Oracle® Retail

Functional Artifact Generator Guide

Release 16.0

E80316-01

December 2016

ORACLE®

Oracle Retail Functional Artifact Generator Guide, Release 16.0

E80316-01

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Primary Author: Gayathri Joshi

Contributing Author: Maria Andrew

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(ii) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(iii) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(iv) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR

Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Send Us Your Comments vii

Preface ix

Audience ix
Documentation Accessibility ix
Related Documents ix
Customer Support x
Review Patch Documentation x
Improved Process for Oracle Retail Documentation Corrections x
Oracle Retail Documentation on the Oracle Technology Network xi
Conventions xi

1 Introduction

Concepts 1-1
Functional Artifact Types 1-2
RTG XML Schemas (XSD) 1-2
RTG JAXB Java Beans 1-2
RTG Objects (Oracle Objects) 1-3
Sample XML File 1-3
Technical Specifications 1-3
Supported Operating Systems 1-3
Command Line 1-3
Graphical User Interface (GUI) 1-3

2 Installation and Basic Setup

Determining the Type of Installation 2-1
Installing As a Standalone Application 2-1
Installing As an Application Inside rib-home 2-2
Installing As a Web Application in Oracle WebLogic 2-3
Prerequisites 2-3
Deploying the Artifact Generator Application 2-3
Creating the agAdminGroup 2-4
Verifying the Artifact Generator Web Application 2-4
Redeploying the Application 2-5

3 General Usage

Customizing Payloads - Prerequisites 3-1
Rules for Customization/Extension 3-1
Directory Structure 3-2
Customization 3-2
Localization 3-2
Customizing and Localizing the Template Creator 3-3
Template Creator Overview 3-3
Arguments 3-3
Commands 3-4
Template Creator - Command Line Interface 3-5
Using the Command Line Interface 3-5
Customizing Or Localizing the Elements 3-7
Adding Customized Optional Elements 3-7
Adding Localized Optional Elements 3-8
Adding Elements To A New ExtOfxxx.xsd 3-9
Creating a Work Area - Graphical User Interface 3-10
Template Creator - Graphical User Interface 3-12
Using the Graphical User Interface 3-12
Graphical User Interface - Examples 3-15
Adding Optional Elements 3-17

A Appendix: Install the Artifact Generator

Creating the Role/User in the Artifact Generator A-1
Installing as a Web Application in Oracle WebLogic A-1
Deploying the Artifact Generator Application A-1
Verifying the Artifact Generator Web Application A-7
Redeploying the Application A-9

B Appendix: Example of Customization and Localization for ASNInDesc

C Appendix: References

Send Us Your Comments

Oracle Retail Functional Artifact Generator Guide, Release 16.0

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Applications Release Online Documentation CD available on My Oracle Support and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

The *Oracle Retail Functional Artifact Generator Guide* provides information about the tool as well as installation instructions.

Audience

The *Oracle Retail Functional Artifact Generator Guide* is intended for the Oracle Retail Integration application integrators and implementation staff, as well as the retailer's Information Technology personnel.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Retail Integration Bus documentation set:

- *Oracle Retail Integration Bus Implementation Guide*
- *Oracle Retail Integration Bus Installation Guide*
- *Oracle Retail Integration Bus Operations Guide*
- *Oracle Retail Integration Bus Release Notes*
- *Oracle Retail Integration Bus Hospital Administration Guide*
- *Oracle Retail Integration Bus Security Guide*
- *Oracle Retail Integration Bus Support Tools Guide*
- *Oracle Retail Enterprise Integration Guide*
- *Oracle Retail Integration Bus Integration Gateway Services Guide*

- *Oracle Retail Functional Artifact Guide*
- *Oracle Retail Service-Oriented Architecture Enabler Tool Guide*
- *Oracle Retail Integration Bus Java Messaging Service (JMS) Console Guide*
- *Oracle Retail Integration Bus Data Model*
- *Oracle Retail Payload Mapper Guide*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 16.0) or a later patch release (for example, 16.0.1). If you are installing the base release, additional patch, and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

Oracle Retail product documentation is available on the following web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

Data Model documents are not available through Oracle Technology Network. You can obtain them through My Oracle Support.)

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1
Introduction

The Artifact Generator is a collection of tools designed to create the various artifacts used within the Oracle Retail messaging infrastructure from an XML Schema (XSD). These XSDs are called Business Objects. They represent the functional definition and technical structure of a Retail Business Entity.

Messages (business objects) that flow between the retail applications are XML messages. Oracle Retail XML message definitions are defined statically through XML schemas (XSDs). The integration infrastructure works with multiple technologies (Java EE, PL/SQL) and so has different ways of representing the same functional XML message structure in different technologies. To make it easier to maintain the various functional artifacts, the Artifact Generator was developed.

The Artifact Generator tool is being made available to give customers the ability to add/modify data which flows from one application to the other.

This guide provides details on the installation and configuration of the tool.

Note: For more complete scenarios and best practices on usage of the tool, see Chapter 11, "Customization and Extension," in the *Oracle Retail Integration Bus Implementation Guide*. Also see the *Oracle Retail Service-Oriented Architecture Enabler Tool Guide*.

Concepts

The functional artifacts are different representations of the same message structure/definition in different technologies (Java EE, PL/SQL). Depending on the retail application's technology, appropriate artifacts are used, converting one from the other as needed.

The core concept is that the single source of truth is the XSD. The XSDs are strict and used by the Artifact Generator to produce the design time physical objects used by the application's API technology (PL/SQL or Java), as well as the runtime validations used by the various integration components.

The most common customization requirements in messaging are the addition of new elements to existing payloads, or the creation of new payloads to support custom business logic added to the base integrated applications.

Each retail message family and type combination maps to one and only one functional message definition. One functional message definition can map to one or more than one family/type combination within the same family. The RTG Integration Guide details these objects and the relationships.

Note: See the *Oracle Retail Integration Bus Integration Guide*.

The Oracle AIA approach and Enterprise Business Object (EBO) model, as well as other industry standards have defined an approach using well known tags and locations to separate the custom extension from the base. This allows the extensions to be preserved as updates to base are applied. The Oracle Retail Business Objects have been designed and constructed to accommodate customer extensions following the Oracle AIA EBO standards and guidelines.

For details and in depth examples, see:

- *Oracle Retail Functional Artifacts Guide*
- Oracle Application Integration Architecture - Enterprise Object Library: Enterprise Business Objects and Messages XML Naming and Design Rules

These standards and conventions define Business Object extension and customization as customer side activities. The RGBU governance process produces a Business Object that is enterprise wide. Changes or additions handled by versioning are packaged as part of a release and defined as Base Objects.

The Artifact Generator is the core tool used for customization and extension of the Business Objects used by the RIB and Web Services generated by the Oracle Retail Service-Oriented Architecture Enabler Tool (RSE).

For details and examples, see:

- *Oracle Retail Service-Oriented Architecture Enabler Tool Guide.*
- Chapter 11, "Customization and Extension," of the *Oracle Retail Integration Bus Implementation Guide*.

Functional Artifact Types

The functional artifacts are different representations of the same message structure/definition in different technologies (Java EE, PL/SQL). Depending on the Oracle Retail application's technology, RTG uses the appropriate artifacts, converting one from the other as needed. The following are the RTG functional object definitions.

RTG XML Schemas (XSD)

The functional XML message structure is a contract between the integrating retail applications and is defined by the XML schemas. All the other artifacts are generated from the XML schemas. XML schemas are the inputs required by the artifact generator.

