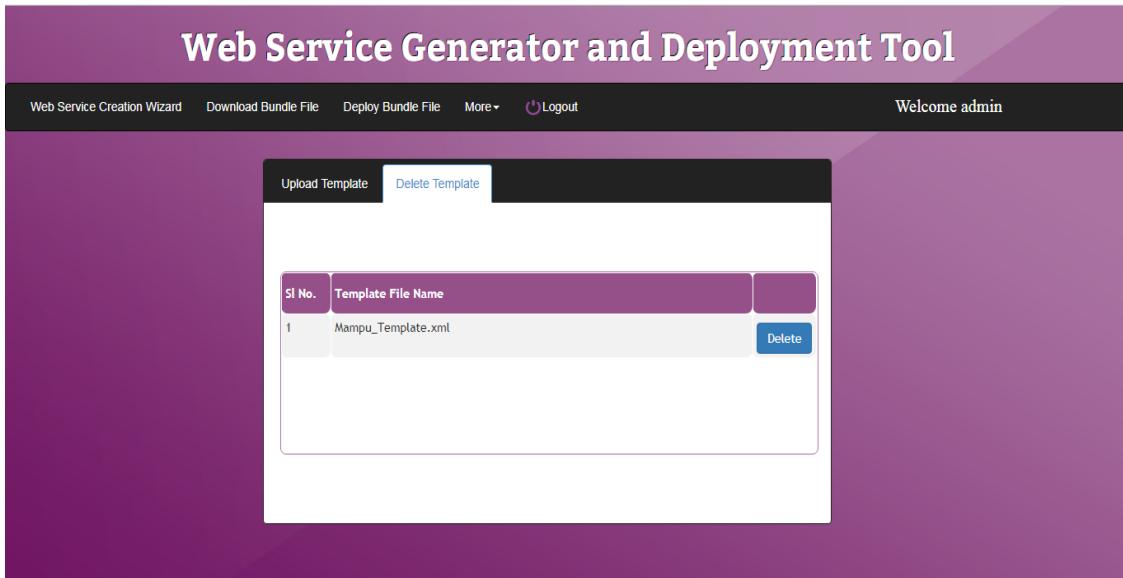
#### **3.1.1.5.1 Upload Template:**

To upload the template user shall click on Choose File to select the template to upload. After that the user shall click on Upload Template button. Uploaded template shall be saved in the configured location. If the file already exists, the incremental sequence number shall be maintained as version number and same shall be appended to the file name.



### **3.1.1.5.2 Delete Template:**

To delete the available uploaded template user may be able to click on Delete Template Tab. User can delete any number of uploaded templates by checking the check boxes and by clicking on Delete Template button.



### **3.1.1.8 Data Validation:**

Service name and service version shall be validated with available data in the database table with name service\_details.

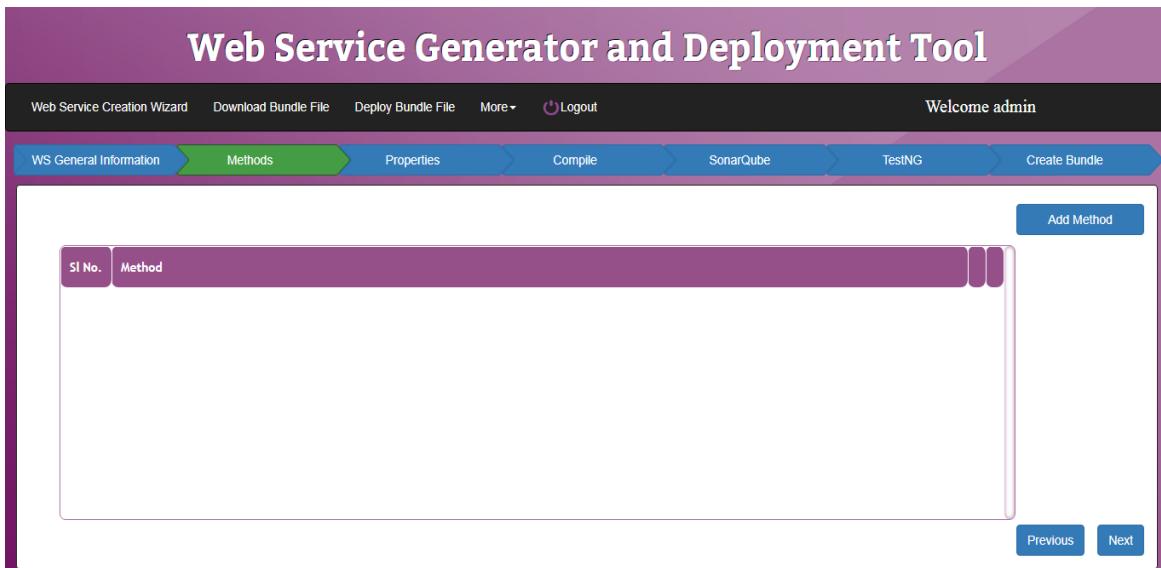
Also, the agencyCode, agencyAppAuthKey and agencyAppAuthCode shall be validated based on the data available in the data base.

### 3.1.1.9 Validation Error Response Messages:

The following section represents the general error codes that shall be used in the response messages.

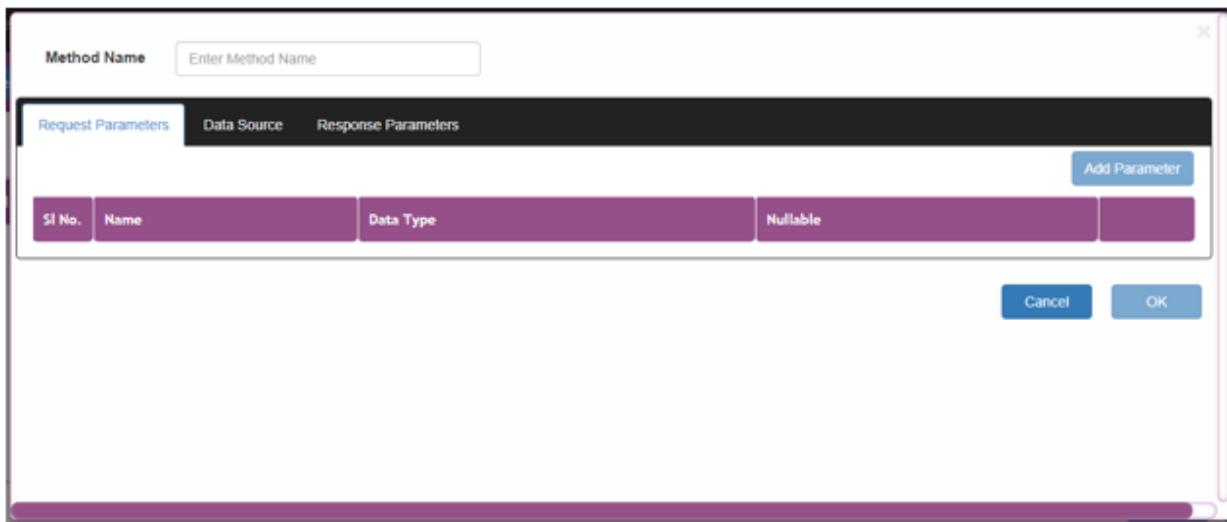
```
No_Error = 0;
User_Not_Found = 1;
Invalid_UserUID = 2;
Required_Parameter_Is_Missing = 3;
Required_Parameter_Contains_Data_Format_Error = 4;
Required_Parameter_Contains_No_Data = 5;
Internal_Software_Exception = 6;
Database_Exception = 7;
Invalid_Message_Format = 8;
Unsupported_Command = 9;
Optimistic_Locking_Error = 10;
Too_Many_Payloads = 11;
Timestamp_Malformed = 12;
NotEnough_Header_Rows = 13;
WS_Timeout = 14;
WS_Cannot_Connect = 15;
All_Targets_Unavailable = 16;
WS_Route_Not_Found = 17;
WS_Exceeds_Connection_Limit = 18;
WS_Rejected_Due_To_Filter_Rules = 19;
Duplicate_Record = 20;
Field_Too_Long_To_Store_In_Database = 21;
WS_Down_For_Maintenance = 22;
Warnings_Present_Without_Error = 1000;
Multiple_Verification_Errors = 1001;
Decompression_Error = 1111;
```

**Screen 1.**

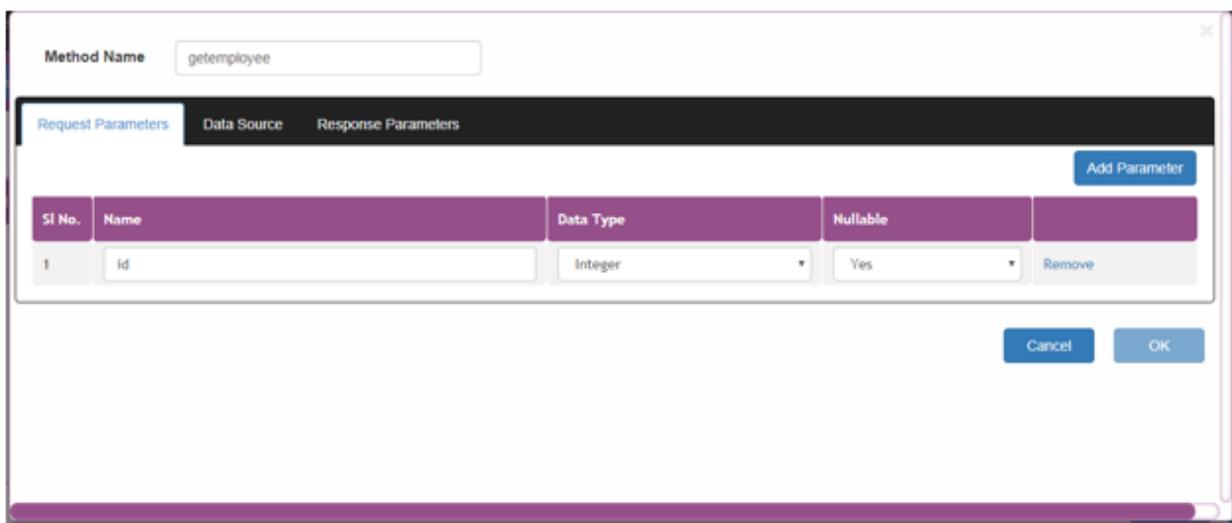


- The Above screen describes adding methods.

## Screen 2.



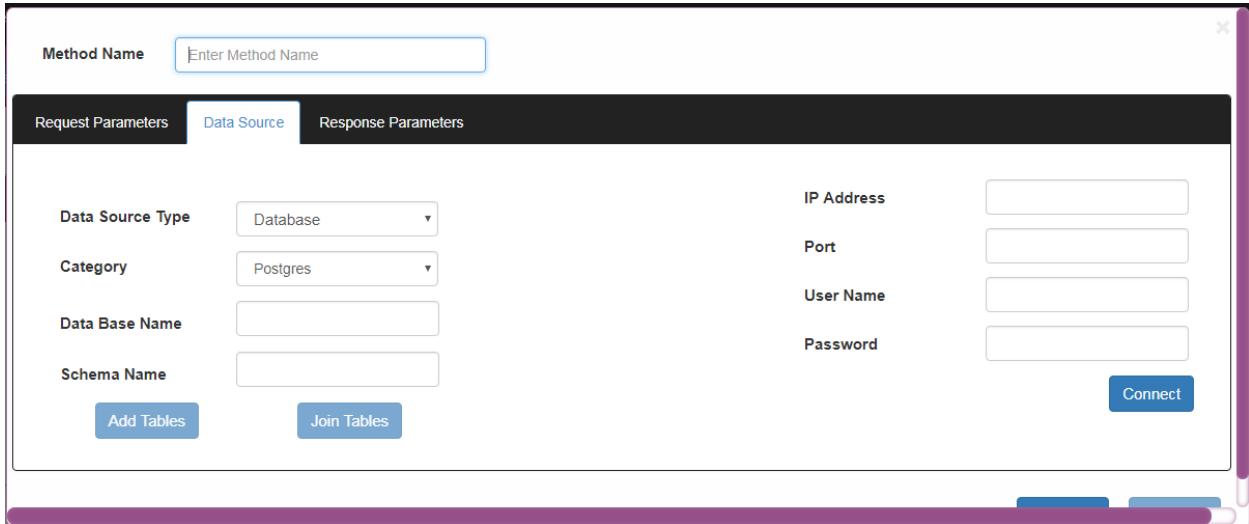
- The above screen is to provide request parameter details for methods.

**Screen 3.**


The screenshot shows a dialog box for defining request parameters. The method name is set to "getemployee". The "Request Parameters" tab is selected, showing one parameter: "Id" (Data Type: Integer, Nullable: Yes). Buttons for "Cancel" and "OK" are visible at the bottom right.

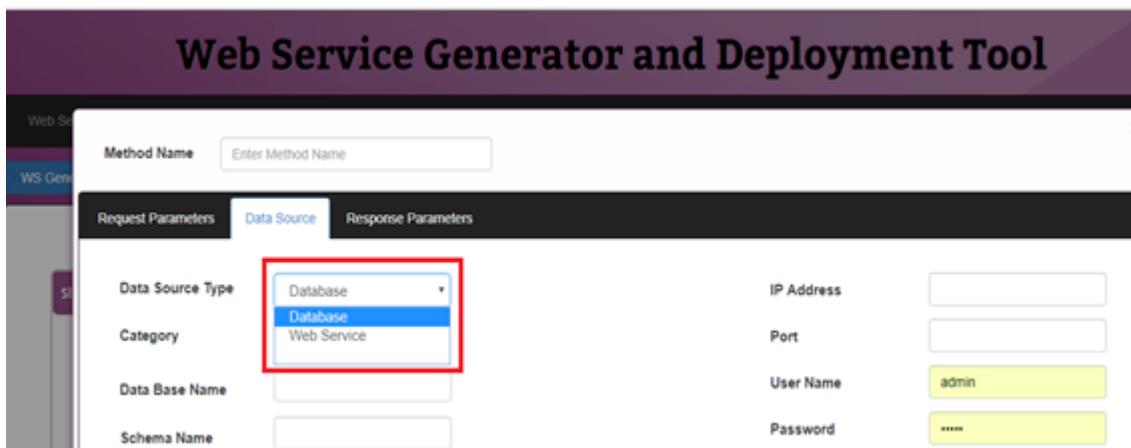
Sl No.	Name	Data Type	Nullable
1	Id	Integer	Yes

- The above screen is to provide request parameter details for methods.

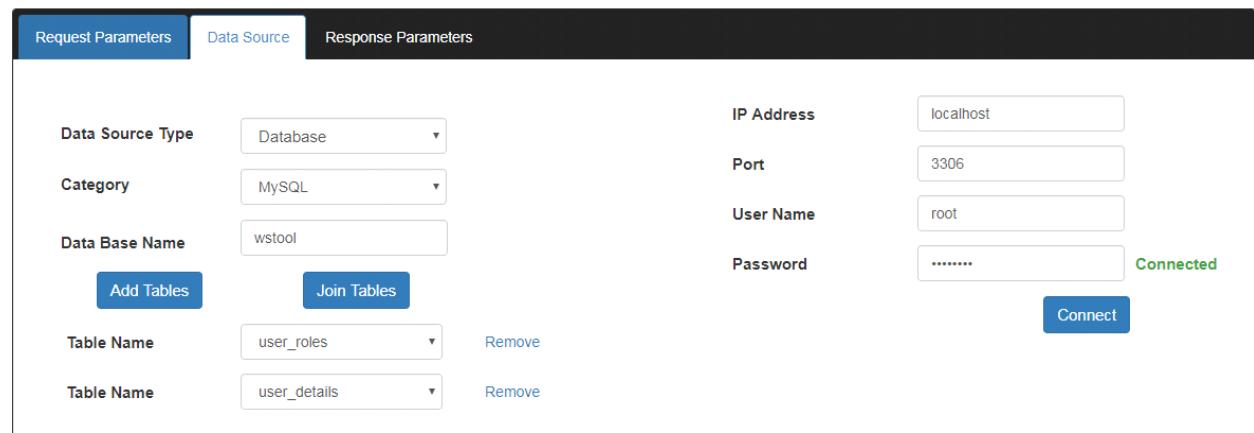
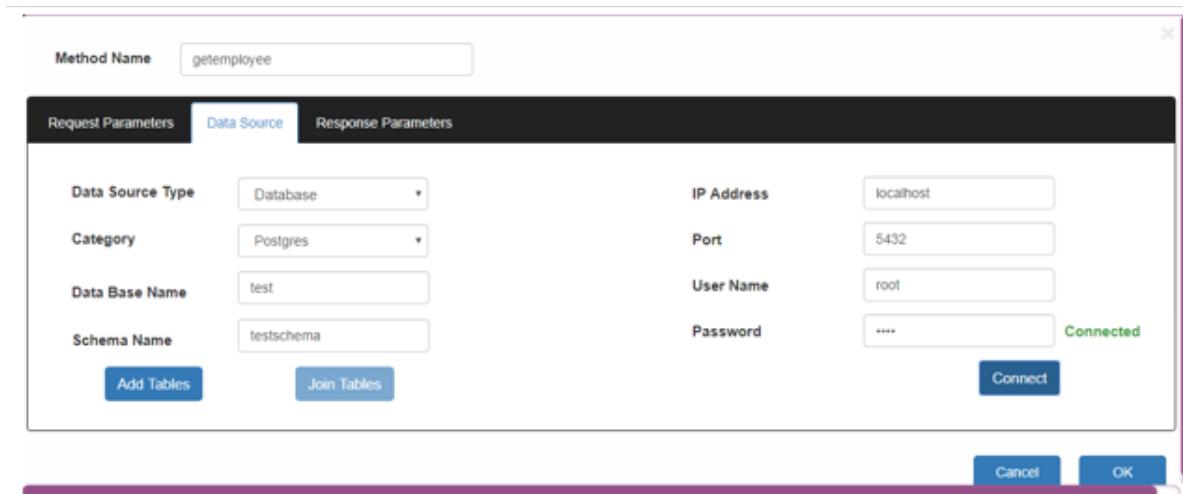
**Screen 4**


The screenshot shows a dialog box for defining data source details. It includes fields for Data Source Type (Database), Category (Postgres), Data Base Name, Schema Name, IP Address, Port, User Name, Password, and a "Connect" button. Buttons for "Add Tables" and "Join Tables" are also present.

- The above screen is to provide Data source details for methods.
- Data Source type are database with connection details.

**Screen 5.**


- Data Source type are database with connection details.



Request Parameters	Data Source	Response Parameters
Data Source Type: Database Category: MSSQL Data Base Name: TestDB <a href="#">Add Tables</a> <a href="#">Join Tables</a> Table Name: Persons <a href="#">Remove</a> Table Name: Inventory <a href="#">Remove</a>	IP Address: localhost User Name: sa Password: ..... <span style="color: green;">Connected</span> <a href="#">Connect</a>	
Data Source Type: Database Category: Postgres Data Base Name: test Schema Name: testschema <a href="#">Add Tables</a> <a href="#">Join Tables</a> Table Name: employee <a href="#">Remove</a> Table Name: department <a href="#">Remove</a>	IP Address: localhost Port: 5432 User Name: root Password: .... <span style="color: green;">Connected</span> <a href="#">Connect</a>	
Data Source Type: Database Category: Oracle Service Name: TestDB <a href="#">Add Tables</a> <a href="#">Join Tables</a>	IP Address: localhost Port: 1630 User Name: SA Password: ..... <span style="color: green;">Connected</span> <a href="#">Connect</a>	

- The above screen is to provide Data source details for methods.
- Data Source type are database with connection details.

## Screen 6.

### Join Tables

Sl No	Join Type	Left Table Name	Left Column Name	Right Table Name	Right Column Name	Remove
						<button style="border: none; background-color: inherit;">Remove</button>

Add Joins
Cancel OK

### Join Tables

Sl No	Join Type	Left Table Name	Left Column Name	Right Table Name	Right Column Name	Remove
1	LEFT JOIN	department	dept_id	employee	department_id	<button style="border: none; background-color: inherit;">Remove</button>

Cancel OK

- The above screen is to join two or more tables.