RTG JAXB Java Beans

JAXB is a standard Java XML binding technology. It provides the mechanism to convert XML instances to Java objects (and vice versa) in a standard way. The Java EE Web service infrastructure internally uses JAXB to marshall and unmarshall the SOAP messages. For every payload XSD, the artifact generator generates the corresponding JAXB beans.

RTG Objects (Oracle Objects)

PL/SQL retail applications communicate with the integration infrastructure using Oracle Objects. These objects are user-defined database objects that define the XML message structure inside the database.

Sample XML File

The tool generates XML files for base artifacts that represent instances of XML message schemas. Each element is present and has appropriate data to the full declared length.

Technical Specifications

The Oracle Retail Artifact Generator has dependencies on Oracle Retail Application installations, as well as on the Oracle Application Servers. This section covers these requirements.

Supported Operating Systems

For the Artifact Generator tool, there are separate requirements for the Command Line and the GUI.

Command Line

Supported On	Version Supported
Operating System	OS certified: ■ Oracle Linux 6 & 7 for x86-64 (Actual hardware or Oracle virtual machine) ■ Red Hat Enterprise Linux 6 & 7 for x86-64 (Actual hardware or Oracle virtual machine) ■ AIX 7.1 (Actual hardware or LPARs) ■ Solaris 11.2 SPARC (Actual hardware or logical domains) ■ HP-UX 11.31 Integrity (Actual hardware, HPVM, or vPars)

Graphical User Interface (GUI)

Supported On	Version Supported
Application Server OS	OS certified with Oracle Fusion Middleware 12c (12.2.1). Options are: ■ Oracle Linux 6 & 7 for x86-64 (Actual hardware or Oracle virtual machine) ■ Red Hat Enterprise Linux 6 & 7 for x86-64 (Actual hardware or Oracle virtual machine) ■ AIX 7.1 (Actual hardware or LPARs) ■ Solaris 11.2 SPARC (Actual hardware or logical domains) ■ HP-UX 11.31 Integrity (Actual hardware, HPVM, or vPars)

Supported On	Version Supported
Application Server	Oracle Fusion Middleware 12c(12.2.1) **Components**: ■ Oracle WebLogic Server 12c(12.2.1) ■ JDK 8+ 64 bit with latest security updates

2

Installation and Basic Setup

This chapter provides instructions for installing and deploying the Oracle Retail Artifact Generator.

Determining the Type of Installation

The Oracle Retail Artifact Generator can be installed and used in any of the following configurations:

- Standalone application
- Application inside rib-home
- Web-application in Oracle WebLogic

Installing As a Standalone Application

To install the Oracle Retail Artifact Generator, complete the following steps.

1. Determine the user and the location to install the Artifact Generator.
2. Verify the JAVA_HOME environment variable is set for the user. The JAVA_HOME must be set to a Java 1.8 JDK with latest security updates. If the user is located on the same server as the Application Server, then setting the JAVA_HOME to $ORACLE_HOME/jdk is recommended.

   ```
   > echo $JAVA_HOME
   >/home/aia1/oracle/middleware/jdk
   ```

3. Create a directory for the Artifact Generator.

   ```
   > mkdir ArtifactGeneratorStandalone
   ```

4. Download and extract the Artifact Generator to the Artifact Generator home directory.

   ```
   > cd ArtifactGeneratorStandalone
   > cp /u00/stage/RIB16.0.0/ArtifactGenerator16.0.0ForAll16.x.xApps_eng_ga.tar.
   > tar -xvf  ArtifactGenerator16.0.0ForAll16.x.xApps_eng_ga.tar
   ```

   This step creates the Artifact Generator root directory structure. For example: /user/aia1/ArtifactGenerator/retail-func-artifact-gen.

   This structure becomes AG_HOME.

Note: AG_HOME is assumed to be the artifact-generator-home directory in the following steps.

```
> export AG_HOME=/user/home/aia1/ArtifactGenerator/retail-func-artifact-gen
```

Note: For a base artifact, use the RibFunctionalArtifacts for RIB. For Retail Services Backbone, use the RetailFunctionalArtifacts instead.

5. Download the RIB Functional Artifact tar or the Retail Functional Artifact tar to the AG_HOME/base-func-artifacts directory.

```
> cd $AG_HOME/base-func-artifacts
> cp /u00/stage/RIB16.0.0/RibFuncArtifact16.0.0ForAll16.0.0Apps_eng_ga.tar.
```

Do not untar it. This will be handled by the setup scripts.

6. Execute the setup script.

```
> cd $AG_HOME
>$JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain
com/oracle/retail/integration/artifact/generator/SetupWorkArea.groovy
```

7. Installation is complete. See Chapter 3, "General Usage."

Installing As an Application Inside rib-home

For the following steps, $RIB_HOME is assumed to be the rib-home directory in the rib-app-builder directory tree structure.

For example: /u00/rib/Rib1600ForAll16xxApps/rib-home

1. Verify the JAVA_HOME environment variable. The JAVA_HOME must be set to a Java 1.8 JDK with latest security updates.. If the <RIB_HOME> workspace is located on the same server as the Application Server, then setting the JAVA_HOME to $ORACLE_HOME/jdk is recommended.

```
> echo $JAVA_HOME
> /home/aia1/oracle/middleware/jdk
```

2. Download and extract the Artifact Generator to the Artifact Generator into the rib-home/tools-home directory. There already will be a placeholder directory /retail-func-artifact-gen.

```
> cd $RIB_HOME/tools-home
> cp /u00/stage/RIB16.0.0/ArtifactGenerator16.0.0ForAll16.x.xApps_eng_ga.tar
> tar -xvf ArtifactGenerator16.0.0ForAll16.x.xApps_eng_ga.tar
```

Note: For a base artifact, use the RibFunctionalArtifacts for RIB. For Retail Services Backbone, use the RetailFunctionalArtifacts instead.

3. Download the RIB Functional Artifact tar or the Retail Functional Artifact tar to the AG_HOME/base-func-artifacts directory.

```
> cd $AG_HOME/base-func-artifacts
> cp /u00/stage/RIB160.0/RibFuncArtifact16.0.0ForAll16.0.0Apps_eng_ga.tar
```

Do not untar it. This will be handled by the setup scripts.

4. Execute the setup script.

```
> cd $AG_HOME
>$JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain
com/oracle/retail/integration/artifact/generator/SetupWorkArea.groovy
```

5. Installation is complete. See Chapter 3, "General Usage."

Installing As a Web Application in Oracle WebLogic

The steps below describe how to deploy the Oracle Retail Artifact Generator to an Oracle WebLogic Application Server as a Web application.

Note: See the section "Supported Operating Systems," in Chapter 1.

Prerequisites

The following are prerequisites for installation.

- The retail-func-artifact-gen-gui<version>.war file is located within the directory structure of the ArtifactGenerator16.0.0ForAll16.x.xApps_eng_ga.tar. It is recommended that the Artifact Generator be deployed from the rib-home location, although the .war file can be obtained from the stand-alone installation as well.

  Note: See the section, "Installing As an Application Inside rib-home," in this chapter.

- The installation and base configuration of the Oracle WebLogic Server is beyond the scope of this document. Work with the Application Server Administration team to determine the physical and logical placement of the retail-func-artifact-gen-gui component within the WebLogic Server deployment.

  Note: See Oracle WebLogic Server 12c (12.2.1) Installation Guide.