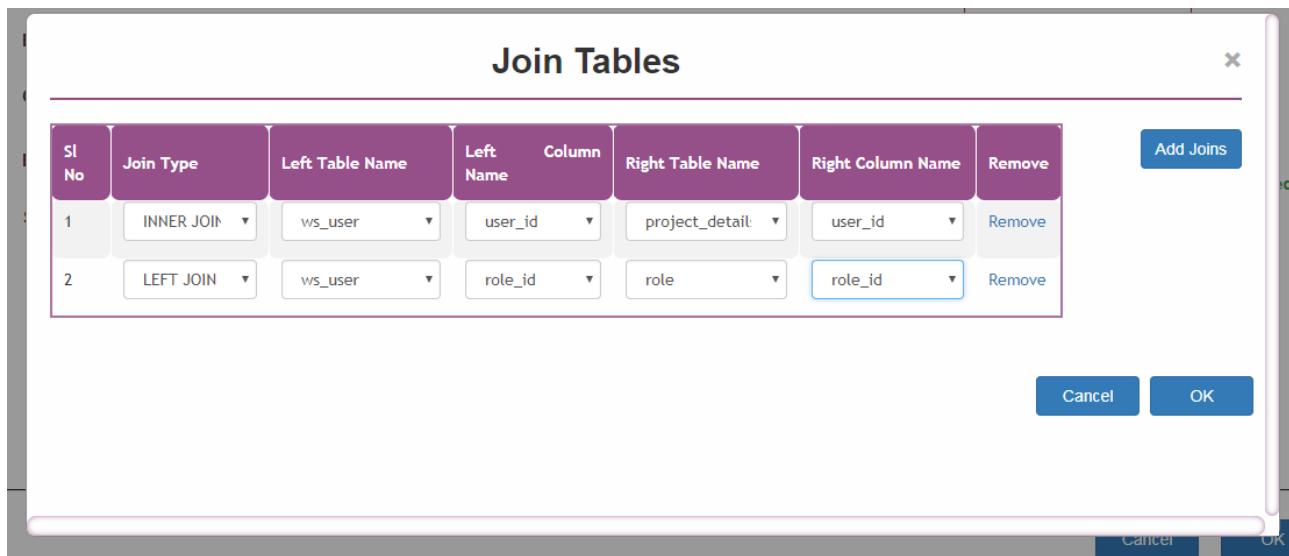
#### Example of Joins

Tables With Common Column :- ws\_user, role, project\_details

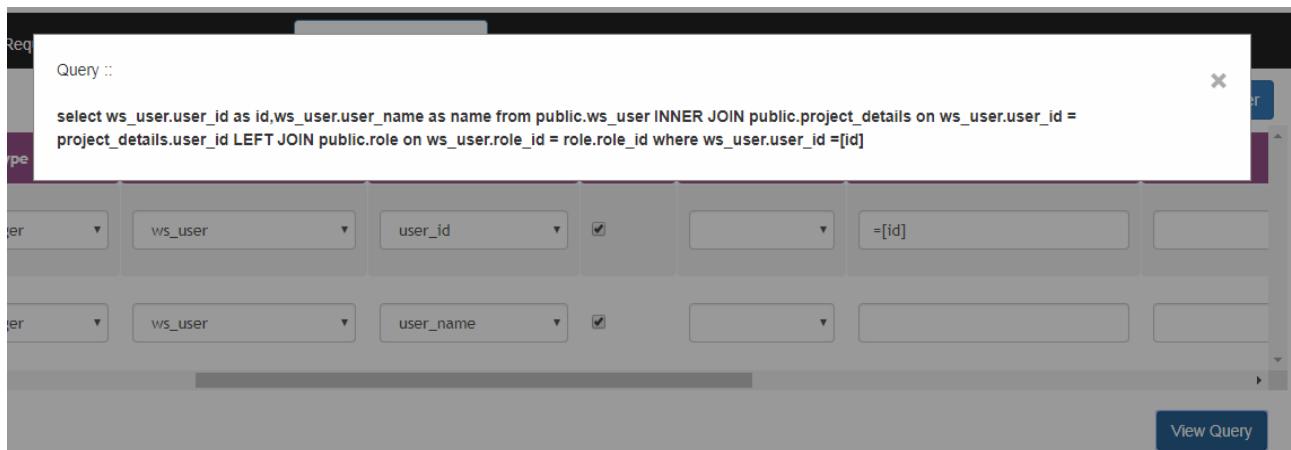
Step1 – Joining the tables using common column/field.

1<sup>st</sup> JOIN -Inner join between the tables ws\_user and project\_details which have the common column user\_id.

2<sup>nd</sup> JOIN – Left join between the tables ws\_user and role which have the common column role\_id.



Step 2- Getting the select and from clause from the response parameter table and displaying the query



Step3 – Executing query in postgres.

The screenshot shows a software interface for generating Web Services. At the top, there's a toolbar with various icons for file operations like save, open, and search, along with a dropdown menu set to "No limit". Below the toolbar is a blue header bar with the text "wstool on postgres@PostgreSQL 9.6". The main area contains a SQL query:

```
1 select ws_user.user_id as id,ws_user.user_name as name from public.ws_user
2 INNER JOIN public.project_details on ws_user.user_id = project_details.user_id
3 LEFT JOIN public.role on ws_user.role_id = role.role_id where ws_user.user_id =2
```

Below the query, there are tabs for "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, showing a table with two columns: "id" and "name". The data row is:

	<b>id</b>	<b>name</b>
1	2	admin

Example2 :-

1<sup>st</sup> Join – Right join between the tables project\_details and ws\_user

2<sup>nd</sup> Join – Full join between the tables ws\_user and role.

## Join Tables

Sl No	Join Type	Left Table Name	Left Column Name	Right Table Name	Right Name	Column	Remove
1	RIGHT JOIN	project_detail	user_id	ws_user	user_id		<a href="#">Remove</a>
2	FULL JOIN	ws_user	role_id	role	role_id		<a href="#">Remove</a>

[Add Joins](#)

[Cancel](#)
[OK](#)

Query for example 2.

Query ::

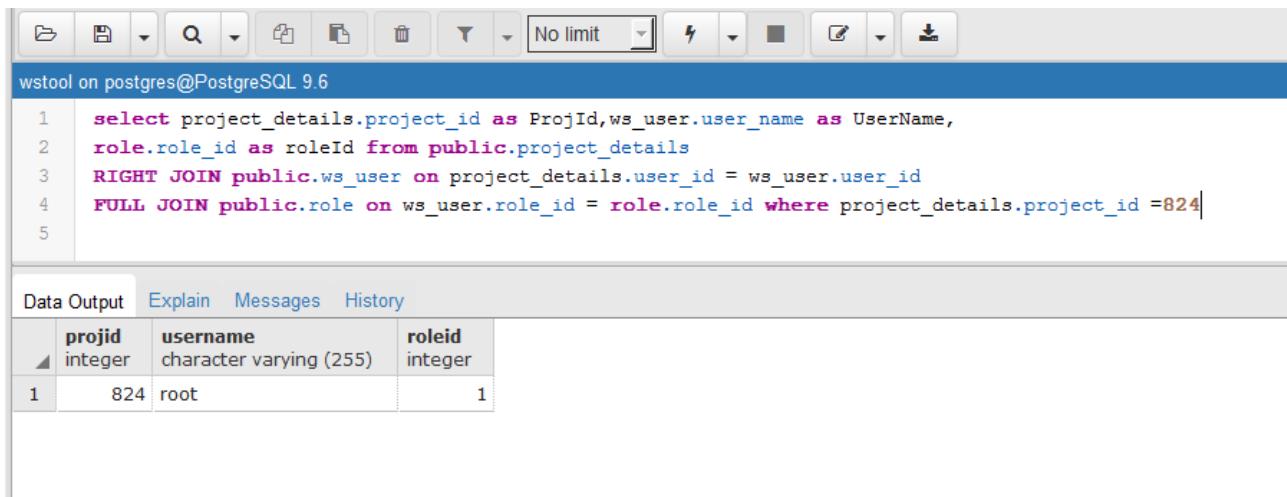
```
select project_details.project_id as ProjId,ws_user.user_name as UserName,role.role_id as roleId from public.project_details RIGHT JOIN public.ws_user
on project_details.user_id = ws_user.user_id FULL JOIN public.role on ws_user.role_id = role.role_id where project_details.project_id =[id]
```

Integer

role\_id

[ ]
[View Query](#)

Result for the above query.



wstool on postgres@PostgreSQL 9.6

```

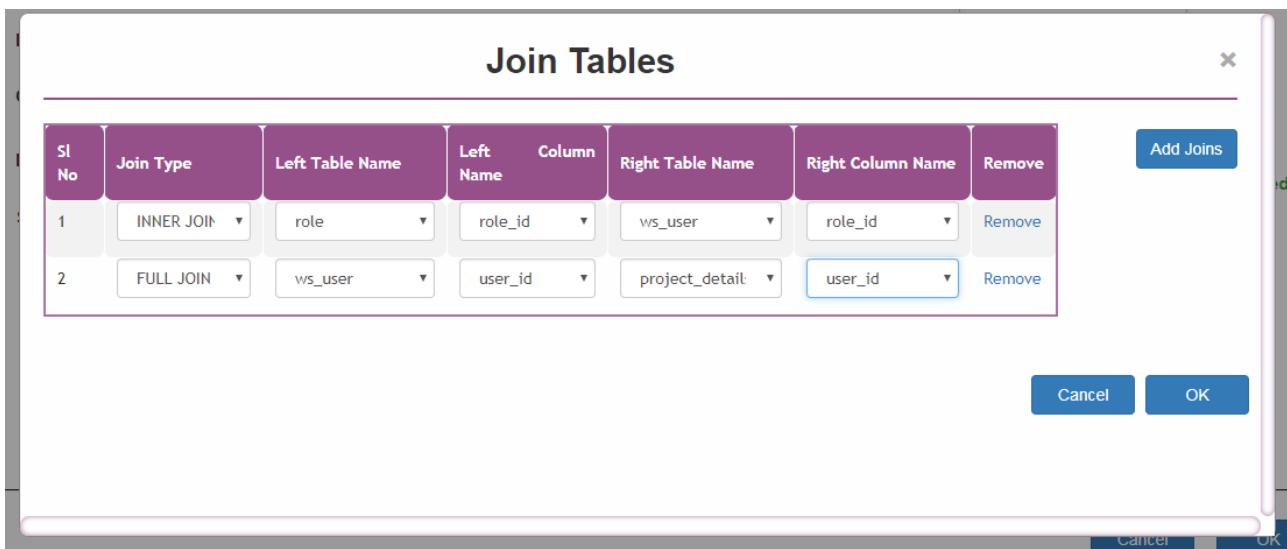
1 select project_details.project_id as ProjId,ws_user.user_name as UserName,
2 role.role_id as roleId from public.project_details
3 RIGHT JOIN public.ws_user on project_details.user_id = ws_user.user_id
4 FULL JOIN public.role on ws_user.role_id = role.role_id where project_details.project_id =824|
5

```

Data Output			Explain	Messages	History
projid	username	roleid			
integer	character varying (255)	integer			
1	824	root			1

Example 3.

1<sup>st</sup> Join – Inner Join between the tables role and ws\_user.  
 2<sup>nd</sup> Join- Full Join between the tables ws\_user and project\_details.



Query For Example 3.

1 Query ::

```
2 select role.role_id as RoleId,ws_user.first_name as UserFirstName,project_details.project_name as ProjName from public.role INNER JOIN public.ws_user
on role.role_id = ws_user.role_id FULL JOIN public.project_details on ws_user.user_id = project_details.user_id where role.role_id =[id]
```

3 ProjName String project\_details project\_name  View Query

Result for the above query.

wstool on postgres@PostgreSQL 9.6

```
1 select role.role_id as RoleId,ws_user.first_name as UserFirstName,project_details.project_name as ProjName from public.role
2 INNER JOIN public.ws_user on role.role_id = ws_user.role_id
3 FULL JOIN public.project_details on ws_user.user_id = project_details.user_id where role.role_id =1
```

Data Output Explain Messages History

	roleid integer	userfirstname character varying (255)	projname character varying (255)
1	1	root	test
2	1	admin	test1

Example 4.

1<sup>st</sup> Join – Left join between the tables project\_details and ws\_user.

2<sup>nd</sup> Join- Right join between the tables role and ws\_user.

**Join Tables**

Sl No	Join Type	Left Table Name	Left Column Name	Right Table Name	Right Name	Column	Remove
1	LEFT JOIN	project_detail	user_id	ws_user	user_id		Remove
2	RIGHT JOIN	ws_user	role_id	role	role_id		Remove

Add Joins Cancel OK

Query for Example 4.

Distinct Results  Add Response Parameter

Query ::

```
select project_details.project_id as ProjId,ws_user.is_active as UserActive,role.role_name as RoleName from public.project_details LEFT JOIN public.ws_user on project_details.user_id = ws_user.user_id RIGHT JOIN public.role on ws_user.role_id = role.role_id where project_details.project_id =[id]
```

X

Result for the above query.

B D U Q F L C T N No limit ▼ L ▼ D ▼ A ▼

wstool on postgres@PostgreSQL 9.6

```
1 select project_details.project_id as ProjId,ws_user.is_active as UserActive,role.role_name as RoleName from public.project_details
2 LEFT JOIN public.ws_user on project_details.user_id = ws_user.user_id
3 RIGHT JOIN public.role on ws_user.role_id = role.role_id where project_details.project_id =824
```

▼

Data Output Explain Messages History

	projid	useractive	rolename
	integer	boolean	character varying (255)
1	824	true	ADMIN

### Screen 7

Method Name: getemployee

Request Parameters   Data Source   Response Parameters

Sl No.	Name	Data Type	Table Name	Mapped Data Field	Visible	Aggregate	Where
1	name	String	employee	first_name	<input checked="" type="checkbox"/>		
2	deptname	String	department	department_name	<input checked="" type="checkbox"/>		

Add Response Parameter   View Query

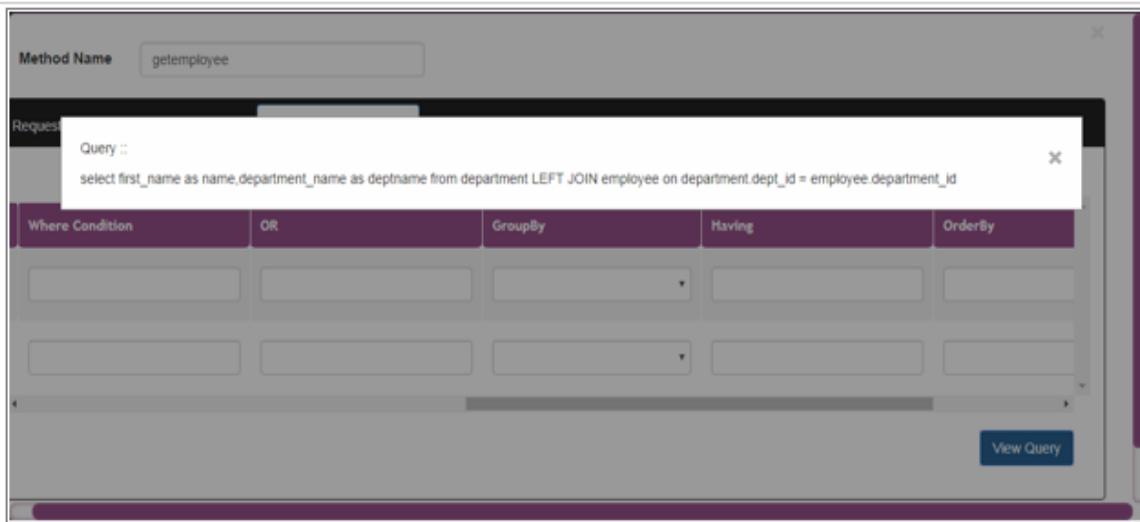
Method Name: getemployee

Request Parameters   Data Source   Response Parameters

Where Condition	OR	GroupBy	Having	OrderBy

Add Response Parameter   View Query

- The above screen is to provide response parameter details for methods.

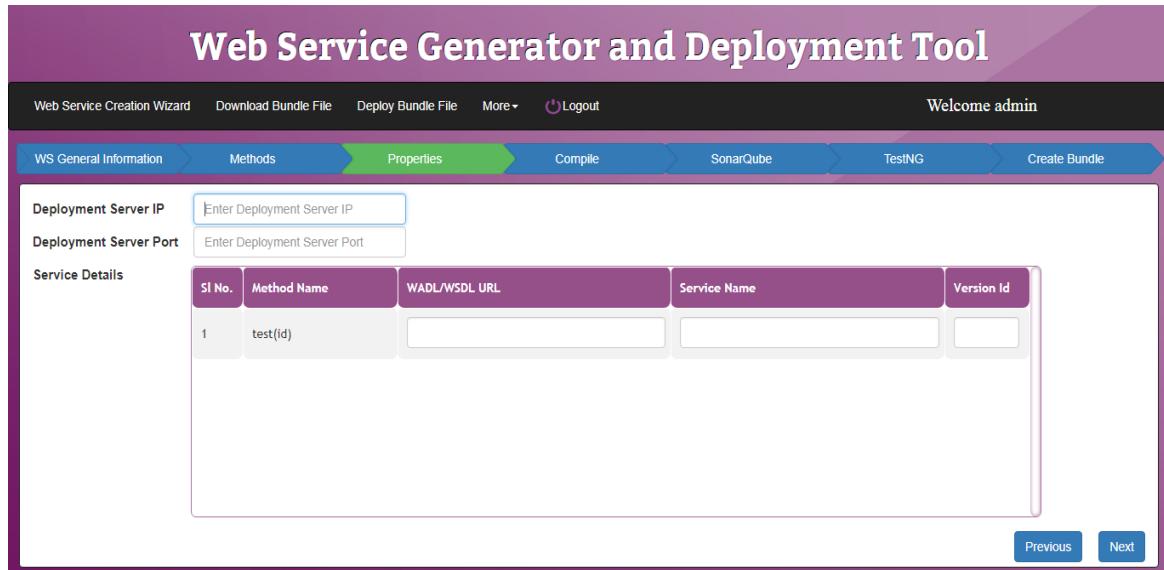
**Screen 8.**


The screenshot shows a modal window titled "Method Name" with the value "getemployee". Below it is a "Request" section with a "Query" input field containing the SQL code:

```
select first_name as name,department_name as deptname from department LEFT JOIN employee on department.dept_id = employee.department_id
```

Below the query are five filter fields: "Where Condition", "OR", "GroupBy", "Having", and "OrderBy". A "View Query" button is located at the bottom right of the modal.

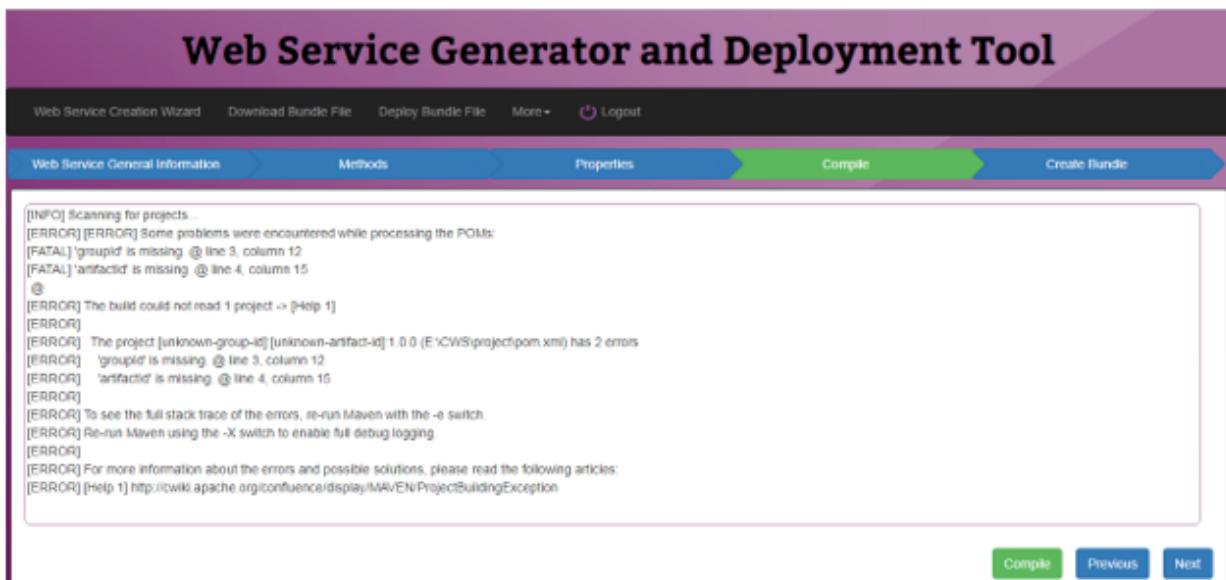
- The above screen displays the query.

**Screen 9**


SI No.	Method Name	WADL/WSDL URL	Service Name	Version Id
1	test(id)			

- The above Screens is to provide the property details.