Deploying the Artifact Generator Application

Using the WebLogic Server Administration Console, complete the following steps:

Note: For instructions with illustrations (screen captures), see "Appendix: Install the Artifact Generator."

1. Navigate to the Deployments page.
2. Click **Install**.

Note: If the application has already been installed, see the section, "Redeploying the Application," in this chapter.

The "Locate deployment to install and prepare for deployment" page is displayed. Follow the instructions to locate the retail-func-artifact-gen-gui-<version>.war file.

3. Select **Upload Files**.
4. On the **Upload a Deployment to the Administration server** page, use the Browse button to locate the retail-func-artifact-gen-gui-<version>.war file in the "Deployment Archive."
5. Select the retail-func-artifact-gen-gui-<version>.war
6. Click **Next** and move to **Choose targeting style**.
7. Select **Install this deployment as an application**. Click **Next** and move to **Select Target Deployments**. Select the target server for the Artifact Generator Web application.
8. Click **Next** and move to **Optional Settings**.
9. Click **Next** and move to **Review your choices and click Finish**.
10. Select **No, I will review the configuration later**.
11. Click **Finish** to deploy the application.
12. Click Activate Changes to finish install. Go to Deployments page, select the retail-func-artifact-gen-gui<version> application and click on Start > Servicing all requests button. This should change the status of retail-func-artifact-gen-gui application to Active status.

Creating the agAdminGroup

To create the agAdminGroup, do the following:

1. In WebLogic, click on **Security Realms**.
2. Click on **myrealm** and then **Users and Groups**.
3. Click on groups and then **New**.
4. Enter **agAdminGroup** in the name field, leaving the other fields at default.
5. Click **OK**.
6. Add at least one user to the agAdminGroup group.

Verifying the Artifact Generator Web Application

To verify the artifact generator Web application, do the following.

1. Navigate to the Deployments page.
2. Locate the **retail-func-artifact-gen-gui-<version>**on the Summary of Deployments page.
3. Click the name, **retail-func-artifact-gen-gui-<version>**, to move to the "Settings for the rib-func-artifact-gen-gui."
4. Select the Testing tab.
5. Click the link for the index.jsp URL in the Test Point. The URL should open to the Retail Artifact Generator Login page.
6. The installation is complete. For more information, see Chapter 3, "General Usage."
7. Click on the **Setup Work Areas** tab.
8. Click **Choose File**, navigate to, and select the `RibFuncArtifact16.0.0ForAll16.0.0Apps_eng_ga.tar` file.

9. Click **Create Work Area**.

Redeploying the Application

If the retail-func-artifact-gen-gui application has already been deployed, follow these steps.

1. If the retail-func-artifact-gen-gui-<version> application is running, select **Stop** and **When Work Completes** or **Force Stop Now,** depending on the environment. The recommended option always is **When Work Completes.**
2. Click **Delete.**
3. The retail-func-artifact-gen-gui-<version> should now not show on the Summary of Deployment page.
4. Return to the appropriate step in Deploy the Application.

3

General Usage

The Artifact Generator tool can be used through the command line or through the GUI, depending on the installation type selected.

The Artifact Generator implements rules for customization/extension of the Business Objects that are used to create the Functional Artifacts for use in the RTG Integration systems. The tools, regardless of the installation type will process only XSDs that have been added or modified per these rules.

The fundamental rule is that for customization of a base XSD, the ExtOf XSDs must be modified and not the original (base) XSD. The tool scans for customization/extension implemented there, and only there. Any changes to the base XSD will be ignored.

For example, if you want to add an optional element to the Currency Rate flow, use the Template Generator to provide the hooks and placeholder XSDs and then add the optional element to ExtOfCurrRateDesc.xsd and not the CurrRateDesc.xsd.

There are simple examples included in the usage sections, but for more complete scenarios and best practices on usage of the tool, see Chapter 11, "Customization and Extension," in the *Oracle Retail Integration Bus Implementation Guide*. See also the *Oracle Retail Service Oriented Architecture Enabler Tool Guide*.

Customizing Payloads - Prerequisites

The following are the prerequisites you need to consider before customizing payloads:

- Familiarity with the Artifact Generator tool and Template Creator tool.
- Understanding the importance of payloads and how they fit into the overall retail family of products.
- Understanding the impact of customizing a payload on other applications.

Rules for Customization/Extension

The following are rules for customization and extension.

- Always make a backup of the particular files being modified during customization.
- Customizations/Extensions of payloads must also be made accordingly to the application side.
- It is strongly recommended that only optional elements are added. The addition of mandatory elements increases complexity.

- The names of the elements in ExtOfs must not be the same as the names in the parent XSD. For example, if CurrRateDesc.xsd has an element name = **attr1**, ExtOfCurrRateDesc cannot have an element named **attr1**.
- Java, PL/SQL, and XML schema keywords cannot be used in the names of elements in XSDs.

Directory Structure

This section provides examples of the directory structure for customizing and localizing payloads.

Customization

The tool creates a directory structure that contains all the libraries (integration-lib) and generally available (GA) input artifacts required to generate all supported output types.

There are output directories for each type of artifact produced. For example:

```
../output-database-object-types
|----------- dist
|---- custom-retail-public-payload-database-object-types-<version>.jar
./output-jaxb-java-beans
|----------- src
|----
com/oracle/retail/integration/custom/bo/extofasnindesc/v1
|----- ExtOfASNInDesc.java…
ObjectFactory.java
package-info.java
|----------- dist
|---- custom-retail-public-payload-java-beans-<version>.jar
./output-xml-samples
|----------- src - All sample file
|----------- dist - custom-retail-public-payload-xml-samples-<version>.jar
```

Localization

The tool creates a directory structure that contains all the libraries (integration-lib) and generally available (GA) input artifacts required to generate all supported output types.

There are output directories for each type of artifact produced. For example:

```
./output-database-object-types
|----------- dist
|----
localization-retail-public-payload-database-object-types-<version>.jar

./output-jaxb-java-beans
|----------- src
|----
com/oracle/retail/integration/localization/bo/brasnindesc/v1
|----- BrASNInDesc.java…
ObjectFactory.java
package-info.java
|----------- dist
|---- localization-retail-public-payload-java-beans-<version>.jar
./output-xml-samples
|----------- src - All sample xmls
```

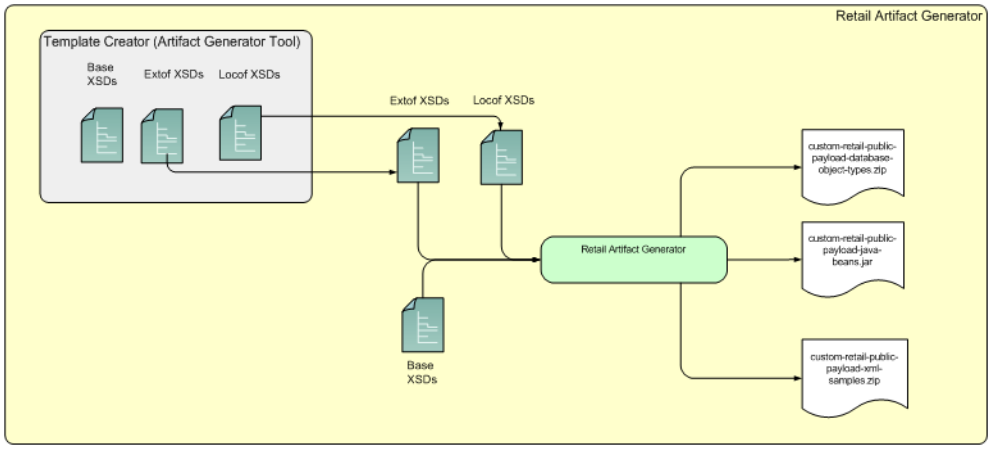
```
|---------- dist - localization-retail-public-payload-xml-samples-<version>.jar
```

Customizing and Localizing the Template Creator

A prerequisite to customization or localization is using the Artifact Generator tool called the Template Creator. This tool constructs the appropriate placeholders in the packaging structure in the correct locations.