### Screen 10

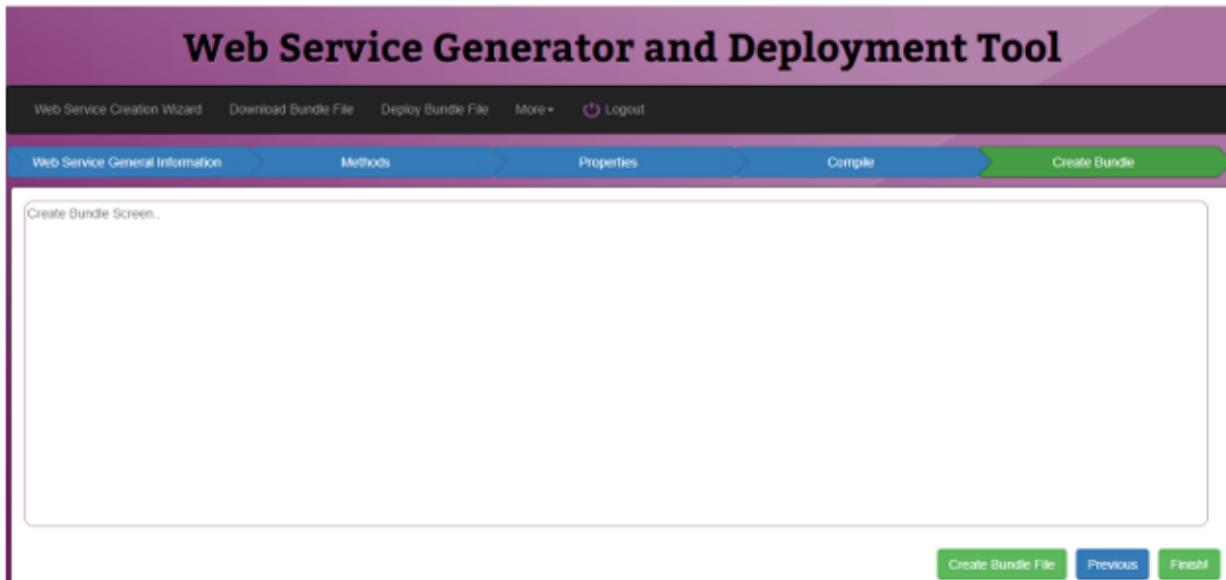


The screenshot shows the 'Compile' step of the tool. The 'Properties' tab is selected. A large text area contains the following Maven build errors:

```
[INFO] [scanning for projects...]
[ERROR] [ERROR] Some problems were encountered while processing the POMs:
[FATAL] 'groupId' is missing. @ line 3, column 12
[FATAL] 'artifactId' is missing. @ line 4, column 15
 @@
[ERROR] The build could not read 1 project -> [Help 1]
[ERROR]
[ERROR]   The project [unknown-group-id] [unknown-artifact-id] 1.0.0 (E:\CWS\project\pom.xml) has 2 errors
[ERROR]
[ERROR]   'groupId' is missing. @ line 3, column 12
[ERROR]   'artifactId' is missing. @ line 4, column 15
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/ProjectBuildingException
```

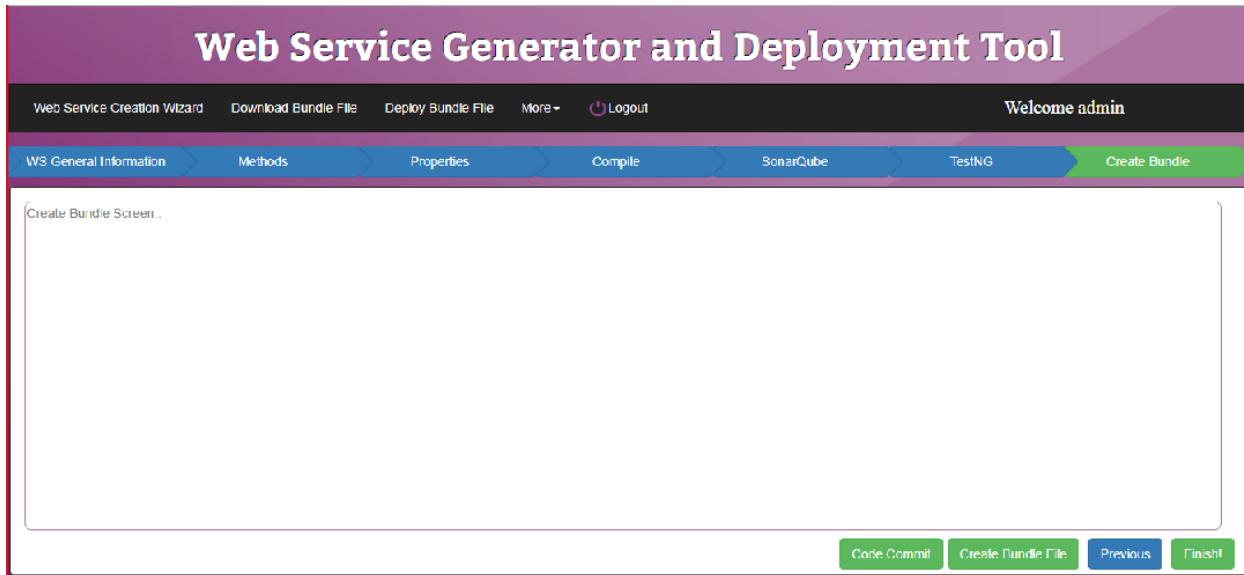
- The above screen is provided to compile the generated source code.

### Screen 11

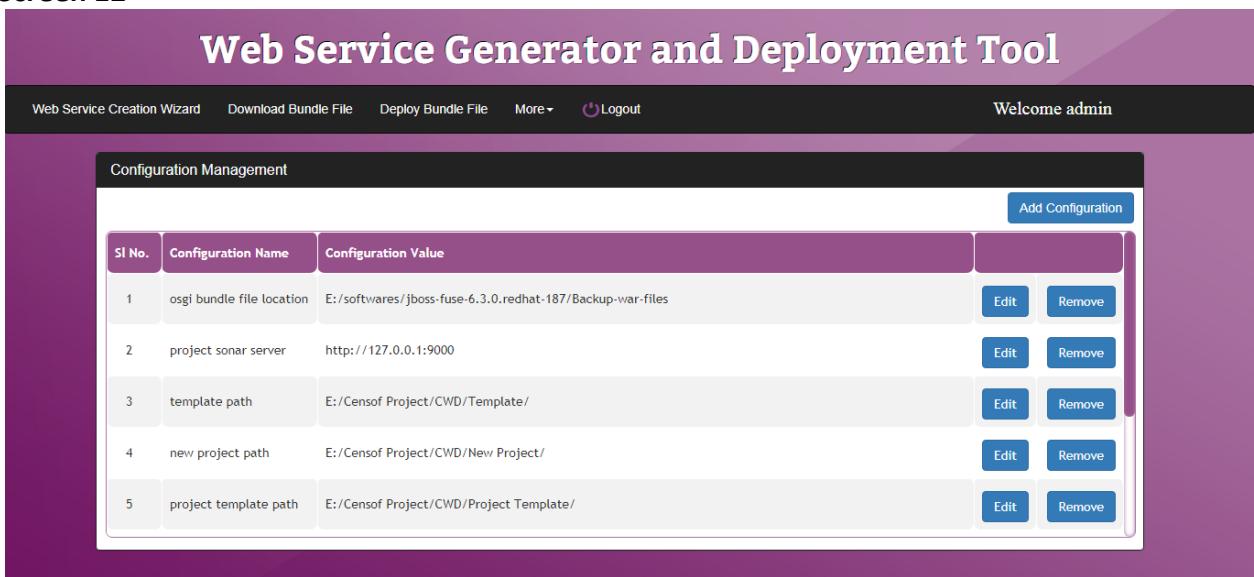


The screenshot shows the 'Create Bundle' step of the tool. The 'Create Bundle' tab is selected. A large text area displays the message: "Create Bundle Screen."

- The above screen is provided to create the bundle file for generated source code.



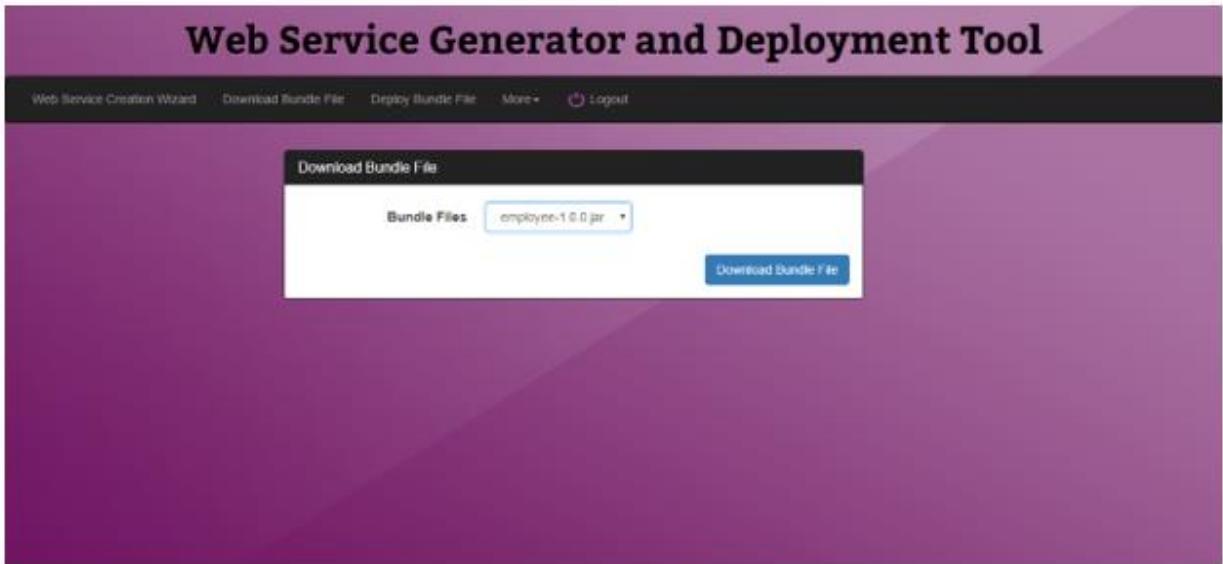
**Screen 12**



The screenshot shows the 'Configuration Management' screen. At the top, there is a navigation bar with links: 'Web Service Creation Wizard', 'Download Bundle File', 'Deploy Bundle File', 'More', and 'Logout'. To the right of the navigation bar, it says 'Welcome admin'. Below the navigation bar is a table titled 'Configuration Management'. The table has columns: 'SI No.', 'Configuration Name', and 'Configuration Value'. There are five rows of data:

SI No.	Configuration Name	Configuration Value	Edit	Remove
1	osgi bundle file location	E:/softwares/jboss-fuse-6.3.0.redhat-187/Backup-war-files	Edit	Remove
2	project sonar server	http://127.0.0.1:9000	Edit	Remove
3	template path	E:/Censof Project/CWD/Template/	Edit	Remove
4	new project path	E:/Censof Project/CWD/New Project/	Edit	Remove
5	project template path	E:/Censof Project/CWD/Project Template/	Edit	Remove

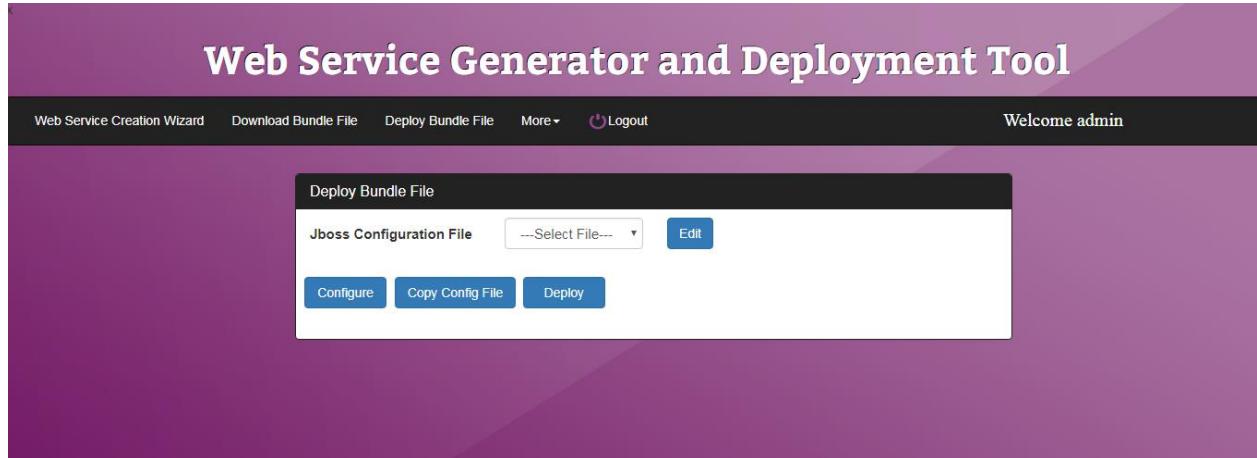
- The above screen is configuration management where user can add delete and edit the configuration.

**Screen 13**

- The Above screen is provided to download the bundle file.

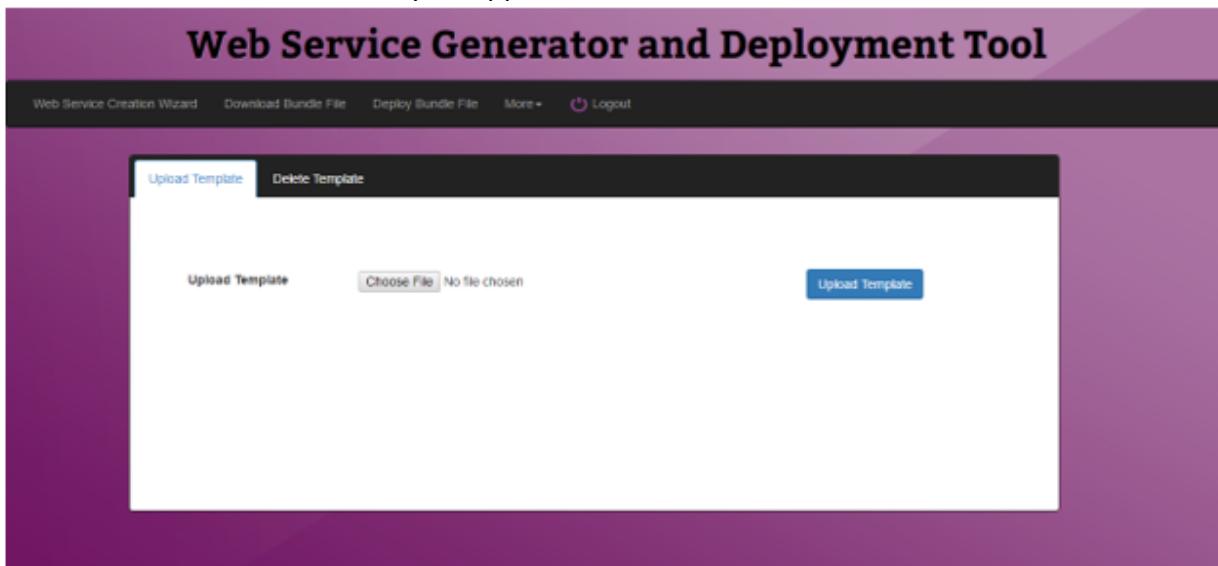
Automatic Deployment screen:

- User shall be allowed to enter the JBoss fuse commands in configuration file by selecting JBoss configuration file from dropdown.
- After saving the configuration the User shall click on the “Configure” button.
- After Jboss server configuration is successful the User shall copy the configuration file generated by tool to JBoss fuse instance.
- After Step (3) the user shall Click the “Deploy” button.
- User shall get the URL which will be displayed on screen to access the list of services deployed in JBoss server.



Template Management Screens:

If the template file already exists, the incremental sequence number shall be maintained as version number and same may be appended to the file name

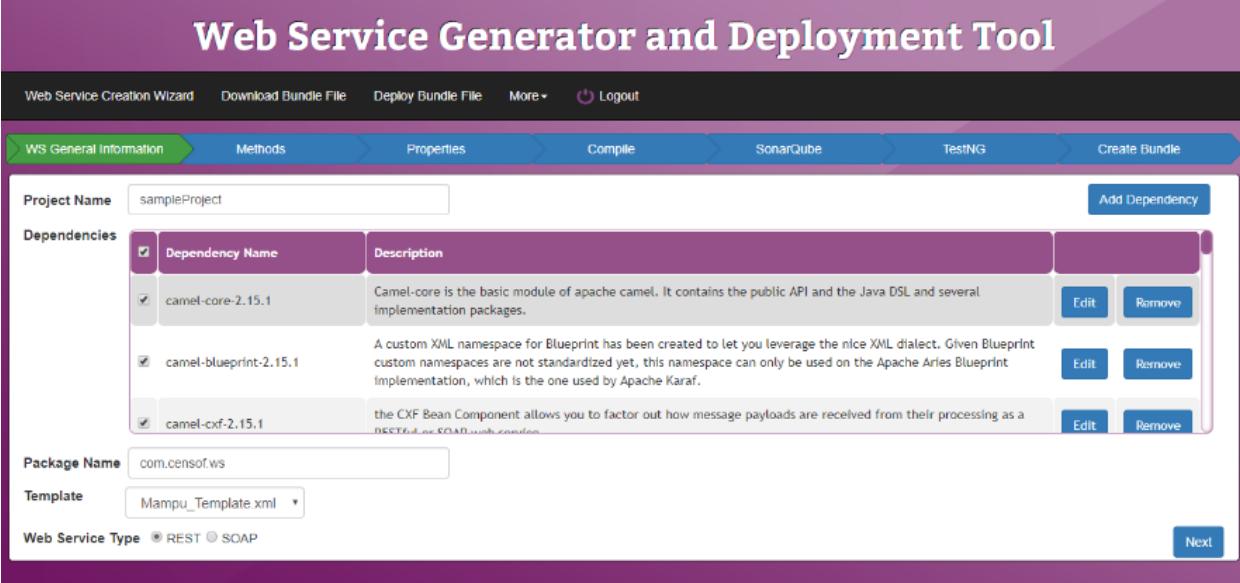


The screenshot shows the interface after a template has been uploaded. The top navigation bar remains the same. Below it, a table displays the uploaded template. The table has columns for 'Sl No.', 'Template File Name', and 'Delete'. There is one entry: '1' in the Sl No. column, 'Mampu\_Template.xml' in the Template File Name column, and a delete icon in the Delete column. At the bottom right of the table area is a blue 'Delete Template' button.

Sl No.	Template File Name	Delete
1	Mampu_Template.xml	

**Screens for the implementation of Reading the parameters from other WSDL and WADL files to generate the SOAP and REST services: (Refer Section 3.8 for details)**

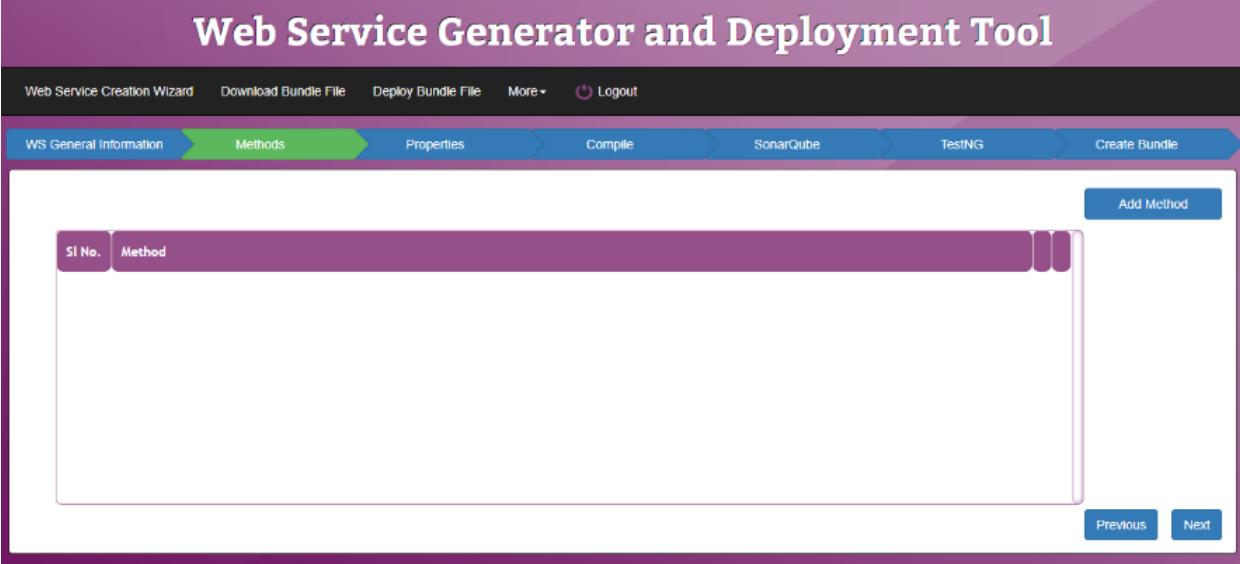
**Screen-1**



The screenshot shows the 'WS General Information' step of the Web Service Creation Wizard. The 'Project Name' is set to 'sampleProject'. Under 'Dependencies', three items are listed: 'camel-core-2.15.1', 'camel-blueprint-2.15.1', and 'camel-cxf-2.15.1'. Each dependency has a description and 'Edit' and 'Remove' buttons. The 'Package Name' is 'com.censof.ws', 'Template' is 'Mampu\_Template.xml', and 'Web Service Type' is selected as 'REST'. A 'Next' button is visible at the bottom right.

Screen 1 is the initial screen where the user shall be allowed to enter Project Name and Dependencies, Package name, Select Template and mention the Web Service Type.

**Screen-2**

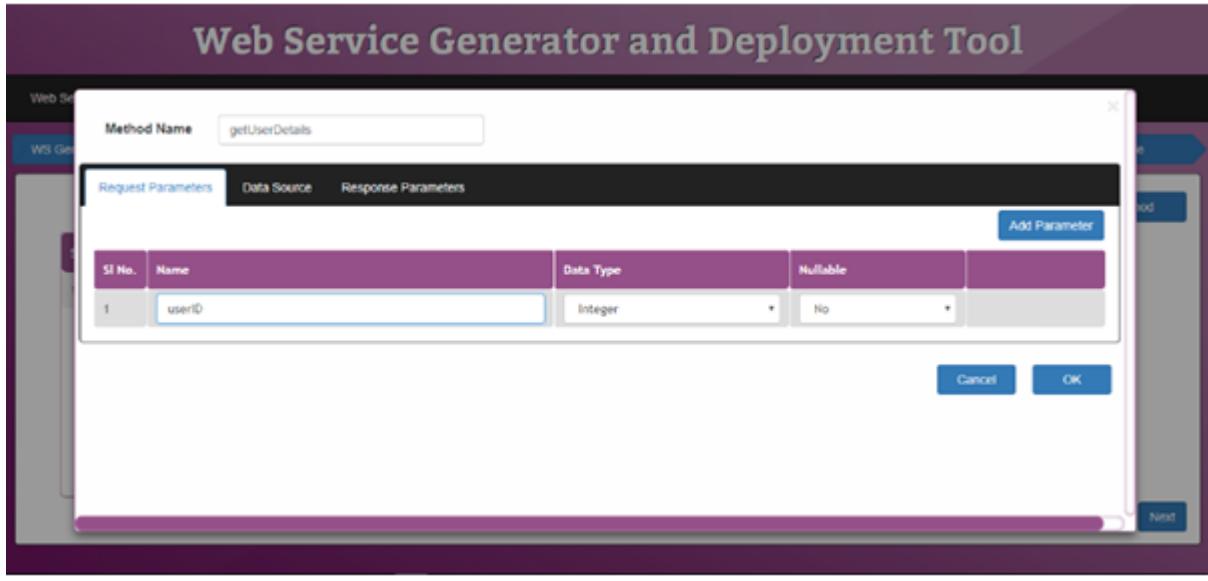


The screenshot shows the 'Methods' step of the Web Service Creation Wizard. A table header with 'Sl No.' and 'Method' columns is shown, but the table body is empty. A 'Add Method' button is at the top right of the table area. Navigation buttons 'Previous' and 'Next' are at the bottom right.

Screen-2 shall allow the user to add methods.

### Screen-3

**Web Service Generator and Deployment Tool**



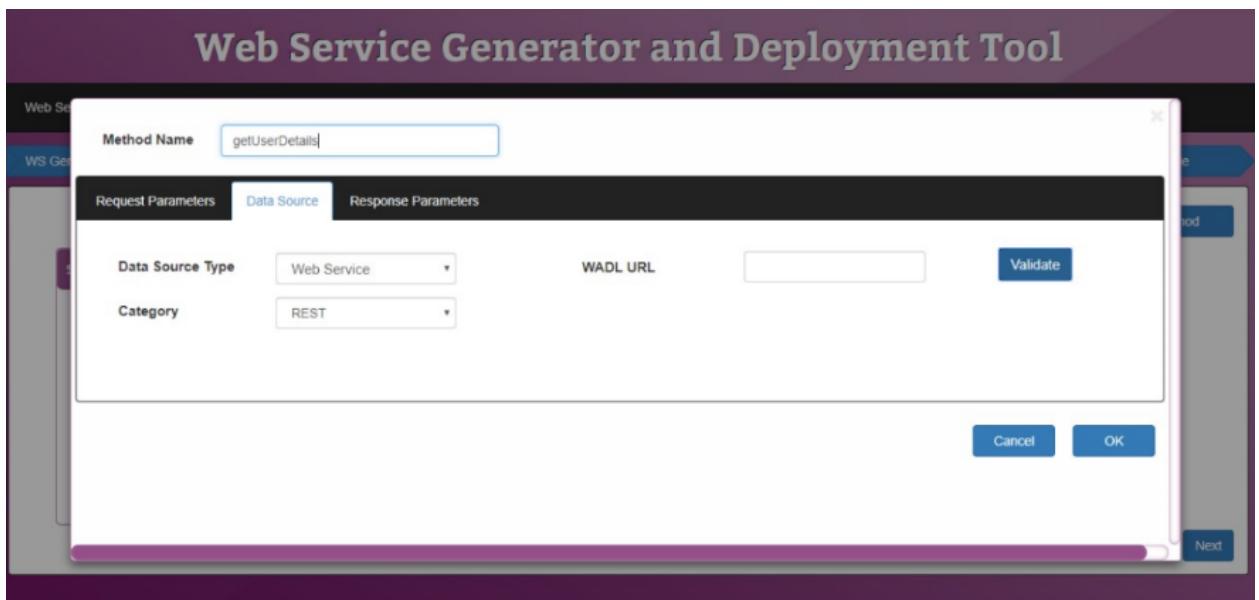
SI No.	Name	Data Type	Nullable
1	userID	Integer	No

Cancel OK

Screen-3 shall allow the user to add the request parameters. If the Data Source Type is Web Service then the Request Parameters shall be populated accordingly based on the arguments of the web method selected.

### Screen-4.1 – Depicting REST

**Web Service Generator and Deployment Tool**



Method Name:

Request Parameters    Data Source    Response Parameters

Data Source Type:     Validate

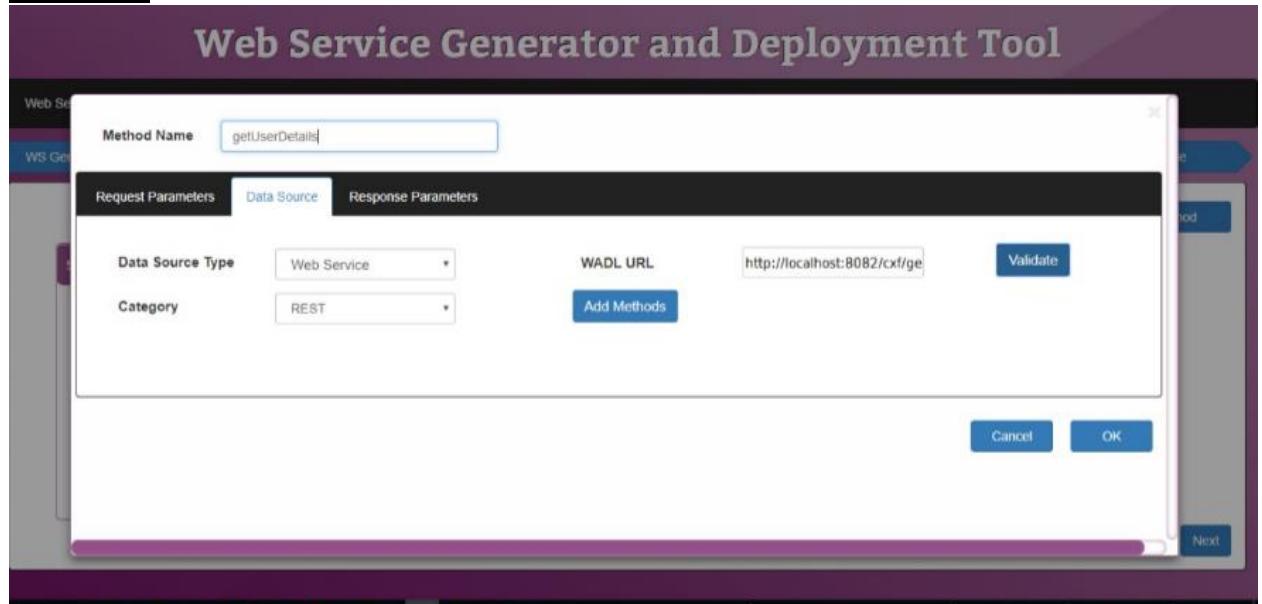
Category:

Cancel OK

Screen-4.1 Allows the user to select the Data Source Type where the user may select the Web services as the Data Source Type. The category shall be populated with REST / SOAP. The user shall be allowed to select any one of the two.

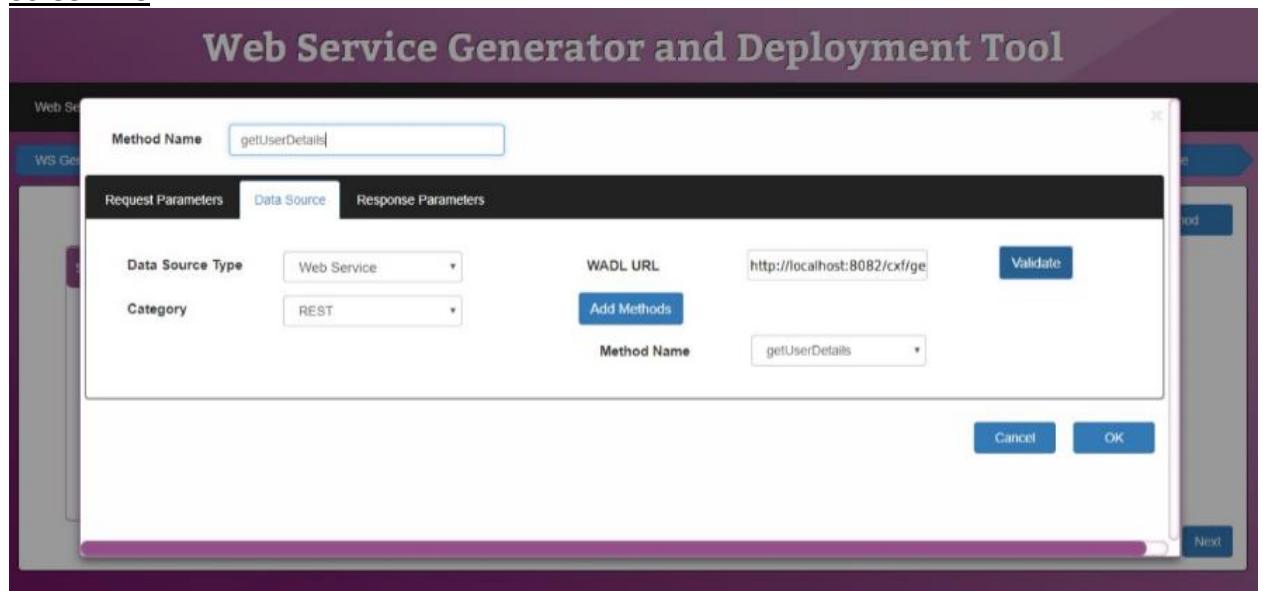
The user shall be allowed to enter the WADL URL if the selection in the Category is REST. There shall be a Validate button to validate the WADL URL.

#### **Screen-4.2**



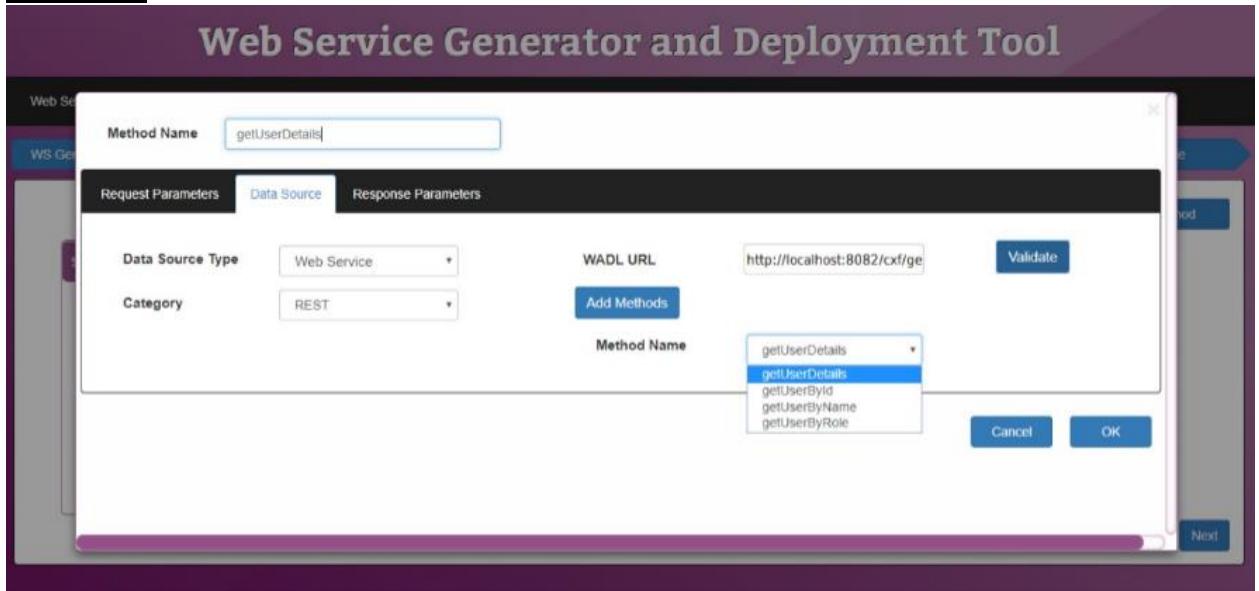
Screen-4.2 shall allow the user to add the Web Methods of the selected Web service by clicking the Add Method Button after the validation of the WADL URL comes out to be successful.

#### **Screen-4.3**



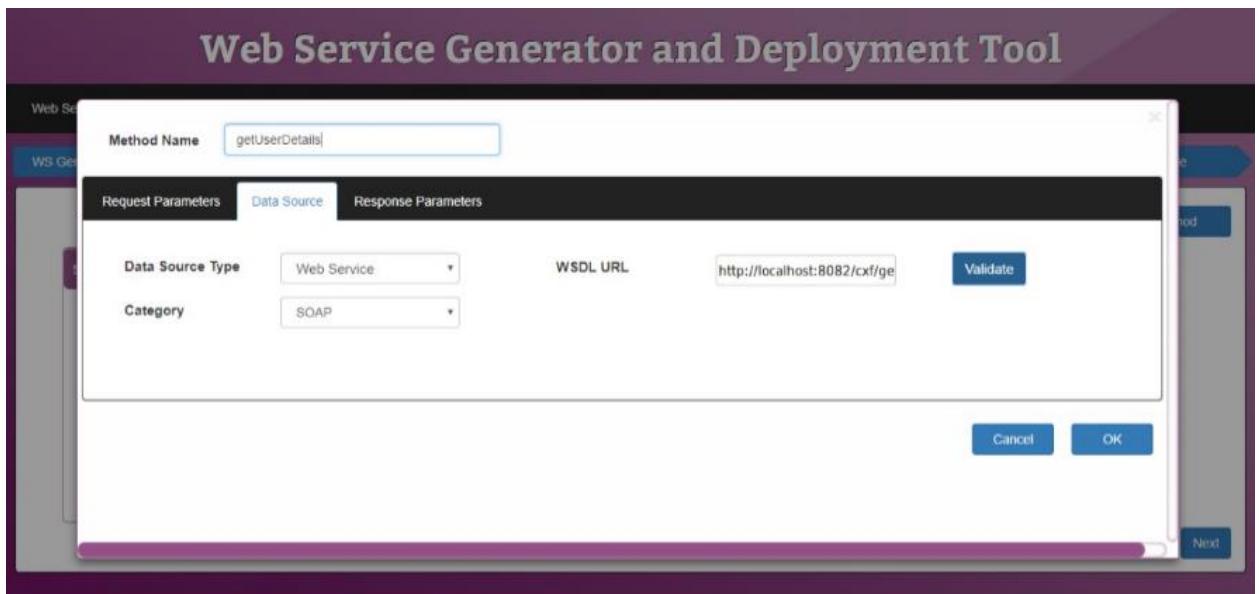
Screen-4.3 There shall be a Method Name dropdown which shall be populated with the Web methods of the selected Web Service.

#### Screen-4.4

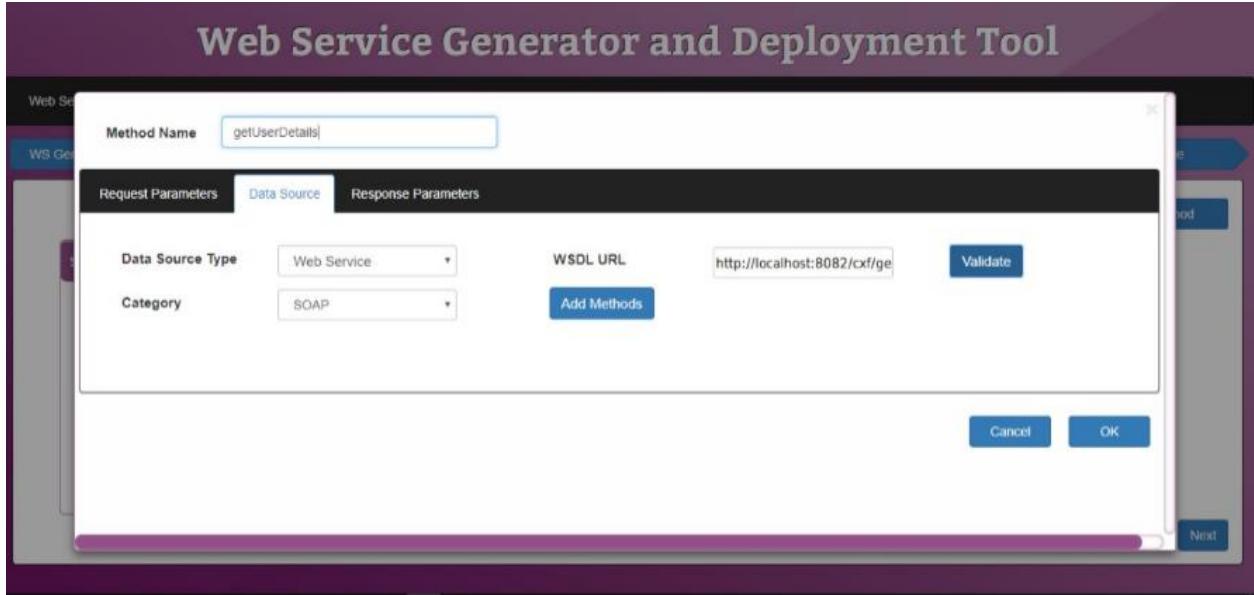


Screen-4.4-The user shall be allowed to select only one Web Method at any instant of time for the creation of the method. There shall be an optional Message Area to display all overloaded methods depending on the presence of overloaded methods.

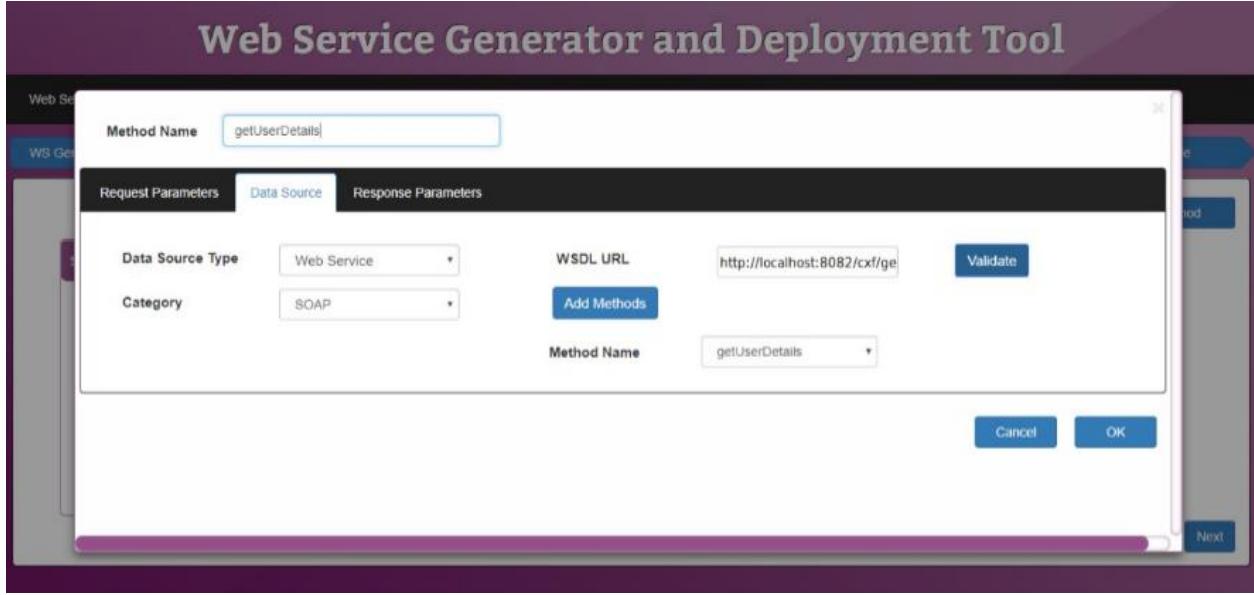
#### Screen-5- Depicting the SOAP



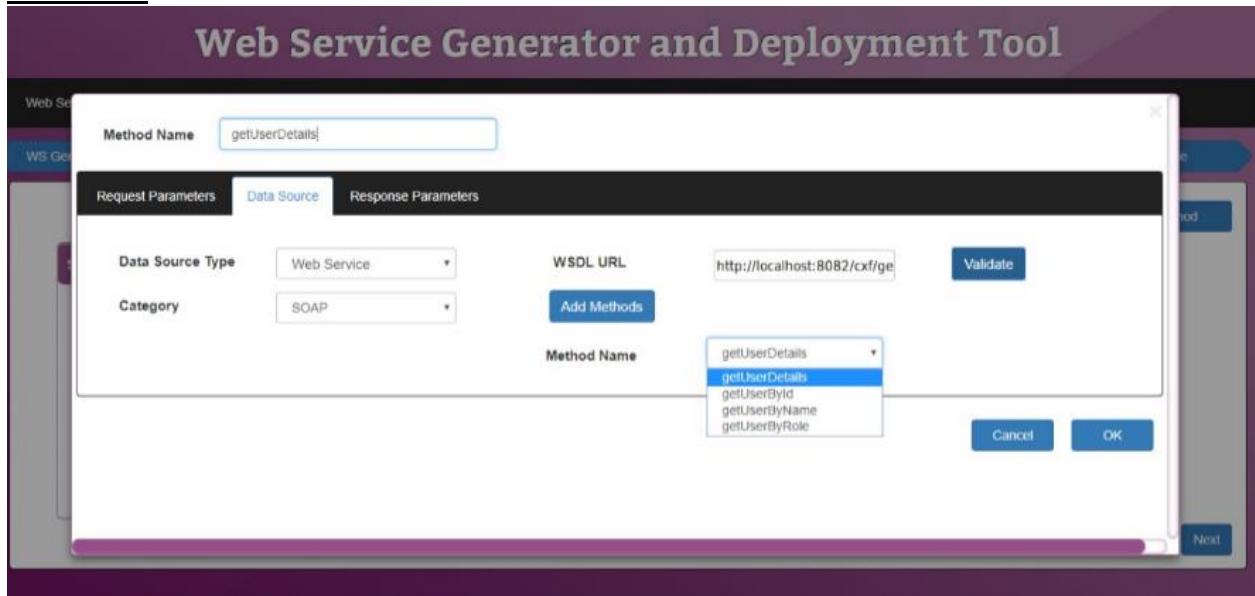
Screen-5 The user shall be allowed to enter the WSDL URL if the selection in the Category is SOAP.