The Functional Artifact Generator tool has been enhanced to generate custom and localized payloads business objects (BO) on demand, based on Oracle Retail Functional Artifact rules.

Unlike past releases, the packaging of the retail business object does not contain placeholders for the customization or localization XSDs, and the base XSDs do not contain the imports to include them. The goal is to reduce the number of functional payload objects to only those in base and those needed to satisfy the site requirements.



Template Creator Overview

The Customization and Localization Template Creator tool is part of the Functional Artifact Generator.

The Artifact Generator tools, including the Template Creator, can be used either as a command line or GUI tool set.

Arguments

The following table summarizes arguments for the Template Creator tool:

Option	Argument	Type	Usage
-c	createCustomTemplate createLocalizationTemplate	Required, other than with -h option	To take input for creating custom or localization templates. For example: -c createCustomTemplate -c createLocalizationTemplate

Option	Argument	Type	Usage
-n	complex-type name	Required with -c option	To input the name of complex-type for which template has to be generated. For example: -n ItemDesc -n DSDDealsDesc/DSDDeals
-l	locale name	Required with -c createLocalizationTemplate option	To input locale name for localization template. For example: Localization for Brazil -c createLocalizationTemplate -l Br
-x	path of alternate input-xsd folder relative to $AG_HOME	Optional	To use input-xsd folder other than the default input-xsd folder in AG_HOME. For example: -x ./alternate-input-xsd
-h	Help	Optional	To read command line help.

Commands

The following table summarizes valid commands for the Template Creator tool, as well as corresponding output.

Command	Output
$JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain com/oracle/retail/integration/artifact/generator/TemplateCreator.groovy -c createCustomTemplate -n ItemDesc	■ Extension hooks added to ItemDesc.xsd ■ ExtOfItemDesc.xsd is created.
$JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain com/oracle/retail/integration/artifact/generator/ TemplateCreator.groovy -c createCustomTemplate -n ASNInDesc/ASNInItem	■ Extension hooks added to ASNInDesc\ASNInItem. ■ ExtOfASNInDesc.xsd is created.
$JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain com/oracle/retail/integration/artifact/generator/ TemplateCreator.groovy -c createLocalizationTemplate -n ItemDesc -l Br	■ Extension hooks added to ItemDesc.xsd ■ LocOfItemDesc.xsd. ■ BrItemDesc.xsd. ■ EOfBrItemDesc.xsd is created.
$JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain com/oracle/retail/integration/artifact/generator/ TemplateCreator.groovy -c createLocalizationTemplate -n ASNInDesc/ASNInItem -l In	■ Extension hooks added to ASNInDesc\ASNInItem. ■ LocOfASNInDesc.xsd. ■ InASNInDesc.xsd. ■ EOfInASNInDesc.xsd is created.
$JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain com/oracle/retail/integration/artifact/generator/ TemplateCreator.groovy -c createCustomTemplate -n CustOrderDesc	ExtOfCustOrderDesc.xsd is created.

Command	Output
$JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain com/oracle/retail/integration/artifact/generator/ TemplateCreator.groovy -c createLocalizationTemplate -n CustOrderDesc -l Br	■ LocOfCustOrderDesc.xsd. ■ BrCustOrderDesc.xsd. ■ EOfBrCustOrderDesc.xsd is created.

Template Creator - Command Line Interface

The following sections describe how you can work with the Template Creator tool using its Command Line Interface.

Using the Command Line Interface

The Template Creator groovy script can be used to create custom/localization template for a base or new (having no existing base) retail business object.

Create a custom template for base retail business object

Command:

$JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain

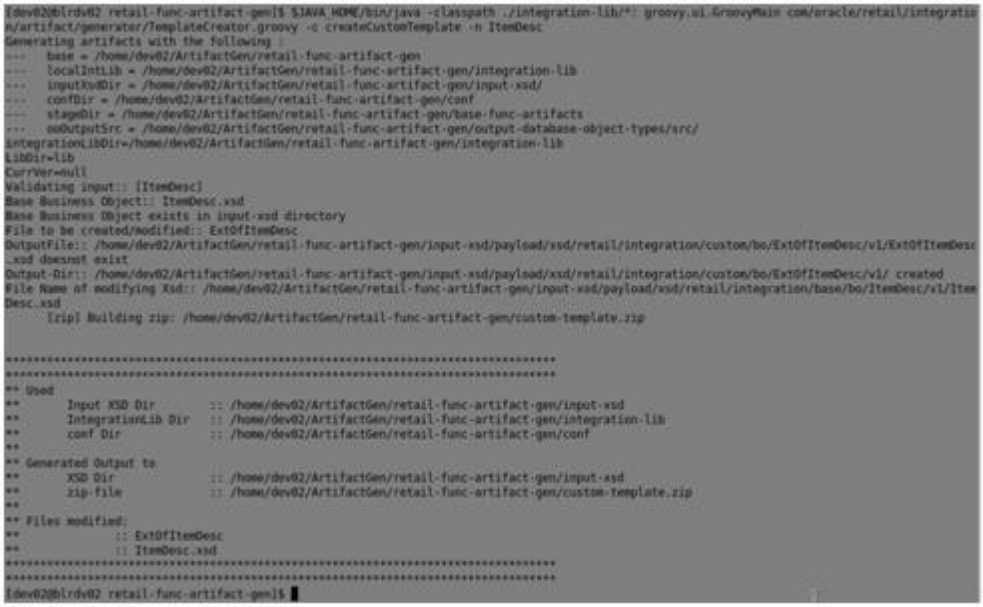
com/oracle/retail/integration/artifact/generator/TemplateCreator.groovy -c createCustomTemplate -n ItemDesc

Output: The following files are created.

ExtOfItemDesc.xsd in $input-xsd/payload/xsd/retail/integration/custom/bo/ExtOfItemDesc/v1/ folder.

Base XSD is modified:

ItemDesc.xsd in base/bo/ItemDesc/v1/ folder.



Create a custom template for new business object (no existing base retail business object)

Command:

$AG_HOME> $JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain com/oracle/retail/integration/artifact/generator/TemplateCreator.groovy -c createCustomTemplate -n CustomerOrdDesc

Output: The following files are created.

ExtOfCustOrderDesc.xsd in $input-xsd/payload/xsd/retail/integration/custom/bo/

ExtOfCustOrderDesc/v1/ folder.

Create a localization template for base retail business object

Command:

$AG_HOME> $JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain com/oracle/retail/integration/artifact/generator/TemplateCreator.groovy

-c createLocalizationTemplate -n ItemDesc -l Br

Output: The following files are created.