**Screen-5.1**

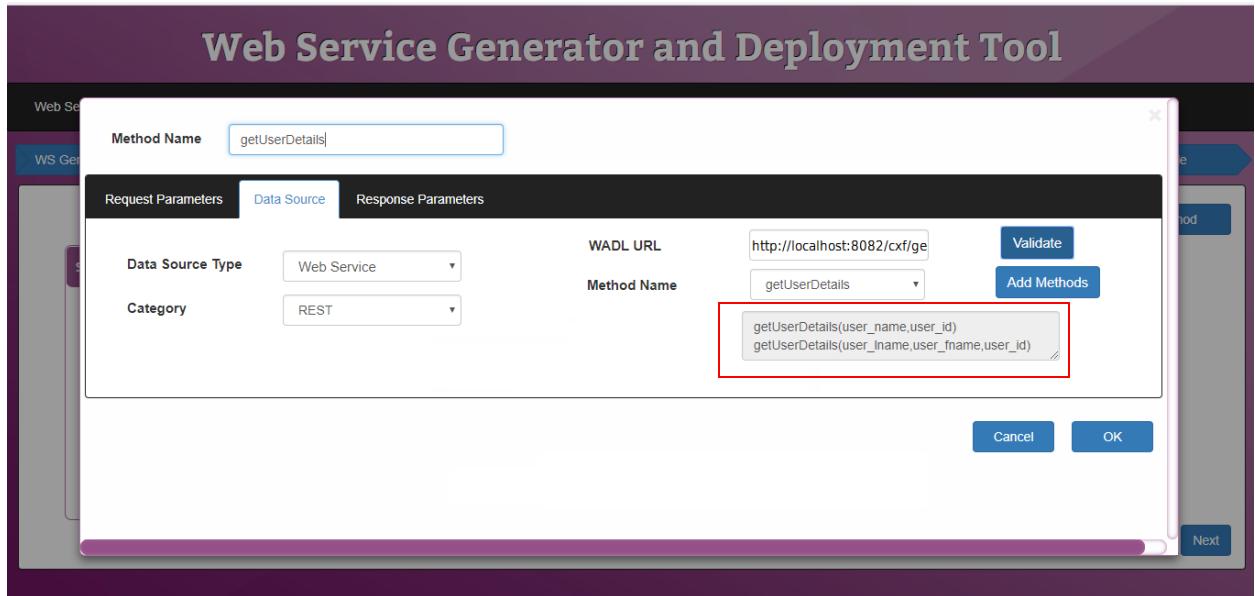
Screen-5.1 There shall be a Validate button to validate the WSDL URL.

**Screen-5.2**

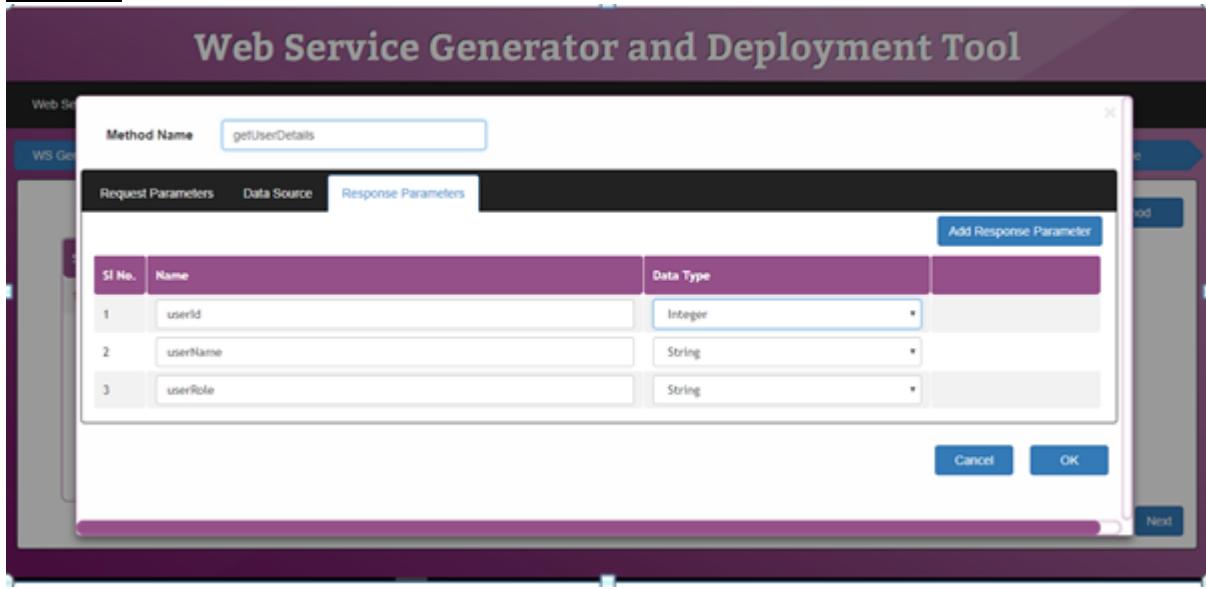
Screen-5.2 shall allow the user to add the Web Methods of the selected Web service by clicking the Add Method Button after the validation of the WSDL URL comes out to be successful.

**Screen-5.3**


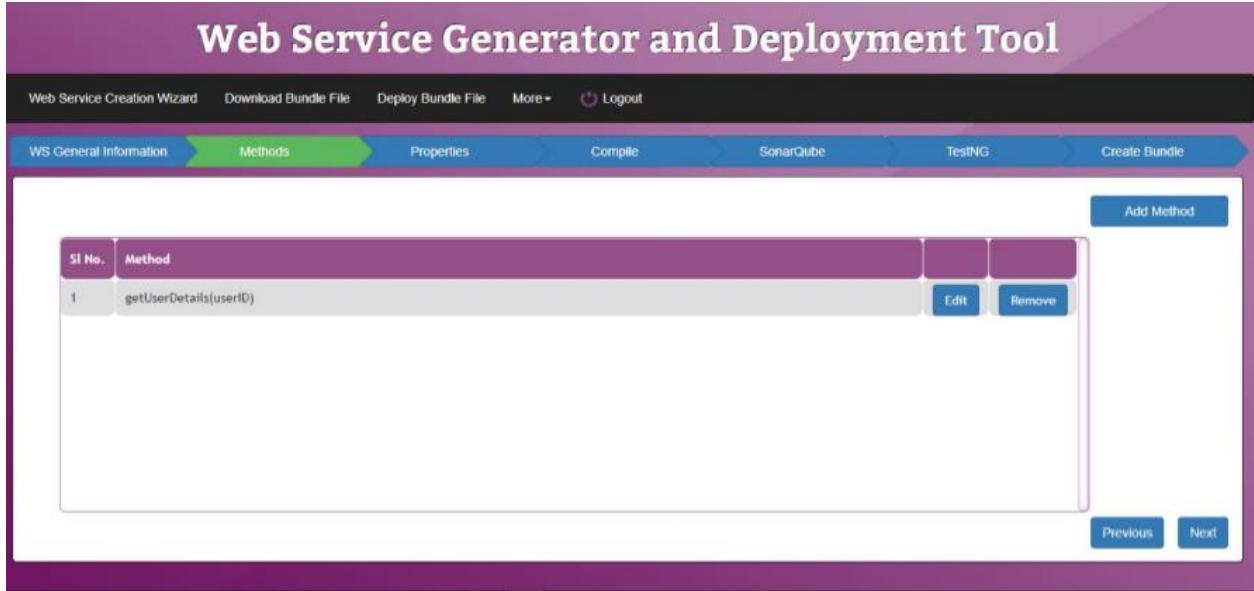
Screen-5.3 There shall be a Method Name dropdown which shall be populated with the Web methods of the selected Web Service.

**Screen-5.4**


Screen-5.4 There shall be an optional Message Area to display all overloaded methods depending on the presence of overloaded methods.

**Screen-6**


Screen-6 The user shall be able to view the Response parameters which may be populated based on the return type of web methods.

**Screen-7**


Screen 7- The Method(s) created shall be displayed in the method summary.

If the Web Method takes arguments then those may be treated as the Request Parameters.

The return values of the Web Method may be treated as the Response parameters.

### 3.1.2 Hardware Interfaces

**Operating System:** Windows 7 and 10/Linux (Ubuntu 16.04 LTS)

**RAM:** Minimum 8GB

**Processor:** Core i5

**Hard Disk:** 256 GB or larger

### 3.1.3 Software Interfaces

Java 8, Apache Maven 3.5.3 , Postgres SQL 10.3, MySQL 5.7.21, MSSQL 14.0, Oracle 12.2.0.1, Sonar 6.5 ,Jboss-fuse-6.3.0. redhat-187

## 3.2 Functional Requirements

### 3.2.1 Web Service Generation Tool

3.2.1.1 The system shall provide a wizard-based tool to Generate Web Services

3.2.1.2 The Wizard shall have the following main sections

- a. General
- b. Method General Info
- c. Method Data Source
- d. Method Response Parameters
- e. Query Viewer (For Database Tables only)
- f. Property File Generator
- g. Code Compiler
- h. Code Analysis (SonarQube)
- i. Automation (TestNG framework)
- j. Bundle Creation

### 3.2.2 Wizard General Page: Web Service General Information

3.2.2.1 The Wizard General Page shall allow the user to enter the top level basic details of the Web Service.

3.2.2.2 The General Page shall allow the user to enter **Project Name**.

3.2.2.2.1 There shall be an input text box with label name Project Name in the screen, where user can enter the Project Name.

3.2.2.2.2 The Project Name shall be a mandatory field. The maximum length of the field shall be 25 characters. No space shall be allowed within the Project Name.

3.2.2.2.3 Based on the entered Project Name the Project shall be created at the back end.

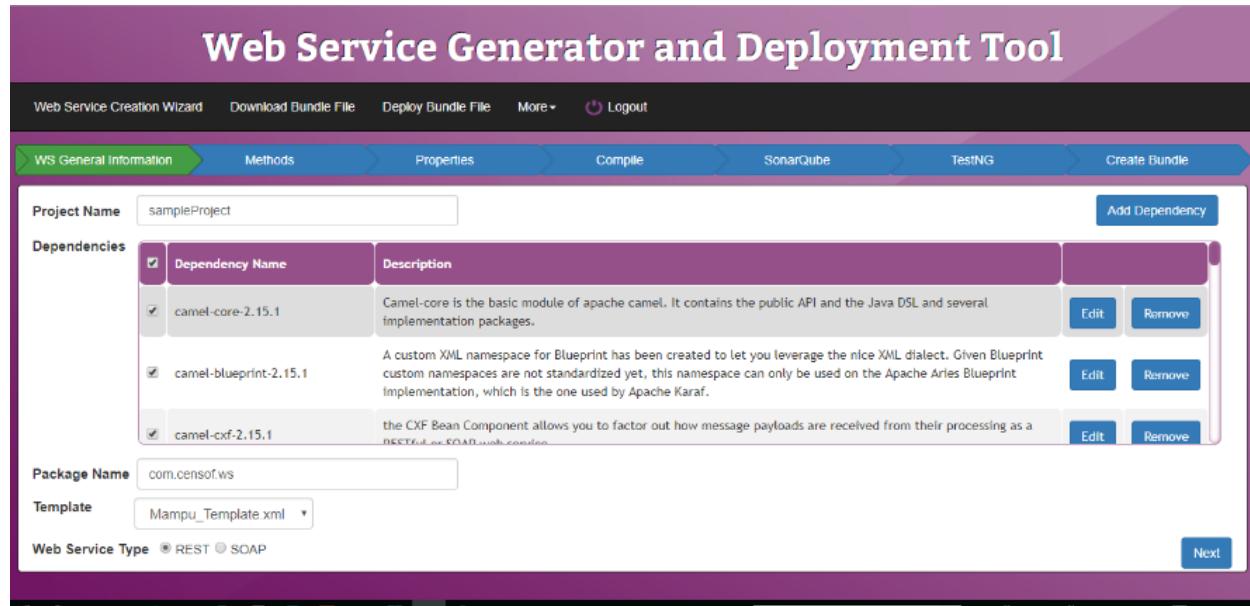
3.2.2.3 The General Page shall allow the user to select the **web service type**.

3.2.2.3.1 There shall be a radio button to select either Rest or SOAP as web service type.

- 3.2.2.4 The Wizard General Page shall allow the user to select the **Dependencies** for the Web Service. The add edit and delete dependencies shall be maintained from the dependencies Table itself in the Wizard General Page.
  - 3.2.2.4.1 The multiple check boxes containing the Dependencies shall be populated from the database.
  - 3.2.2.4.2 The multiple check boxes shall display the Dependency name, Dependency description and version.
  - 3.2.2.4.3 The user shall be allowed to select multiple Dependencies.
  - 3.2.2.4.4 For the selected dependencies a pom.xml file shall be generated at the back end.
- 3.2.2.5 The Wizard General Page shall allow the user to enter Package Name for the Web Service. Package name should have minimum 3 dots and it should start with domain names com, edu, gov, net and org and it should have 3 characters in between the dots.
  - 3.2.2.5.1 There shall be an input text box with label name Package Name in the screen where user can enter the Package Name.
  - 3.2.2.5.2 The Package Name shall be a mandatory field.
  - 3.2.2.5.3 At the back end the package shall be created and all the generated classes/interfaces shall be in the specified package.
- 3.2.2.6 The Wizard General Page shall allow the user to select the **Predefined Template** for the Web Service.
  - 3.2.2.6.1 There shall be a label named Predefined Template
  - 3.2.2.6.2 There shall be a drop-down option to select Predefined Templates.
  - 3.2.2.6.3 The drop down shall be populated with predefined templates from the configured location.
  - 3.2.2.6.4 There shall be a screen to upload the template. If the template name is already available then the file name shall be suffixed with increment numbers.
  - 3.2.2.6.5 Template shall be displayed in dropdown. User can access the old template from the dropdown.
  - 3.2.2.6.6 The user shall be allowed to select any one Predefined Template from the drop down.
  - 3.2.2.6.7 The selection of Predefined Template by the user shall be mandatory.
  - 3.2.2.6.8 In the back end the source code shall be generated based on the selected predefined template. The tool shall be going to generate the header.java file and the processor java file to validate the payload structure and to validate the header parameters like mandatory fields, allowing only numbers, allowing only alphanumeric.
  - 3.2.2.6.9 The Predefined Template shall contain the following information in xml format for each method. Template shall be used to share the common functionality for all the methods. There shall be a screen to upload the template. Once the template is uploaded, the uploaded template shall be shown in the dropdown. It is mandatory to select the template to generate the web services.

3.2.2.6.10 The format of the template is xml. – Please refer to Appendix1

3.2.2.6.11 Web Service General Information Wizard Page shall have the following layout.



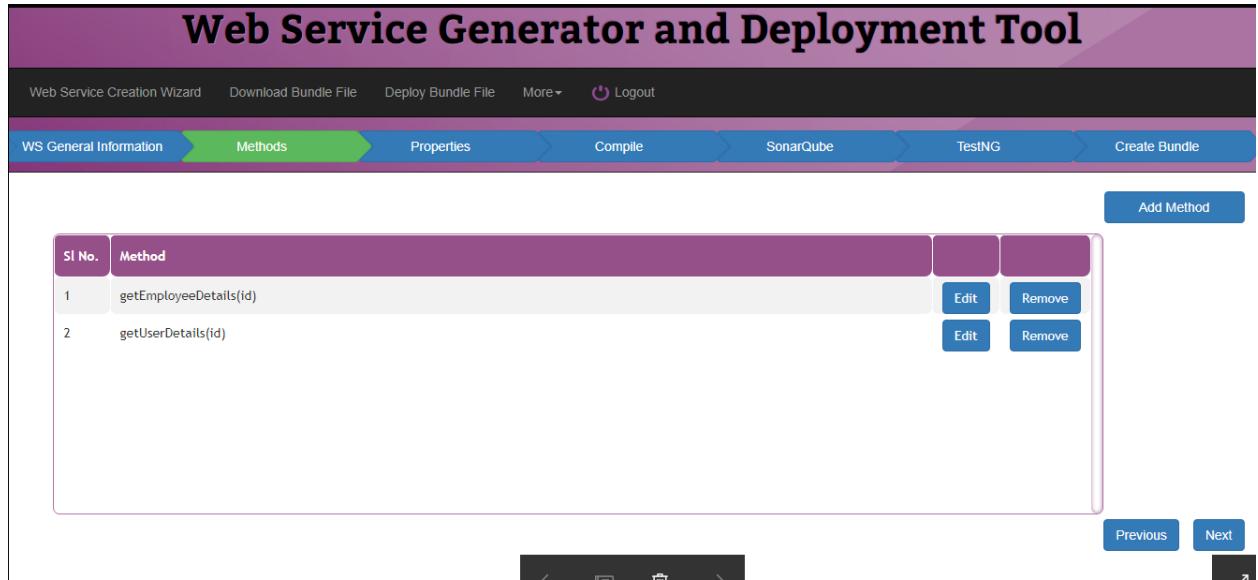
Dependency Name	Description	Edit	Remove
camel-core-2.15.1	Camel-core is the basic module of apache camel. It contains the public API and the Java DSL and several implementation packages.	Edit	Remove
camel-blueprint-2.15.1	A custom XML namespace for Blueprint has been created to let you leverage the nice XML dialect. Given Blueprint custom namespaces are not standardized yet, this namespace can only be used on the Apache Aries Blueprint Implementation, which is the one used by Apache Karaf.	Edit	Remove
camel-cxf-2.15.1	the CXF Bean Component allows you to factor out how message payloads are received from their processing as a DIRECT or LOAD_BALANCE endpoint.	Edit	Remove

### 3.2.3 Wizard Method General Info Page

3.2.3.1 The Wizard Method General Page shall allow the user to enter the top level basic details of **Method Name** and **Request Parameters** of the Web Service

- 3.2.3.1.1 Wizard Method General Page shall allow the user to enter **Method Names**
- 3.2.3.1.2 Wizard Method General Page shall display the Methods created in a tabular view with Method Names along with Request parameters.
- 3.2.3.1.3 The user shall be able to Remove or edit a Method by selecting the Method.
- 3.2.3.1.4 The Wizard Method General Page shall contain a Button to add Method Names.
- 3.2.3.1.5 The Button shall have the caption Add Method Name.
- 3.2.3.1.6 The user shall be allowed to add multiple Method Names.
- 3.2.3.1.7 The user shall be allowed to click the Add Method Name Button to add a new Method Name.
- 3.2.3.1.8 When the user clicks the Add Method Name Button a child screen shall open up with an input text box with label name Method Name in the screen.
- 3.2.3.1.9 The Method Name shall be a mandatory field.
- 3.2.3.1.10 The user shall add at least one Method Name for compilation.
- 3.2.3.1.11 The child screen shall also contain a multi tab container each for **Request Parameters, Data Source and Response Parameters**.
- 3.2.3.1.12 In the back-end, methods shall be created for each method name entered by the user.

### 3.2.4 Screen



The screenshot shows the 'Methods' tab of the 'Web Service Generator and Deployment Tool'. The top navigation bar includes links for 'Web Service Creation Wizard', 'Download Bundle File', 'Deploy Bundle File', 'More...', and 'Logout'. Below the navigation is a horizontal breadcrumb menu with tabs: 'WS General Information', 'Methods', 'Properties', 'Compile', 'SonarQube', 'TestNG', and 'Create Bundle'. The 'Methods' tab is currently active, indicated by a green background. The main content area displays a table with two rows of methods:

Sl No.	Method		
1	getEmployeeDetails(id)	<button>Edit</button>	<button>Remove</button>
2	getUserDetails(id)	<button>Edit</button>	<button>Remove</button>

At the bottom right of the table are 'Previous' and 'Next' buttons. A vertical scroll bar is visible on the right side of the content area.

Once user click on Edit button, similar to add method screen will be opened with previously entered data in the screen. User can edit/modify the pre-populated data.

#### 3.2.4.1 The Wizard Method General Page shall allow the user to enter **Request Parameters** for the mentioned Method.

- 3.2.4.1.1 The Wizard Method General Page shall contain a Button to add Request Parameters Names.
- 3.2.4.1.2 The Button shall have the caption Add Parameter.
- 3.2.4.1.3 The user shall be allowed to add multiple Request Parameters.
- 3.2.4.1.4 The user shall be allowed to click the Add Request Parameter Button to add a new Request Parameter.
- 3.2.4.1.5 When the user clicks the Add Parameter Button in the Request Parameter tab page a new row shall be created in the table with an input text box with column name Parameter Name in the screen dynamically created, where user can enter the Parameter Name.
- 3.2.4.1.6 The Parameter Name shall be a mandatory field.
- 3.2.4.1.7 Alongside with the Parameter Name input box there shall be one Drop Down for Data Type.
- 3.2.4.1.8 The user shall be allowed to select a valid Data Type from the Data Type drop down.
- 3.2.4.1.9 The valid Data Types shall include the following for multiple database support:
  - String
  - Boolean

- Integer
- Double
- Float
- Long
- Date Time
- Date
- Time

3.2.4.1.10 The selection of Data Type corresponding to Parameter Name shall be mandatory.

3.2.4.1.11 The user shall be allowed to Set Validation Rule corresponding to the Parameter name entered by the user.

3.2.4.1.12 The Setting of Validation Rule shall not be mandatory.

3.2.4.1.13 There may be multi-check box populated with Validation Rule.

3.2.4.1.14 The user may be allowed to check on the check boxes against each Validation Rule.

3.2.4.1.15 The user may also be allowed to select multiple Validation Rules.

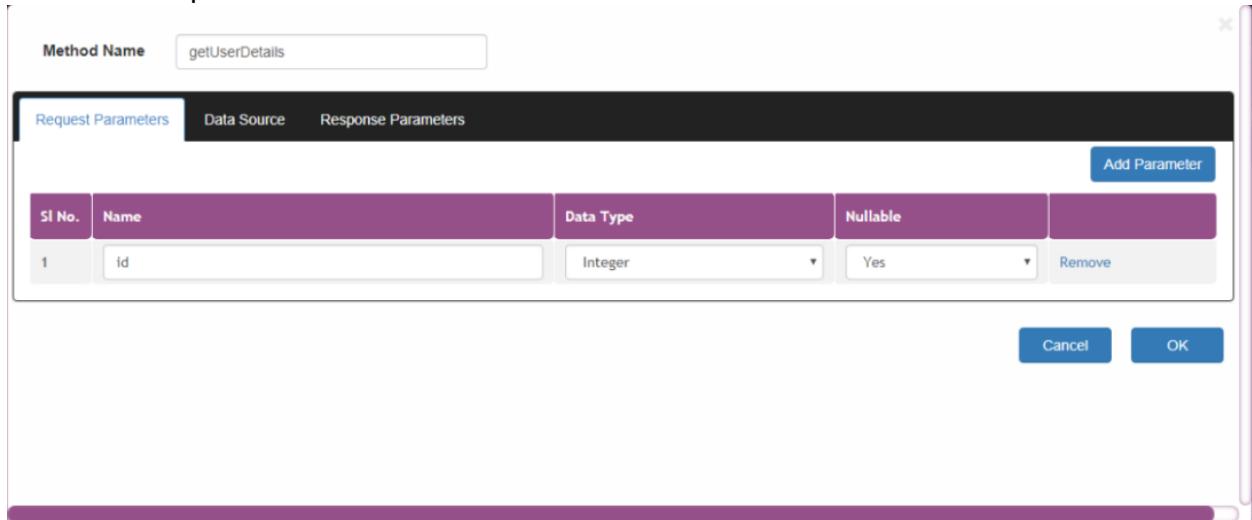
3.2.4.1.16 The validation rule for Request Parameters shall include:

- Not Null
- Minimum Length
- Maximum Length
- Only Number
- Alpha numeric

3.2.4.1.17 In the back-end request class shall be generated and this class shall have all the parameter names.

3.2.4.1.18 In the back-end source code shall be generated to validate the selected validations for the request parameters.

3.2.4.1.19 Screen for Request Parameter



SI No.	Name	Data Type	Nullable
1	id	Integer	Yes

### **3.2.5 Wizard Method Data Source Page**

3.2.5.1 The Wizard Method Data Source Page shall allow the user to enter the top level basic details of **Data Sources** of the Web Service.

3.2.5.1.1 In this screen all the entered method names shall be displayed to select the data source details.

3.2.5.1.2 Each method shall be mapped with one data source

3.2.5.1.3 The Wizard Method Data Source Page shall allow the user to select **Data Source Type**.

3.2.5.1.3.1 There shall be a drop-down option with label name Data Source Type.

3.2.5.1.3.2 The selection of the Data Source Type by the user shall be mandatory.

3.2.5.1.3.3 The valid contents of the Data Source Type drop down are:

- Data Base Table (Postgres, MySQL, MSSQL, Oracle)
- Web Service

3.2.5.1.3.4 The Data Source Type shall be a mandatory field.

3.2.5.1.4 The Wizard Method Data Source Page shall allow the user to select **Data Source Category**

3.2.5.1.4.1 There shall be a drop-down option with label name Data Source Category.

3.2.5.1.4.2 The user shall be allowed to select the Data Source Category.

3.2.5.1.4.3 Depending on the selection of the Data Source Type the Data Source Category shall be populated with contents dynamically.

3.2.5.1.4.4 The valid contents of the Data Source Category drop down for “DB Table” are

- Postgres
- MySQL
- Oracle
- MS SQL

3.2.5.1.4.5 The valid contents of the Data Source Category drop down for “Web Service” are

- SOAP
- Rest

3.2.5.1.4.6 The Data Source Type shall be a mandatory field

3.2.5.1.5 The Wizard Method Data Source Page shall allow the user to enter **Data Source Name**

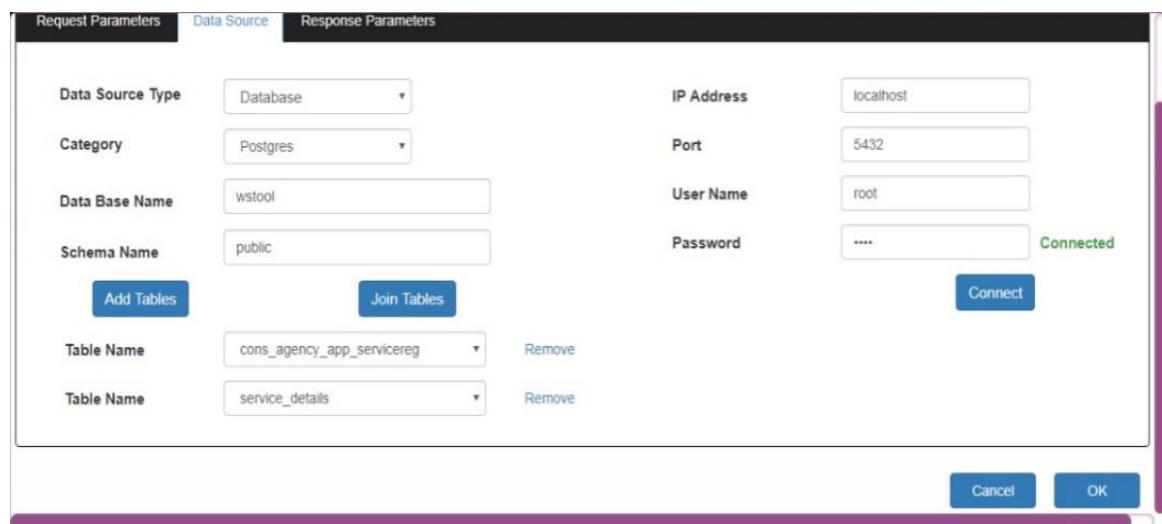
3.2.5.1.5.1 There shall be an input text box with label name Data Source Name in the screen, where user can enter the Data Source Name

3.2.5.1.5.2 The Data Source Name shall be a mandatory field

3.2.5.1.5.3 The user shall be allowed to enter the Data Source Name depending on the selection of the Data Source Type and Category.

3.2.5.1.5.3.1 If the user selects Database Table as Data Source Type and any one of Postgres, MySQL, Oracle or MS SQL as Data Source Category then the interface shall have a button named Add Data Source.

- 3.2.5.1.5.3.2 The user shall be allowed to add Data Source Name by Clicking the Button.
- 3.2.5.1.5.3.3 By clicking the Add Data Source Button the interface shall have two input boxes dynamically created one with label Database Name and the other with label Table Name.
- 3.2.5.1.5.3.4 The user shall be allowed to enter the Database Name and the Table Name in the respective input boxes.
- 3.2.5.1.5.3.5 If the user selects SOAP or Rest as Data Source Category then the user shall be provided with an input text box to enter web service address.
- 3.2.5.1.5.3.6 If the user selects CSV, WLS and XML as Data Source Category then the user shall be provided with an input text box to enter file name.
- 3.2.5.1.6 The Wizard Method Data Source Page shall allow the user to **Join Data Tables**.



Request Parameters		Data Source		Response Parameters	
Data Source Type	Database			IP Address	localhost
Category	Postgres			Port	5432
Data Base Name	wstool			User Name	root
Schema Name	public			Password	....
<b>Add Tables</b>		<b>Join Tables</b>	<b>Connect</b>		
Table Name	cons_agency_app_servicereg	Remove			
Table Name	service_details	Remove			
<input type="button" value="Cancel"/> <input type="button" value="OK"/>					

**Join Tables**

Sl No	Join Type	Left Table Name	Left Column Name	Right Table Name	Right Column Name	Remove
1	INNER JOIN OUTER JOIN LEFT JOIN RIGHT JOIN FULL JOIN	cons_agency_app_s		cons_agency_app_s		<a href="#">Remove</a>

[Add Joins](#) [Cancel](#) [OK](#)

**Join Tables**

Sl No	Join Type	Left Table Name	Left Column Name	Right Table Name	Right Column Name	Remove
1	LEFT JC	service_details	service_id	cons_agency_app_se	service_id	<a href="#">Remove</a>

[Add Joins](#) [Cancel](#) [OK](#)

### 3.2.6 Wizard Method Response Parameters Page

3.2.6.1 The Wizard Method General Page shall allow the user to enter the top level basic details of **Response Parameters** of the Web Service.

3.2.6.1.1 The Wizard Method General Page shall contain a Table to add Response Parameters Names.

3.2.6.1.2 The user shall be allowed to add multiple Response Parameters.

3.2.6.1.3 The user shall be able to provide the **Parameter Name** in an input text box with column caption **Field Name**.

3.2.6.1.4 Alongside with the Parameter Name input column in the next column there shall be one Drop Down with column caption **Data Type**.

3.2.6.1.5 The user shall be allowed to select a valid Data Type from the Data Type drop down.

3.2.6.1.6 The valid Data Types shall include:

- String
- Boolean
- Integer
- Double
- Float
- Long
- Date Time
- Date
- Time

3.2.6.1.7 The selection of Data Type corresponding to Parameter Name shall be mandatory.

3.2.6.1.8 Next to the Data Type column the user shall be able to select Table Name from a drop down in that column with caption **Table Name**

3.2.6.1.9 The user shall be allowed to select the **Mapped Source Data Field** from the dropdown Filed names of respective selected Table of that particular row.

3.2.6.1.10 The user shall be allowed to map each response parameter with Source Data Field.

3.2.6.1.11 The Mapped Source Data Field drop down shall be populated from the fields of the Data Source Type.

3.2.6.1.12 The mapping of the Source Data Field is mandatory.

3.2.6.1.13 Adjacent to the Mapped Source Data field there shall be a **visible** column with drop-down.

3.2.6.1.14 The valid Visible values are:

True and False

Visible – **True** implies that the user shall be able to see the respective field in the query output.

Visible – **False** implies that the user shall not be able to see the respective field in the query output. It is only used to build the query.

3.2.6.1.15 The visible Field Names shall be treated as **Response Parameters**.

3.2.6.1.16 The Field names which are not visible shall be used internally to build up the Query.

3.2.6.1.17 The user shall be allowed to apply **Transformation**.

3.2.6.1.17.1 The valid Transformations are

- SUM
- AVERAGE

3.2.6.1.18 The user shall be allowed to apply **Fetch Condition**.

3.2.6.1.19 The Fetch Condition shall include multiple Request Parameters

3.2.6.1.20 The Fetch Condition shall be based on

3.2.6.1.20.1 Aggregate Function shall include

- <none>
- Count

- Count(Distinct)
- Max
- Min

#### 3.2.6.1.20.2 Where Condition shall include

- =
- !=
- <
- <=
- >
- >=
- BETWEEN
- EXISTS
- IN
- IS NOT NULL
- IS NULL
- LIKE
- NOT BETWEEN
- NOT IN
- NOT LIKE

#### 3.2.6.1.20.3 The user may also be allowed to use the Group By, Having Condition, Sort By, Order By clauses.

3.2.6.1.21 The Response Parameter tab page shall act as the Query Builder by itself.

3.2.6.1.22 The Response Parameter tab page shall contain a Button with caption **View Query**.

3.2.6.1.23 The user shall click the View Query button to view the Query formed.

3.2.6.1.24 The Response Parameter tab page shall have the following layout.

**Step 2 Methods**

Method Name:

**Request Parameter**   **Data Source**   **Response Parameters**

Sl	Field Name	Data Type	Table Name	Mapped Source Data Field	Visible	Aggregate	Where Condition	OR	Group By	Having	Order By	Sort
1	Dept_Id	Integer	TBL_DEPT	dept_id	FALSE		= [ID]		Group(1)			
2	Department Name	String	TBL_DEPT	dept_name	TRUE							
3	SALARY	Decimal	TBL_EMPSAL_MONT	Salary	TRUE	AVG						
4	Dept_Id	Integer	TBL_EMPLOYEE	dept_id	FALSE		=TBL_DEPT.Dept_Id					
5	Active	Boolean	TBL_EMPLOYEE	Active	FALSE		= [TRUE]					
6	Date	Date	TBL_EMPSAL_MONT	Date	FALSE		BETWEEN/[Start Date] AND [End Date]					

**View Query**   **OK**   **Cancel**

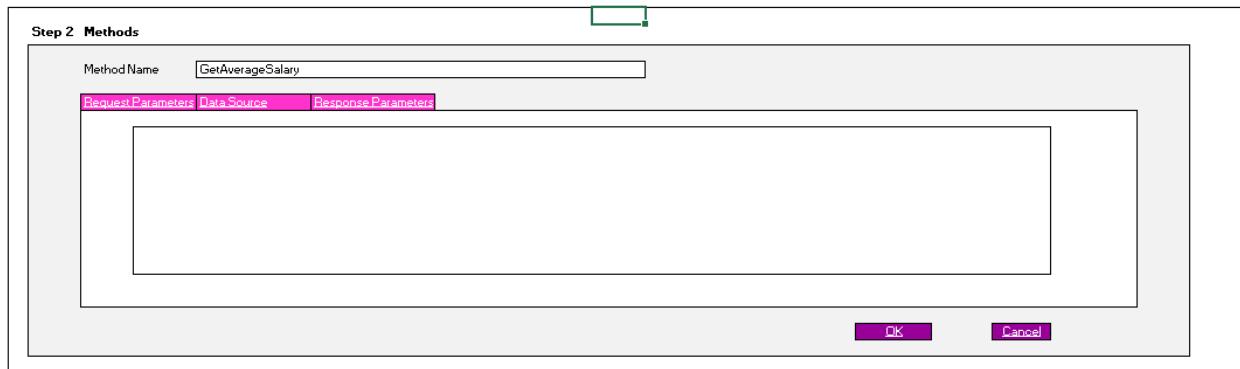
#### 3.2.7 Wizard Query Viewer Page

3.2.7.1 The Wizard Query Viewer Page shall allow the user to see the generated **Query**.

3.2.7.2 The generated query shall be displayed in a Rich Text Box / Multiline Text Box.

3.2.7.3 The Rich Text Box / Multiline Text Box shall be read only.

3.2.7.4 The Query Viewer Page shall have the following layout.

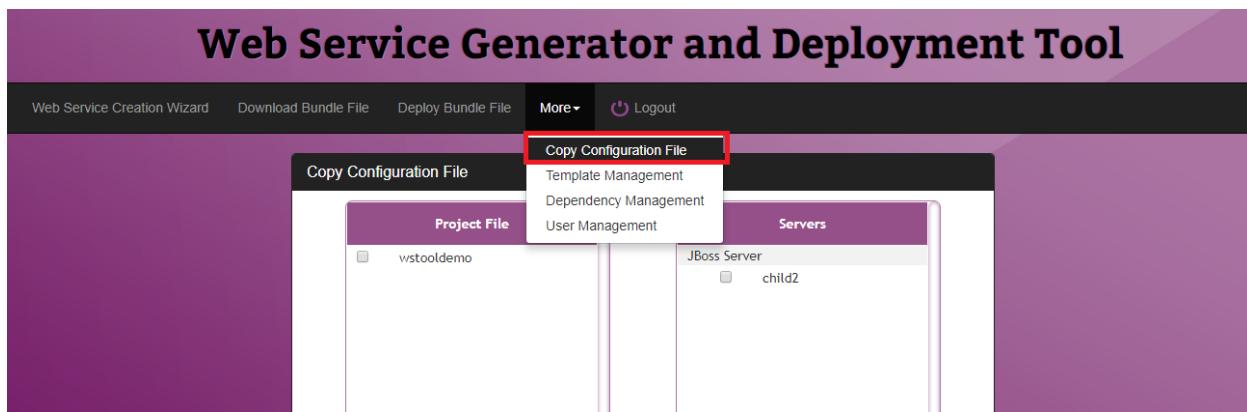


### 3.2.8 Wizard Property File Generator Page

3.2.8.1 The Wizard Property File Generator Page shall allow the user to **Specify Property Parameters**

To copy the generated configuration files in the server, user need to click on Copy Configuration File under More menu.

In this screen user need to select the list of projects and the list servers where the generated configuration files need to be copied.



3.2.8.2 In this screen user shall be allowed to enter all the database configuration properties like driver name, user name, password, database name, url. Driver name will not be changed. Latest driver will be automatically updated by updated dependency jar file version.

3.2.8.3 Based on the entered database configuration properties, the database config properties file shall be generated at the back end.

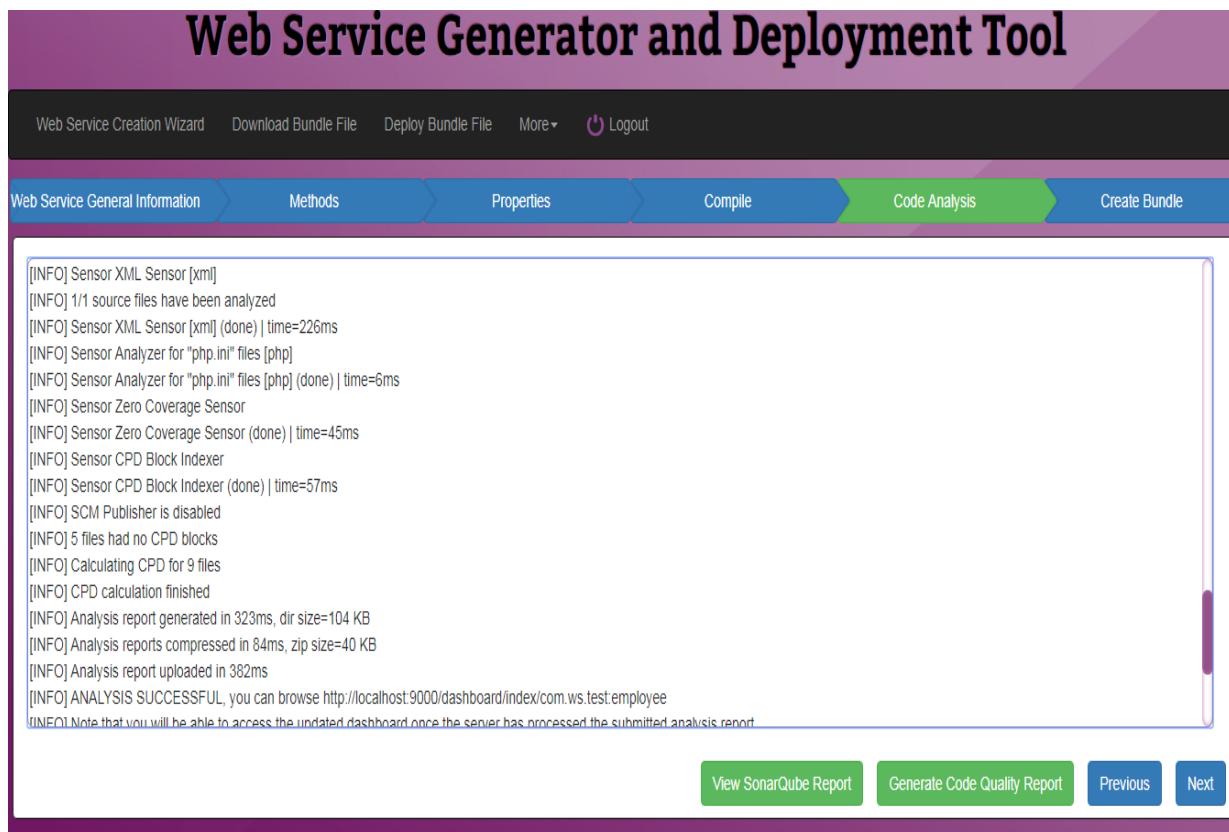
3.2.8.3.1 The user shall be allowed to View and Edit the Property Parameters from the Property File Generated by the code.