LocOfItemDesc.xsd in $input-xsd/payload/xsd/retail/integration/base/bo/LocOfItemDesc/v1/

BrItemDesc.xsd in $input-xsd/payload/xsd/retail/integration/localization/bo/BrItemDesc/v1/

EOfBrItemDesc.xsd in $input-xsd/payload/xsd/retail/integration/custom/bo/EOfBrItemDesc/v1/

Base XSD is modified:

ItemDesc.xsd in $input-xsd/payload/xsd/retail/integration/base/bo/ItemDesc/v1/

Create a localization template for new business object (no existing base retail business object)

Command:

$AG_HOME> $JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain com/oracle/retail/integration/artifact/generator/TemplateCreator.groovy -c createLocalizationTemplate -n CustOrderDesc -l Br

Output: The following files are created.

LocOfCustOrderDesc in

$input-xsd/payload/xsd/retail/integration/base/bo/LocOfCustOrderDesc/v1/

BrCustOrderDesc.xsd in

$input-xsd/payload/xsd/retail/integration/localization/bo/BrCustOrderDesc/v1/

EOfCustOrderDesc.xsd in

$input-xsd/payload/xsd/retail/integration/custom/bo/EOfBrCustOrderDesc/v1/

Customizing Or Localizing the Elements

After the Template Creator has been run to create the appropriate placeholders, the actual custom or localization elements can be added.

It is important to understand that the customization/extension of existing Business Objects should be performed in the ExtOfxxx XSDS of a base XSD, not the base XSD. The tool is designed to enforce this best practice. It supports the preservation of customization/extension when there are new versions of the base objects released.

The basics of XSDs are not covered in this document.

Note: See "Appendix: References."

Adding Customized Optional Elements

This section explains how to add an optional element (simple type or complex type) to existing message payloads.

1. Edit the desired payload XSDs in ./input-xsd/payload/xsd/retail/integration/custom/bo directory of Rib Artifact Generator tool installation. Add the optional simple or complex element to the particular message family xsd. If needed, define the type it belongs to if it does not exist.

   ```
   > cd input-xsd/payload/xsd/retail/integration/custom/bo/ExtOfCurrRateDesc/v1
   > vi ExtOfCurrRateDesc.xsd (make changes)
   ```

   The following example shows the modifications to ExtOfCurrRateDesc.xsd required to add the optional element, **country**.

   ```
   <xs:schema elementFormDefault="qualified"
   targetNamespace="http://www.oracle.com/retail/integration/custom/bo/ExtOfCurrRa
   teDesc/v1"
       version="1.0"

   xmlns="http://www.oracle.com/retail/integration/custom/bo/ExtOfCurrRateDesc/v1"
       xmlns:retailDoc="http://www.w3.org/2001/XMLSchema"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
       <xs:element name="ExtOfCurrRateDesc">
   ```

```
        <xs:complexType>
            <xs:sequence>
                        <xs:element minOccurs="0" name="country"
type="varchar23">
                        </xs:element>
             </xs:sequence>
        </xs:complexType>
    </xs:element>
        <xs:simpleType name="varchar23">
                <xs:restriction base="xs:string">
                        <xs:maxLength value="3"/>
                </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

2. Run the Artifact Generator to generate various functional artifacts.

```
> $JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain
com/oracle/retail/integration/artifact/generator/GenArtifacts.groovy -g
generateCustom
```

3. All necessary artifacts are generated as follows:
   - custom-retail-public-payload-java-beans-<version>.jar is generated in retail-func-artifact-gen/output-jaxb-java-beans/dist folder.
   - The java source files are generated in retail-func-artifact-gen/output-jaxb-java-beans/src folder
   - custom-retail-public-payload-database-object-types-<version>.jar is generated in retail-func-artifact-gen/output-database-object-types/dist folder.

Adding Localized Optional Elements

This section explains how to add an optional element (simple type or complex type) to existing message payloads.

1. Edit the desired payload XSDs in ./input-xsd/payload/xsd/retail/integration/localization/bo directory of Rib Artifact Generator tool installation. Add the optional simple or complex element to the particular message family xsd. If needed, define the type it belongs to if it doesn't exist.

```
> cd input-xsd/payload/xsd/retail/integration/localization/bo/BrCurrRateDesc/v1
> vi BrCurrRateDesc.xsd (make changes)
```

   The following example shows the modifications to BrCurrRateDesc.xsd required to add the optional element, country.

```
 <xs:schema elementFormDefault="qualified"

targetNamespace="http://www.oracle.com/retail/integration/localization/bo/BrCur
rRateDesc/v1"
    version="1.0"

xmlns="http://www.oracle.com/retail/integration/localization/bo/BrCurrRateDesc/
v1"

xmlns:EOfBrCurrRateDesc="http://www.oracle.com/retail/integration/custom/bo/EOf
BrCurrRateDesc/v1"
    xmlns:retailDoc="http://www.w3.org/2001/XMLSchema"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
    <xs:import

namespace="http://www.oracle.com/retail/integration/custom/bo/EOfBrCurrRateDesc
/v1"
schemaLocation="../../../../custom/bo/EOfBrCurrRateDesc/v1/EOfBrCurrRateDesc.xs
d">
        <retailDoc:annotation>
            <retailDoc:documentation>It's a referenced element. For detailed
description, please refer referenced element doc.</retailDoc:documentation>
        </retailDoc:annotation>
    </xs:import>
<xs:element name="BrCurrRateDesc">
<xs:complexType>
            <xs:sequence>
<xs:element minOccurs="0" name="country" type="varchar23">

            <xs:element maxOccurs="1" minOccurs="0"
ref="EOfBrCurrRateDesc:EOfBrCurrRateDesc">
               <retailDoc:annotation>
                 <retailDoc:documentation>Provide an extension hook to
customize CurrRateDesc
                 </retailDoc:documentation>
               </retailDoc:annotation>
             </xs:element>
            </xs:sequence>
        </xs:complexType>
</xs:element>
        <xs:simpleType name="varchar23">
                <xs:restriction base="xs:string">
                        <xs:maxLength value="3"/>
                </xs:restriction>
        </xs:simpleType>

</xs:schema>
```

2. Run the Artifact Generator to generate various functional artifacts.

```
> $JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain
com/oracle/retail/integration/artifact/generator/GenArtifacts.groovy  -g
generateLocalization
```

3. All the necessary artifacts are generated as follows:
   - localization-retail-public-payload-java-beans-<version>.jar is generated in retail-func-artifact-gen/output-jaxb-java-beans/dist folder.
   - The java source files are generated in retail-func-artifact-gen/output-jaxb-java-beans/src folder.
   - localization-retail-public-payload-database-object-types-<version>.jar is generated in retail-func-artifact-gen/output-database-object-types/dist folder.

Adding Elements To A New ExtOfxxx.xsd

The following steps must be completed to add a new custom XSD to a current set of payloads:

1. Add a new element to the ExtOfxxx.xsd.
2. Run the Artifact Generator to generate various functional artifacts.