### **3.2.9 Wizard Code Compiler Page**

- 3.2.9.1 The Wizard Code Complier Page shall allow the user to Compile the Result.
- 3.2.9.2 All the generated java files shall be compiled and the compiled console log details shall be displayed in the screen with build status.
- 3.2.9.3 If the compilation is successful, then at the end the user shall see Build Success.
- 3.2.9.4 If the compilation failed, then at the end user shall see Build Failure.

### **3.2.10 Wizard Code Analysis Page (SonarQube)**

- 3.2.10.1 The Wizard Code Analysis Page shall allow the user to perform the code quality analysis by using SonarQube.
- 3.2.10.2 SonarQube shall check code quality and analyze the generated source code.
- 3.2.10.3 User need to click on SonarQube Report button to see the code quality report in the SonarQube dashboard.



The screenshot shows the 'Code Analysis' step of the wizard. It displays a list of informational messages from the SonarQube analysis process:

```

[INFO] Sensor XML Sensor [xml]
[INFO] 1/1 source files have been analyzed
[INFO] Sensor XML Sensor [xml] (done) | time=226ms
[INFO] Sensor Analyzer for "php.ini" files [php]
[INFO] Sensor Analyzer for "php.ini" files [php] (done) | time=6ms
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=45ms
[INFO] Sensor CPD Block Indexer
[INFO] Sensor CPD Block Indexer (done) | time=57ms
[INFO] SCM Publisher is disabled
[INFO] 5 files had no CPD blocks
[INFO] Calculating CPD for 9 files
[INFO] CPD calculation finished
[INFO] Analysis report generated in 323ms, dir size=104 KB
[INFO] Analysis reports compressed in 84ms, zip size=40 KB
[INFO] Analysis report uploaded in 382ms
[INFO] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard/index/com.ws.test.employee
[INFO] Note that you will be able to access the undated dashboard once the server has processed the submitted analysis report

```

At the bottom, there are buttons for 'View SonarQube Report' (green), 'Generate Code Quality Report' (green), 'Previous' (blue), and 'Next' (blue).

### **3.2.11 Wizard Running Automation TestNG**

- 3.2.11.1 The Wizard Running Automation TestNG Page shall allow the user to enter the test data for all the request parameters for each method.
- 3.2.11.2 There shall be a button with name Generate TestNG Scripts
- 3.2.11.3 User shall click on Generate TestNG Scripts button to generate the TestNG automation scripts.
- 3.2.11.4 There shall be a button with name Run TestNG Scripts

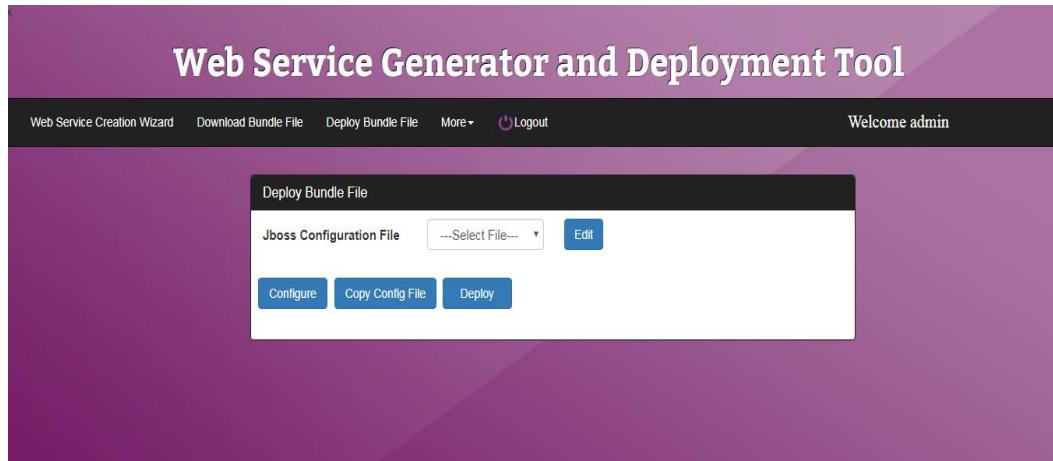
- 3.2.11.5 User shall click on Run TestNG Scripts button to run the generated TestNG scripts for entered test data.
- 3.2.11.6 After running the TestNG script the summary result shall be displayed in the Automation TestNG Page.

### **3.2.12 Wizard Bundle Creation Page**

- 3.2.12.1 The Wizard Bundle Creation Page shall allow the user to create Bundle.
- 3.2.12.2 The build console log details shall be displayed in the screen.
- 3.2.12.3 Build Success in the console log shall indicate that OSGI bundle file has been created successfully.
- 3.2.12.4 Build Failure in the console log shall indicate that OSGI bundle file has not been created.

## **3.3 Deployment**

- 3.3.1 The user shall be allowed to share the Test Deployment Link after Deployment
- 3.3.2 There shall be a screen to deploy the services in the JBoss server. We will be supporting only JBoss Server.
- 3.3.3 In this screen all the generated bundle files and list of servers shall be shown.
- 3.3.4 All the server details like server name and server path shall be saved in the DB.
- 3.3.5 The user shall be allowed to click the Deploy Bundle File menu.
- 3.3.6 Once user clicked the Deploy Bundle File menu then all the server details shall be received from DB.
- 3.3.7 User shall be allowed to select the list of bundle files and list of servers by clicking the checkboxes.
- 3.3.8 After selecting all the bundle files and list of servers the user shall be allowed to click on Deploy button to deploy all the selected bundle files in the selected servers.
- 3.3.9 Deployment Automation:
  - 3.3.9.1 User shall be allowed to enter the JBoss fuse commands in configuration file by selecting JBoss configuration file from dropdown.
  - 3.3.9.2 After saving the configuration the User shall click on the “Configure” button
  - 3.3.9.3 After Jboss server configuration is successful the User shall copy the configuration file generated by tool to JBoss fuse instance
  - 3.3.9.4 After Step (3.3.9.3) the user shall Click the “Deploy” button
  - 3.3.9.5 User shall get the URL which will be displayed on screen to access the list of services deployed in JBoss server



### 3.4 Upload Template Page

- 3.4.1 The upload template page shall allow the user to upload the predefined templates. If the template file is already exists, the incremental sequence number shall be maintained as version number and same shall be appended to the file name.
- 3.4.2 Uploaded template shall be saved in the configured location.

### 3.5 Generating WADL for Restful service

- 3.5.1 The user shall be allowed to configure the WADL url to access the WADL for the restful service in the blueprint .xml file.
- 3.5.2 The user shall be allowed to use the `http://<hostname>:<<portnumber>>/cxf` to see the list of services deployed in the JBoss server.
- 3.5.3 The user shall use `http://<<hostname>>:<<postnumber>>/<<servicename>>?_wadl` to access the WADL.
- 3.5.4 WADL shall have information about web service path, method type, list of parameters and its type, response type etc.

### 3.6 Generating WSDL for SOAP service

- 3.6.1 The user shall be allowed to configure the WSDL url to access the WSDL for the restful service in the blueprint .xml file.
- 3.6.2 The user shall be allowed to use the `http://<hostname>:<<portnumber>>/cxf` to see the list of services deployed in the JBoss server.
- 3.6.3 The user shall use `http://<<hostname>>:<<postnumber>>/<<servicename>>?_wsdl` to access the WSDL.
- 3.6.4 WSDL shall have information about web service path, method type, list of parameters and its type, response type etc.
- 3.6.5 Source file will be saved in local folder. User can access Source file from local folder

### 3.7 UI for testing web services (Rest and SOAP)

- 3.7.1 There shall be a screen where user can test the Restful and SOAP web services

- 3.7.2 <<Censof to share the source code to develop this feature.>> Censof shared static code which is not helpful for this project as expectation of Censof is different and is a dynamic one.
- 3.7.3 **This specific requirement shall be taken up later after the completion of Phase II as a separate change request as shared with Censof.**

### **3.8 Support for reading the parameters from WSDL and WADL**

- 3.8.1 The tool shall generate the source code dynamically for calling Restful web service based on WADL URL provided by user
- 3.8.2 The tool shall generate the source code dynamically for calling SOAP web service based on WSDL URL provided by user
- 3.8.3 The Request and Response parameters screens shall be a view only screen when the Data Source Type is selected as Web services.
- 3.8.4 The Request parameters screen shall display the arguments of the selected web method.
- 3.8.5 The Response parameters screen shall display the return parameter of the web method.
- 3.8.6 There shall be a drop down with value Web Service for Data Source under Data Source tab in Add Methods screen.
- 3.8.7 There shall be a drop with values Rest, SOAP for Category under Data Source tab in Add Methods screen.
- 3.8.8 User may be able to select either Rest or SOAP under Category drop down.
- 3.8.9 User may be able to provide the web service URL based on selected category type.
- 3.8.10 Creating client to call SOAP web services:
  - 3.8.10.1 The tool shall parse the WSDL URL to get the list of methods and corresponding to it the request parameter names, data types, method type and media type.
  - 3.8.10.2 The tool shall generate a source code to create a SOAP client dynamically based on the user provided data.
  - 3.8.10.3 The tool shall generate the source code to call the web service using created SOAP client by providing the URL, method name, method type.
- 3.8.11 UI integration with back end client code for SOAP web services:
  - 3.8.11.1 The tool may be able to modify the user interface to support the web service as data source type
  - 3.8.11.2 There shall be a input text box to enter the WSDL url
  - 3.8.11.3 The tool may be able to validate the entered WSDL url
  - 3.8.11.4 There shall be a drop down in the UI to display the methods available in the WSDL.
  - 3.8.11.5 There may be an optional message box to display all the overloaded methods.
  - 3.8.11.6 The user shall be allowed to select one method at any instant of time.
- 3.8.12 Creating client to call REST web services:
  - 3.8.12.1 The tool shall parse the WADL url to get the list of methods and corresponding to it the request parameter names, data types , method type and media type
  - 3.8.12.2 The tool shall generate a source code to create a REST client dynamically based on the user provided data.

- 3.8.12.3 The tool shall generate the source code to call the web service using created REST client by providing the URL, method name, method type.
- 3.8.13 UI integration with back end client code for REST web services
  - 3.8.13.1 The tool may be able to modify the user interface to support the web service as data source type
  - 3.8.13.2 There shall be a input text box to enter the WADL url
  - 3.8.13.3 The tool may be able to validate the entered WADL url.
  - 3.8.13.4 There shall be a drop down in the UI to display the methods available in the WADL.
  - 3.8.13.5 There may be an optional message box to display all the overloaded methods.
  - 3.8.13.6 The user shall be allowed to select one method at any instant of time.

**Web Service Generator and Deployment Tool**

Method Name:

Request Parameters    Data Source    Response Parameters

Data Source Type:     WSDL URL:     Validate

Category:

**Web Service Generator and Deployment Tool**

Method Name:

Request Parameters    Data Source    Response Parameters

Data Source Type:     WSDL URL:     Validate

Category:

**Web Service Generator and Deployment Tool**

Method Name:

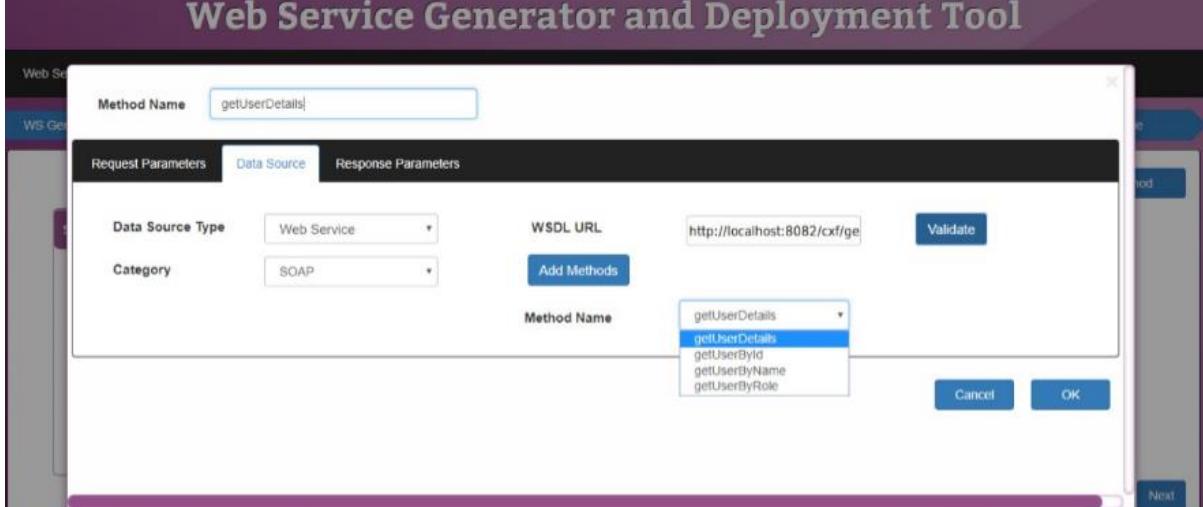
Request Parameters    Data Source    Response Parameters

Data Source Type:     WSDL URL:     Validate

Category:

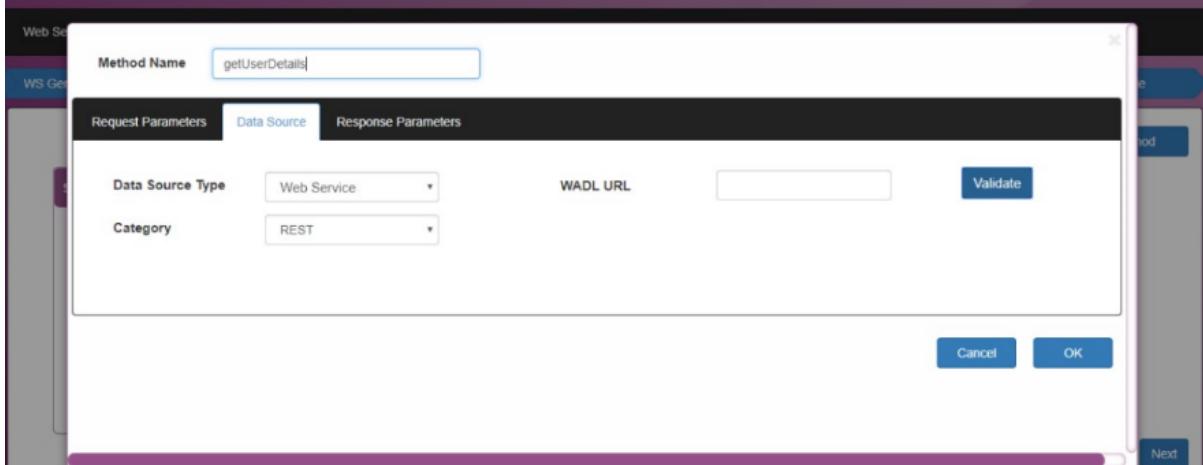
Method Name:

**Web Service Generator and Deployment Tool**



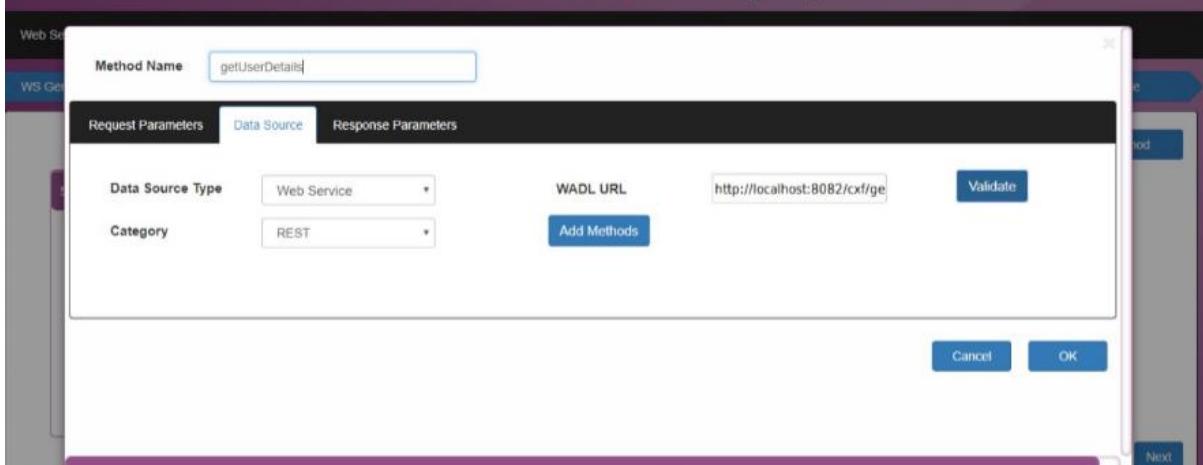
The screenshot shows the 'Data Source' tab of the tool. The 'Method Name' field contains 'getUserDetails'. The 'Data Source Type' dropdown is set to 'Web Service'. The 'Category' dropdown is set to 'SOAP'. The 'WSDL URL' field contains 'http://localhost:8082/cxf/ge'. A 'Validate' button is present. A dropdown menu for 'Method Name' is open, showing options: 'getUserDetails' (selected), 'getUserById', 'getUserByName', and 'getUserByRole'. Below the dropdown are 'Cancel' and 'OK' buttons.

**Web Service Generator and Deployment Tool**



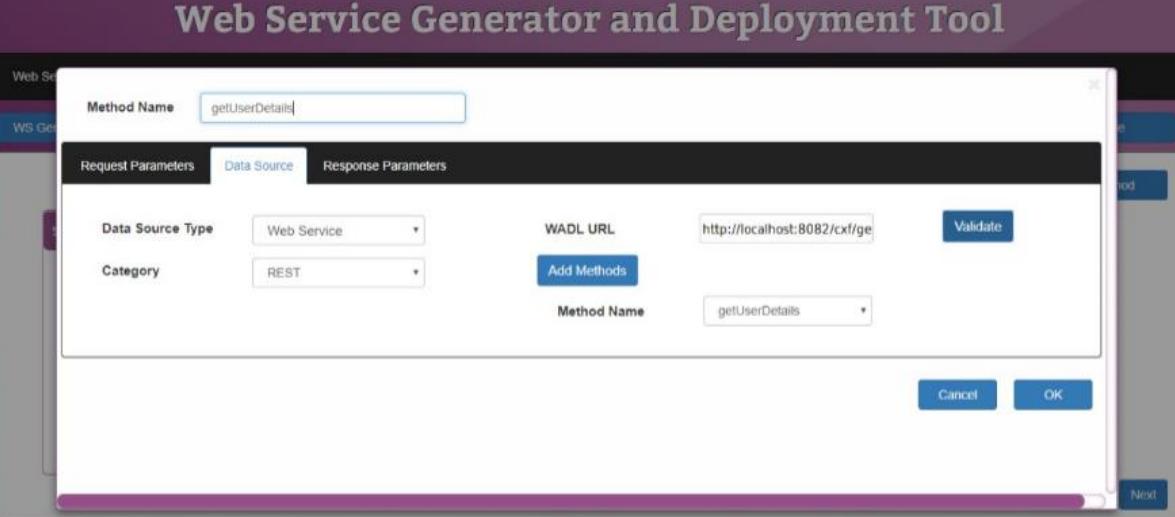
The screenshot shows the 'Data Source' tab of the tool. The 'Method Name' field contains 'getUserDetails'. The 'Data Source Type' dropdown is set to 'Web Service'. The 'Category' dropdown is set to 'REST'. The 'WADL URL' field is empty. A 'Validate' button is present. Below the field are 'Cancel' and 'OK' buttons.