```
> $JAVA_HOME/bin/java -classpath ./integration-lib/*: groovy.ui.GroovyMain
```

```
com/oracle/retail/integration/artifact/generator/GenArtifacts.groovy
```

Upon completion of this step, all necessary artifacts are generated as follows:

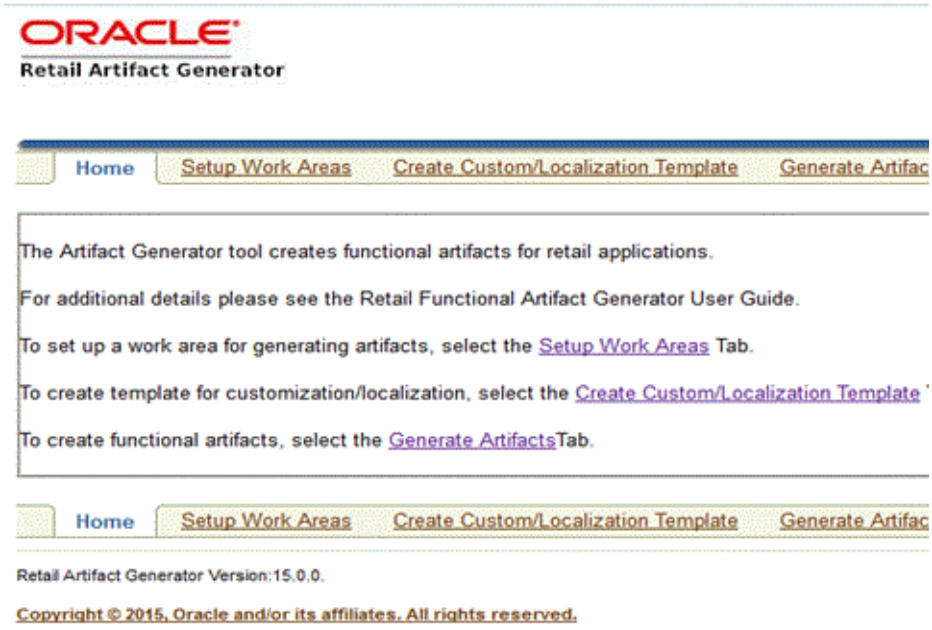
- custom-retail-public-payload-java-beans-<version>.jar is generated in retail-func-artifact-gen/output-jaxb-java-beans/dist folder
- The java source files are generated in retail-func-artifact-gen/output-jaxb-java-beans/src folder
- custom-retail-public-payload-database-object-types-<version>.jar is generated in retail-func-artifact-gen/output-database-object-types/dist folder.

Creating a Work Area - Graphical User Interface

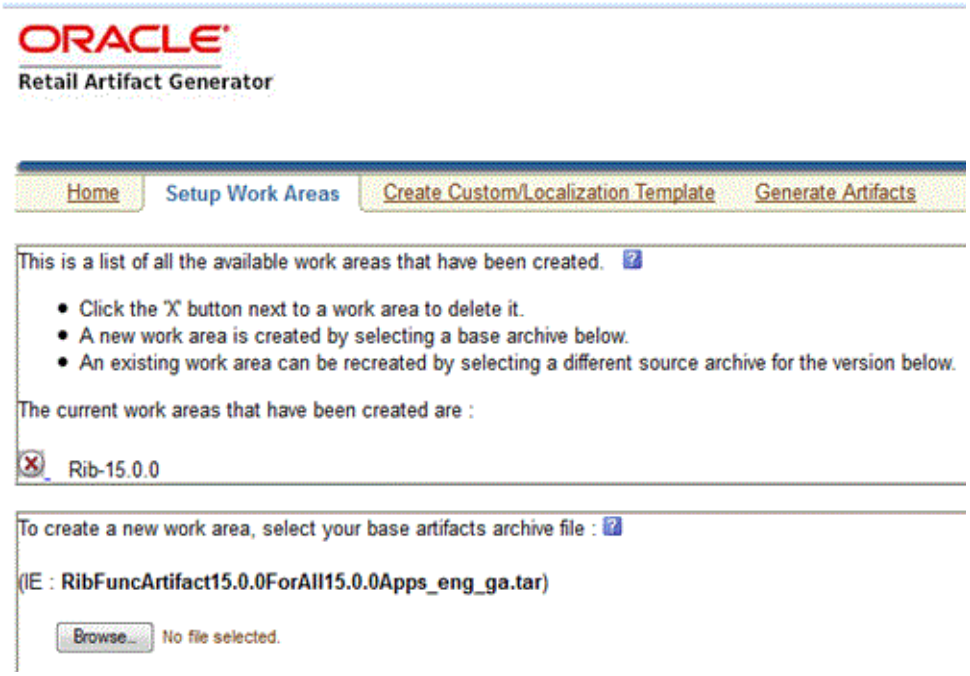
To create a work area, take the following steps:

Note: For a base artifact, use the RibFunctionalArtifacts for RIB. For Retail Services Backbone, use the RetailFunctionalArtifacts instead.

1. Navigate to Retail Artifact Generator Home.

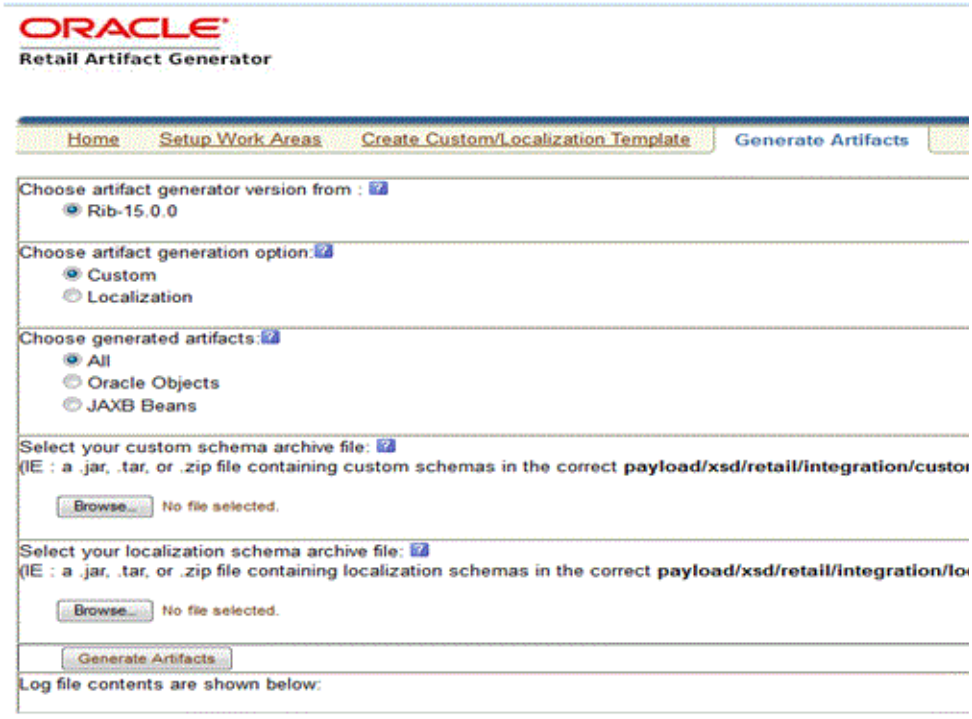


2. To create a new version workspace, select the **SetupWork Areas** tab.



3. On the Setup Work Areas page, use the browse button to locate the RibFuncArtifact16.0.0ForAll16.0.0Apps_eng_ga.tar or the RetailFuncArtifact16.0.0ForAll16.0.0Apps_eng_ga.tar. These .tar files must be in a location that is accessible by the browser.
4. Click **Create Work Area**.
5. To generate artifacts, select the **Generate Artifacts** tab.

Note: At this point, the Template Creator can be used.



6. Choose the artifact generator version.
7. Choose an artifact generation option (Custom/Localization).

   If Custom/Localization is chosen, on the Generate Artifacts page, use the browse button to locate the archive file (for example, .tar, .jar, or .zip) that contains the custom/localized schemas. This file must have the custom/localization schemas in the correct package structure, such as retail/integration/custom/bo* or retail/integration/localization/bo*.

   Note: For information on the packaging structure and guidelines, see the *Oracle Retail Functional Artifacts Guide*.

8. Click **Generate Artifacts** to start the generation process.

   As the process runs, the status of its progress is displayed in the log window. When the process is complete, **save file** dialog windows are displayed with options to save the resulting archive files or open them for review.

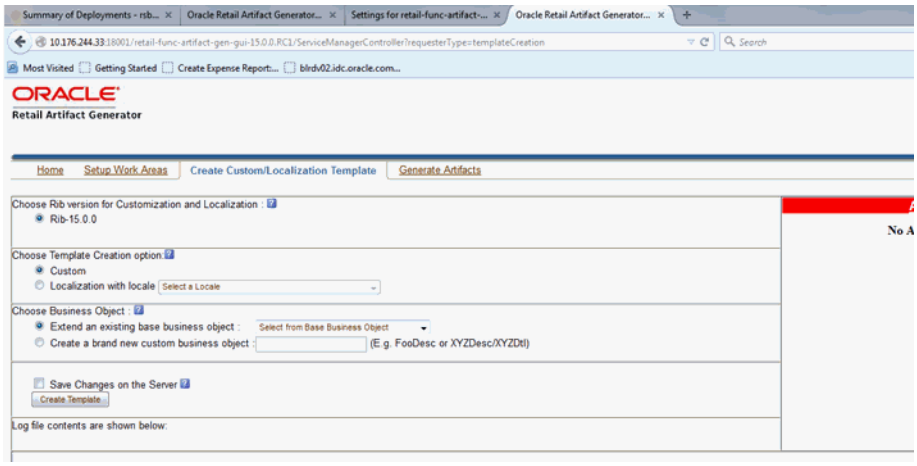
Template Creator - Graphical User Interface

The following sections describe how you can work with the Template Creator tool using its Graphical User Interface (GUI).

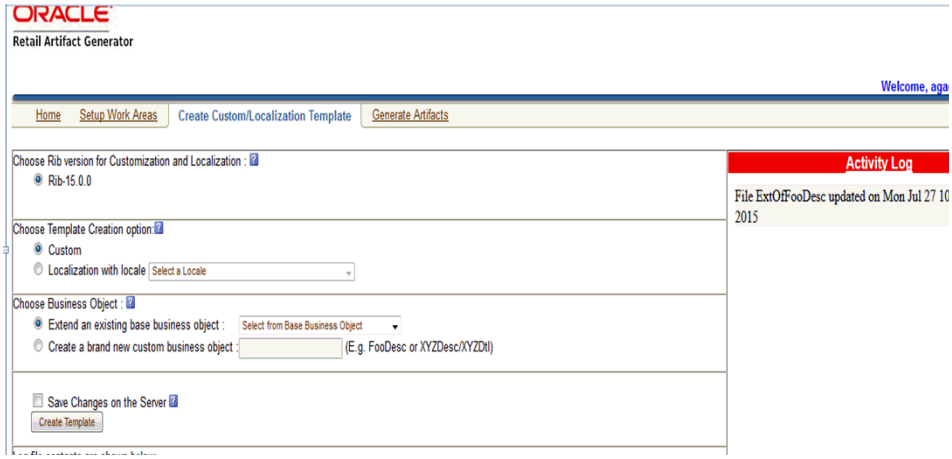
Using the Graphical User Interface

Take the following steps to use the Template Creator through the GUI:

Note: See the Creating a Work Area - Graphical User Interface section in this chapter for how to create work areas.

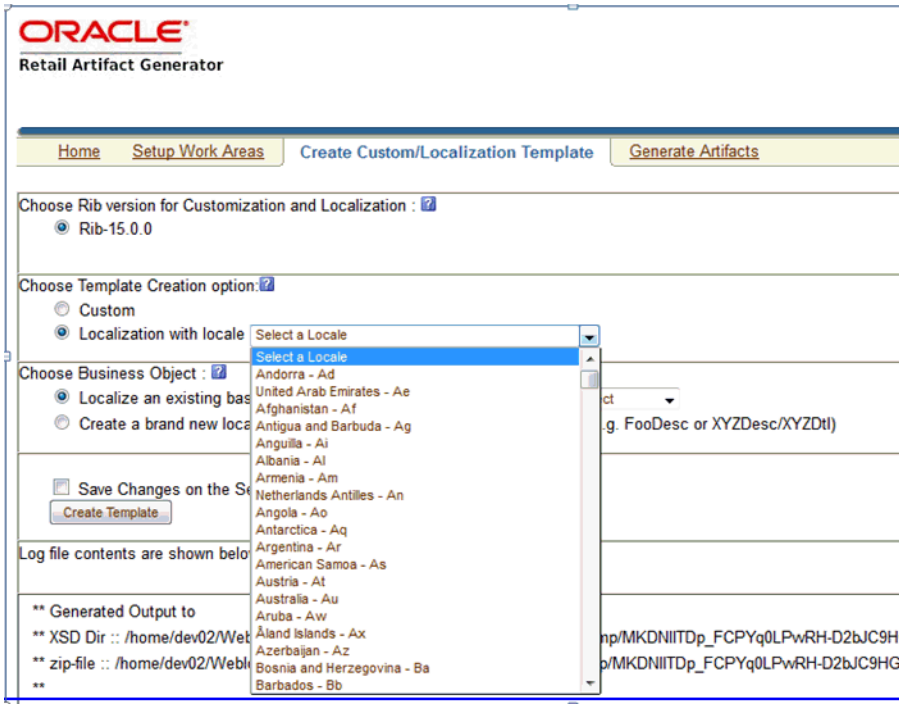


Select a RIB version for customization and localization from the list of RIB versions. The list includes all versions for which the Work Area has been set up. Based on the version selected, the Activity Log panel on the right indicates recent activity, such as a list of changes saved to the server for a specific version.

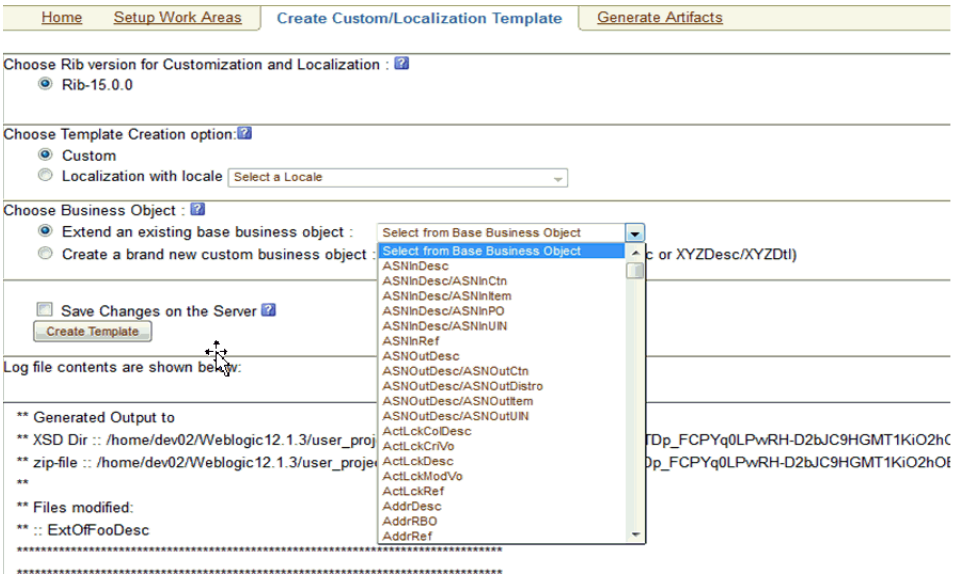


When a version of RIB is selected, the contents of the Activity Log panel includes a list of files, time stamped to indicate when they were updated on the server.

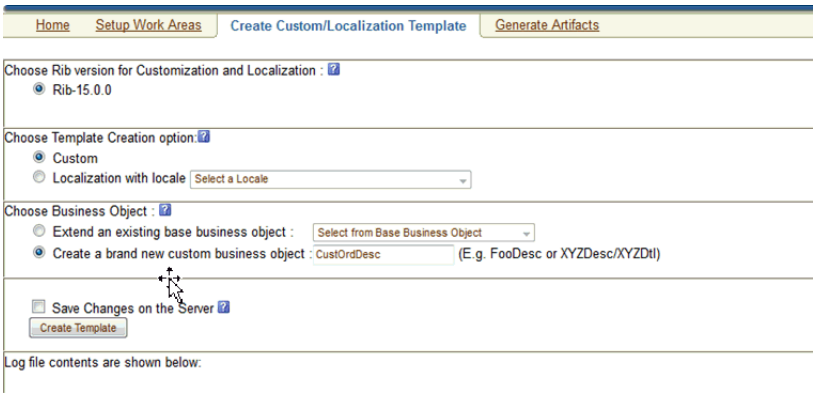
Select the option to create a Custom or Localization template. If the Localization option is selected, you must select a locale from the drop down list. The following screen shows the list of locales available when the Localization option is selected.



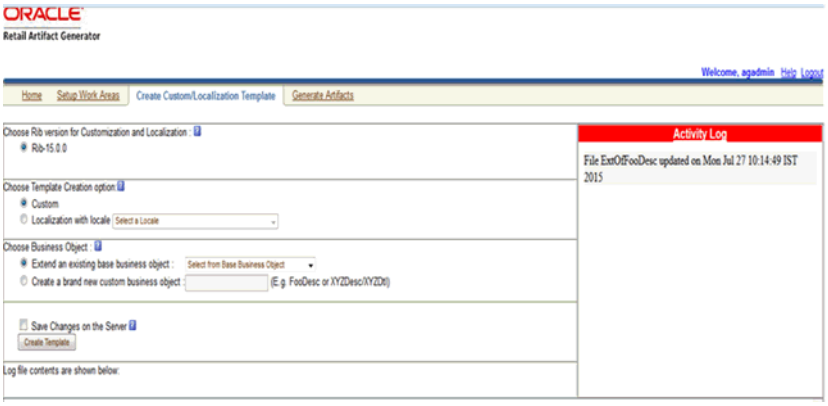
Based on the version selected, a list of all complex types in the form of XPath expression (such as DSDDealsDesc/DSDDeals) is available for template creation. The following screen includes the list of base business objects available for the selected version of RIB. You can also select a complex type from the list to extend/localize an existing base business object, or create a new baseless business object by providing the appropriate details as text input.