**Web Service Generator and Deployment Tool**



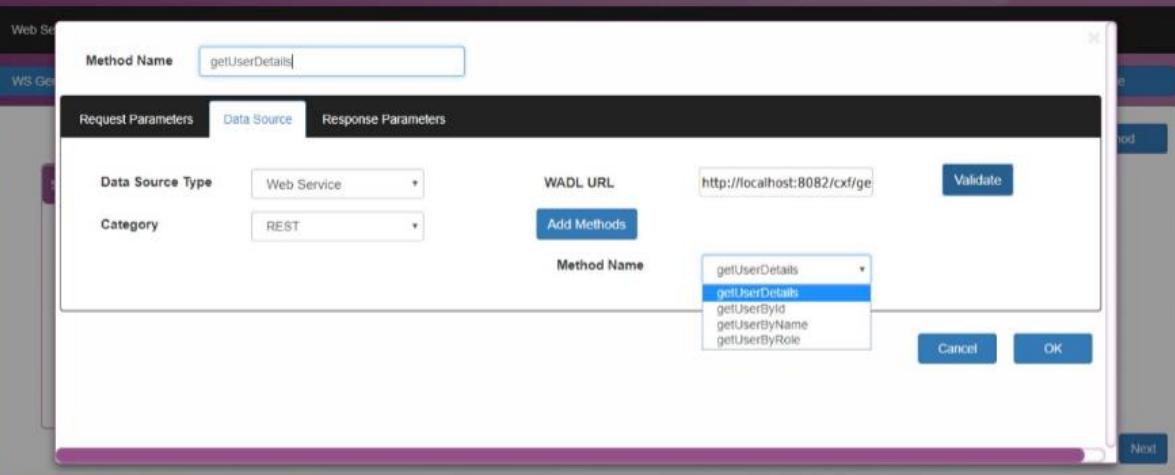
The screenshot shows the 'Data Source' tab of the tool. The 'Method Name' field contains 'getUserDetails'. The 'Data Source Type' dropdown is set to 'Web Service'. The 'Category' dropdown is set to 'REST'. The 'WADL URL' field contains 'http://localhost:8082/cxf/ge'. A 'Validate' button is present. A dropdown menu for 'Method Name' is open, showing options: 'getUserDetails' (selected), 'getUserById', 'getUserByName', and 'getUserByRole'. Below the dropdown are 'Cancel' and 'OK' buttons.

**Web Service Generator and Deployment Tool**



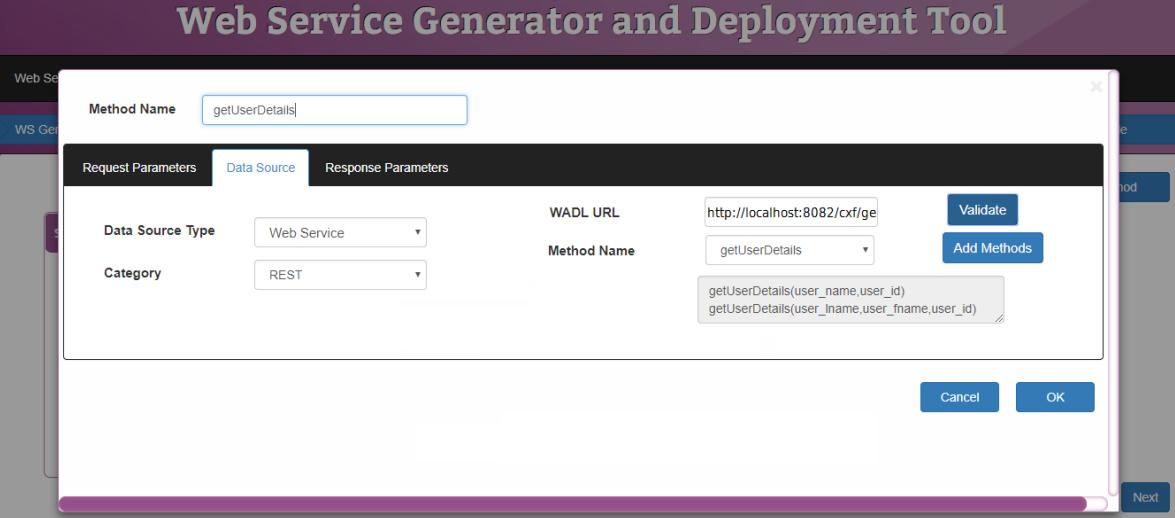
The screenshot shows the initial configuration screen of the tool. The 'Method Name' field contains 'getUserDetails'. The 'Data Source Type' is set to 'Web Service' and 'Category' is set to 'REST'. The 'WADL URL' field contains 'http://localhost:8082/cxf/ge'. A 'Validate' button is present. The 'Method Name' dropdown also lists 'getUserDetails'.

**Web Service Generator and Deployment Tool**



The screenshot shows the 'Method Name' dropdown open, listing several options: 'getUserDetails', 'getUserDetails', 'getUserById', 'getUserByName', and 'getUserByRole'. The first option is selected.

**Web Service Generator and Deployment Tool**



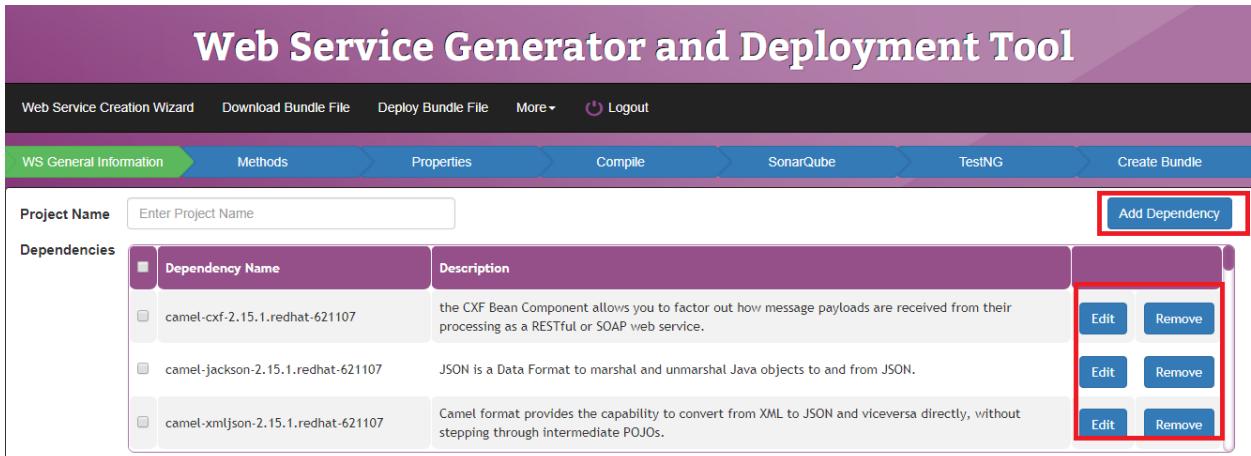
The screenshot shows the expanded configuration screen. The 'Method Name' dropdown now includes additional entries: 'getUserDetails(user\_name,user\_id)' and 'getUserDetails(user\_lname,user\_fname,user\_id)'. The 'Validate' and 'Add Methods' buttons are visible at the bottom.

### 3.9 Automated version control and backup

- 3.9.1 There shall be a button with name Code Backup in Create Bundle Wizard screen.
- 3.9.2 User shall click on Code Backup button to commit the generated source code to version control.
- 3.9.3 User shall provide Version Control URL, user name and password to commit the code.

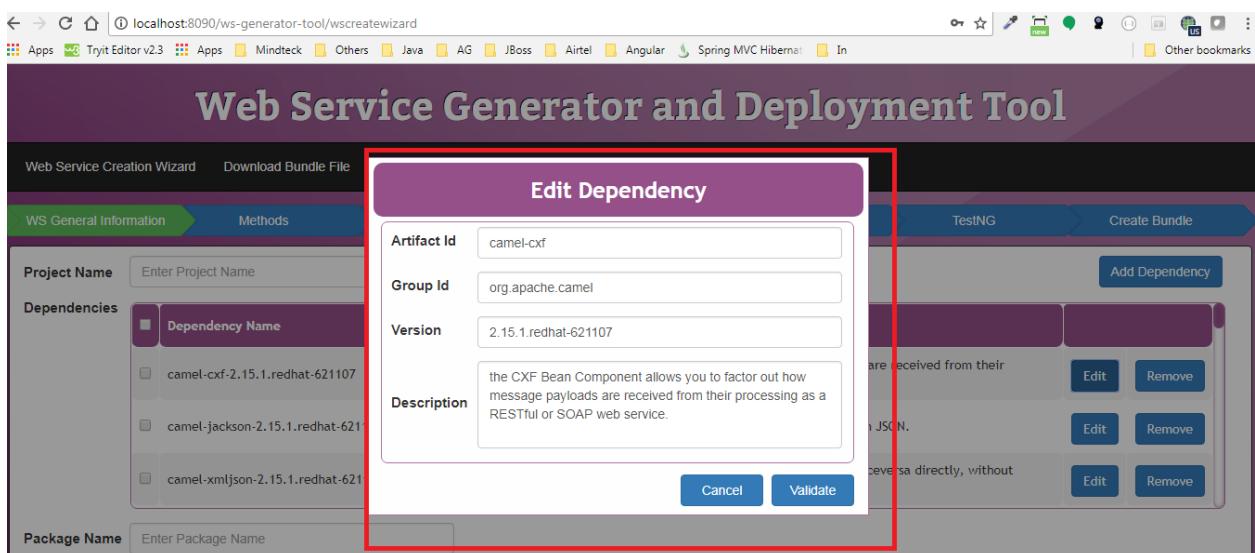
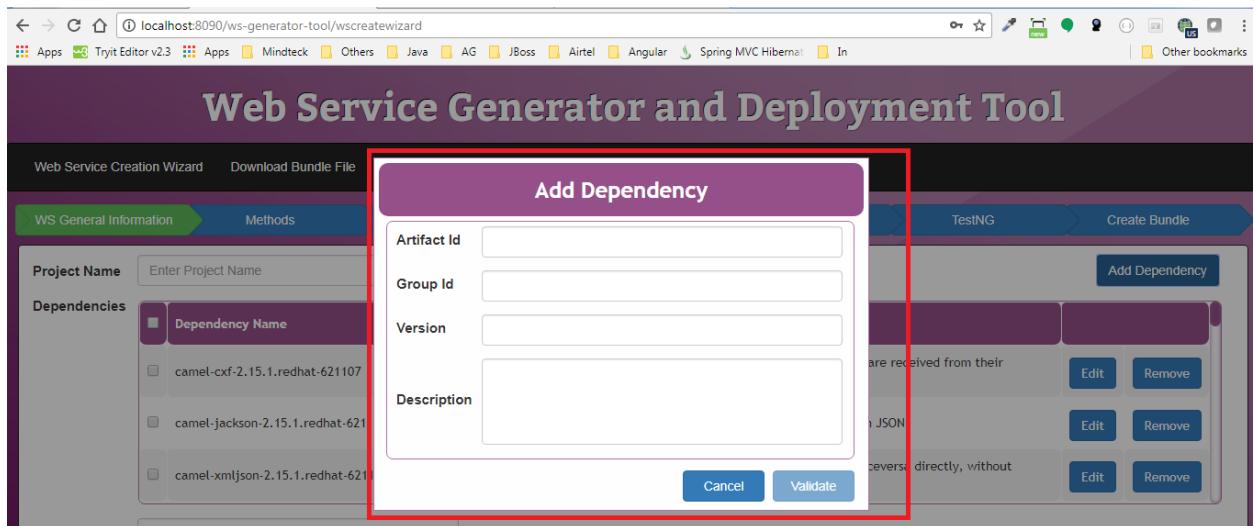
### 3.10 Dependency Management

- 3.10.1 There shall be a screen with name Dependency Management.
- 3.10.2 Dependency Management screen shall have the option to add, edit, delete dependencies.
- 3.10.3 Add button shall allow the user to add new row where a new dependency along with the details like name, description and version number may be entered.
- 3.10.4 The other details like group id, artifact id shall be internally tackled.
- 3.10.5 User shall need to enter all the values for input fields in the new row to Add a dependency.
- 3.10.6 For updating the existing dependency user shall need to select it and modify and shall click on Update button to update the dependency.
- 3.10.7 User may be allowed to delete one or more dependencies by selecting multiple check boxes in the Delete tab.



The screenshot shows the 'Web Service Generator and Deployment Tool' interface. The top navigation bar includes links for 'Web Service Creation Wizard', 'Download Bundle File', 'Deploy Bundle File', 'More...', and 'Logout'. Below the navigation is a breadcrumb menu with tabs: 'WS General Information', 'Methods', 'Properties', 'Compile', 'SonarQube', 'TestNG', and 'Create Bundle'. The main content area is titled 'Dependencies'. It features a table with columns for 'Dependency Name' and 'Description'. Three rows are listed, each with a checkbox next to it. To the right of the table are two groups of 'Edit' and 'Remove' buttons, with the entire group highlighted by a red box. At the top right of the dependencies section is a blue 'Add Dependency' button, also highlighted by a red box.

Dependency Name	Description
<input type="checkbox"/> camel-cxf-2.15.1.redhat-621107	the CXF Bean Component allows you to factor out how message payloads are received from their processing as a RESTful or SOAP web service.
<input type="checkbox"/> camel-jackson-2.15.1.redhat-621107	JSON is a Data Format to marshal and unmarshal Java objects to and from JSON.
<input type="checkbox"/> camel-xmljson-2.15.1.redhat-621107	Camel format provides the capability to convert from XML to JSON and viceversa directly, without stepping through intermediate POJOs.



### 3.4 Basic User Guide

3.4.1 There shall be a separate document containing screen shots with description describing how the user shall use the Tool.

### 3.5 Design Constraints

3.5.1 Testing will be done with only test/local database  
 3.5.2 Software will not be provided by Mindteck

### 3.6 Logical Database Requirements

3.6.1 This application will support below databases  
 3.6.1.1 Postgres  
 3.6.1.2 MySQL

3.6.1.3 Oracle

3.6.1.4 MSSQL

#### **4 Change Management Process**

The changes formally requested in this project would go through Mindteck's standard change management process which would examine and prioritize each change request and estimate the effort and schedule impact for each change. Minor changes may be absorbed within the project schedule without any additional cost implications. For major change requests, the cost / schedule impact must be mutually discussed and agreed upon before taking up the implementation for the change request. The difference between minor and major changes will be based on the work effort required to complete the change and evaluated on a case by case basis. This change request process can be initiated by Censof at any time during the project. Change requests received early would usually have lesser impact on project cost/schedule than the same change request received in the later part of the project execution.

## A. Appendices

### A.1 Appendix 1

The format of the Predefined Template may be as follows:

```
<xml>
<message>
    <header>
        <applicationCode>
            <isMandatory>true</isMandatory>
            <isNumeric>false</isNumeric>
            <isAlphaNumeric>true</isAlphaNumeric>
        </applicationCode>
        <currentPageNum>
            <isMandatory>true</isMandatory>
            <isNumeric>true</isNumeric>
            <isAlphaNumeric>false</isAlphaNumeric>
        </currentPageNum>
        <agencyCode>
            <isMandatory>true</isMandatory>
            <isNumeric>false</isNumeric>
            <isAlphaNumeric>false</isAlphaNumeric>
        </agencyCode>
        <sessionID>
            <isMandatory>false</isMandatory>
            <isNumeric></isNumeric>
            <isAlphaNumeric>false</isAlphaNumeric>
        </sessionID>
        <serviceName>
            <isMandatory>true</isMandatory>
            <isNumeric>false</isNumeric>
            <isAlphaNumeric>false</isAlphaNumeric>
        </serviceName>
        <serviceVersion>
            <isMandatory>true</isMandatory>
            <isNumeric>false</isNumeric>
            <isAlphaNumeric>false</isAlphaNumeric>
        </serviceVersion>
        <resMessageID>
            <isMandatory>false</isMandatory>
            <isNumeric>false</isNumeric>
            <isAlphaNumeric>false</isAlphaNumeric>
        </resMessageID>
        <agencyAppAuthCode>
```

```
<isMandatory>false</isMandatory>
<isNumeric>false</isNumeric>
<isAlphaNumeric>false</isAlphaNumeric>
</agencyAppAuthCode>
<reqMessageID>
    <isMandatory>false</isMandatory>
    <isNumeric>false</isNumeric>
    <isAlphaNumeric>false</isAlphaNumeric>
</reqMessageID>
<agencyAppAuthKey>
    <isMandatory>false</isMandatory>
    <isNumeric>false</isNumeric>
    <isAlphaNumeric>false</isAlphaNumeric>
</agencyAppAuthKey>
<compressedPayload>
    <isMandatory>true</isMandatory>
    <isNumeric>false</isNumeric>
    <isAlphaNumeric>false</isAlphaNumeric>
</compressedPayload>
<totalPageNum>
    <isMandatory>false</isMandatory>
    <isNumeric>false</isNumeric>
    <isAlphaNumeric>false</isAlphaNumeric>
</totalPageNum>
<transactionChecksum>
    <isMandatory>false</isMandatory>
    <isNumeric>false</isNumeric>
    <isAlphaNumeric>false</isAlphaNumeric>
</transactionChecksum>
</header>
<messagePayloadName>hpmk_message</messagePayloadName>
<headerPayloadName>hpmk_message_header</headerPayloadName>
<errorName>hpmk_error</errorName>
<requestDataPayloadName>hpmk_message_payload</requestDataPayloadName>
<responsePayloadName>hpmk_data</responsePayloadName>

<responseDataPayloadName>hpmk_response_data</responseDataPayloadName>
</message>
<validations>
    <validation>
        <errorId>0</errorId>
        <errorCode>0</errorCode>
        <errorDesc>Success</errorDesc>
    </validation>

```

```
<validation>
    <errorId>1</errorId>
    <errorCode>1</errorCode>
    <errorDesc>Failure</errorDesc>
</validation>
<validation>
    <errorId>2</errorId>
    <errorCode>2</errorCode>
    <errorDesc>Invalid_UserUID</errorDesc>
</validation>
<validation>
    <errorId>3</errorId>
    <errorCode>3</errorCode>
    <errorDesc>Required_Parameter_Is_Missing</errorDesc>
</validation>
<validation>
    <errorId>4</errorId>
    <errorCode>4</errorCode>

<errorDesc>Required_Parameter_Contains_Data_Format_Error</errorDesc>
</validation>
<validation>
    <errorId>5</errorId>
    <errorCode>5</errorCode>
    <errorDesc>Required_Parameter_Contains_No_Data</errorDesc>
</validation>
<validation>
    <errorId>6</errorId>
    <errorCode>6</errorCode>
    <errorDesc>Internal_Software_Exception</errorDesc>
</validation>
<validation>
    <errorId>7</errorId>
    <errorCode>7</errorCode>
    <errorDesc>Database_Exception</errorDesc>
</validation>
<validation>
    <errorId>8</errorId>
    <errorCode>8</errorCode>
    <errorDesc>Invalid_Message_Format</errorDesc>
</validation>
<validation>
    <errorId>9</errorId>
    <errorCode>9</errorCode>
```

```
<errorDesc>Unsupported_Command</errorDesc>
</validation>
<validation>
    <errorId>10</errorId>
    <errorCode>10</errorCode>
    <errorDesc>Optimistic_Locking_Error</errorDesc>
</validation>
<validation>
    <errorId>11</errorId>
    <errorCode>11</errorCode>
    <errorDesc>Too_Many_Payloads</errorDesc>
</validation>
<validation>
    <errorId>12</errorId>
    <errorCode>12</errorCode>
    <errorDesc>Timestamp_Malformed</errorDesc>
</validation>
<validation>
    <errorId>13</errorId>
    <errorCode>13</errorCode>
    <errorDesc>NotEnough_Header_Rows</errorDesc>
</validation>
<validation>
    <errorId>14</errorId>
    <errorCode>14</errorCode>
    <errorDesc>WS_Timeout</errorDesc>
</validation>
<validation>
    <errorId>15</errorId>
    <errorCode>15</errorCode>
    <errorDesc>WS_Cannot_Connect</errorDesc>
</validation>
<validation>
    <errorId>16</errorId>
    <errorCode>16</errorCode>
    <errorDesc>All_Targets_Unavailable</errorDesc>
</validation>
<validation>
    <errorId>17</errorId>
    <errorCode>17</errorCode>
    <errorDesc>WS_Route_Not_Found</errorDesc>
</validation>
<validation>
    <errorId>18</errorId>
```

```

<errorCode>18</errorCode>
<errorDesc>WS_Exceeds_Connection_Limit</errorDesc>
</validation>
<validation>
    <errorId>19</errorId>
    <errorCode>19</errorCode>
    <errorDesc>WS_Rejected_Due_To_Filter_Rules</errorDesc>
</validation>
<validation>
    <errorId>26</errorId>
    <errorCode>26</errorCode>
    <errorDesc>Authentication Fail</errorDesc>
</validation>
<validation>
    <errorId>27</errorId>
    <errorCode>27</errorCode>
    <errorDesc>Version or Service is not available</errorDesc>
</validation>
</validations>
</xml>

```

## A.2 Appendix 2

### Traceability Metrics –

**Requirement Traceability Matrix**

SRS/FRS/HRS	Dependent Req	HLD/HDD Ref	LLD Ref	Code / Class / Module / Use case name	Int Testing Ref	System Testing Ref

### Test Case -

Application Functional Testing						
Test Case No.	Req ID.	Test Case Name	Pre-Condition	Status	Steps	Expected output