To create a new custom/localization business object, select the option to create a brand new business object and provides the complex type details in text box as shown in the following screen.



To save the created templates on the server and make them available for future use, you can select Save Changes to the Server. All successful changes are saved to the server, and the activity is logged in the Recent Activity panel.



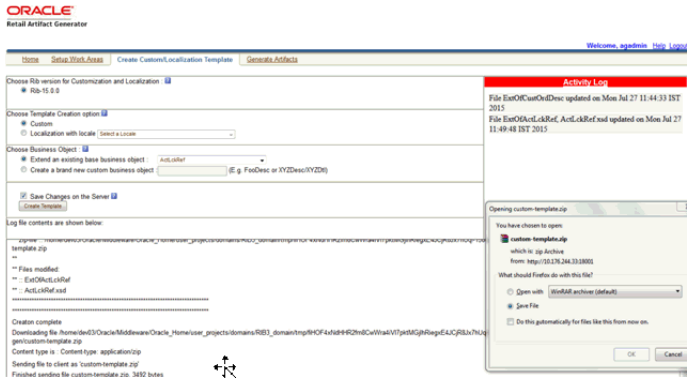
When you click **Create Template**, a request is sent to the server for template creation processing.

Graphical User Interface - Examples

This section provides GUI examples for the Template Creator.

Create a custom template for a base retail business object

In Panel 1, select a version (such as RIB 16.0.0). Select the custom option in Panel 2. Select the option to extend an existing base business. Select a base business object from the list (such as ActLckRef), as shown in the following screen. You have the option of saving the changes to the server.

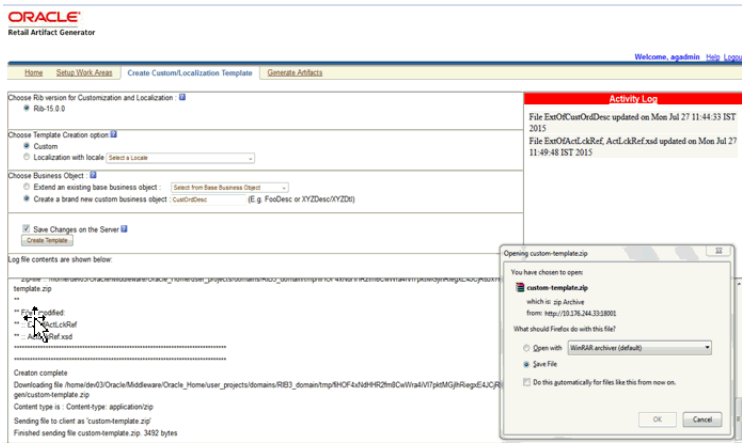


When Create Template is clicked, the custom-template.zip file is created. It can be downloaded, and a local copy can be saved. The zip file includes ActLckRef.xsd and ExtOfActLckRef.xsd.

If Save Changes on the Server is selected, the changes are copied back to the server, and the Activity Log is updated.

Create a custom template for new business object (no existing base retail business object)

In Panel 1, select a version (such as RIB 16.0.0). Select custom option in Panel 2. Select the option to create a brand new custom business object, and input the complex type (such as CustOrdDesc), as shown in the following screen. You have the option of saving the changes to the server.

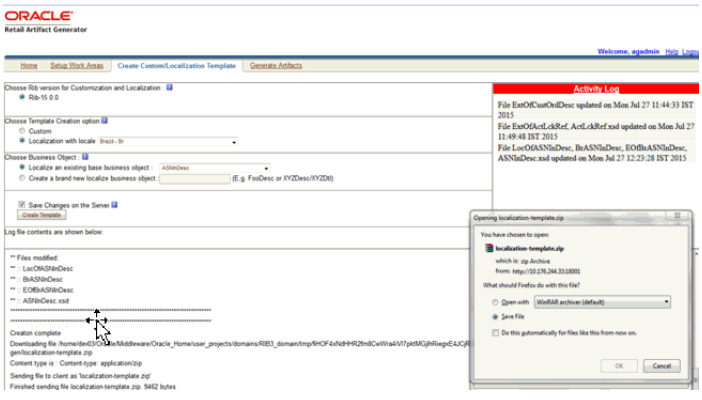


Note: The above Save File window appears only in Mozilla Firefox.

When Create Template is clicked, the custom-template.zip file is created. It is available to download, and local copy can be saved. The zip file includes ExtOfCustOrdDesc.xsd. If you select Save Changes on the Server option, the changes are copied back to the server, and an Activity Log is updated.

Create a localization template for base retail business object

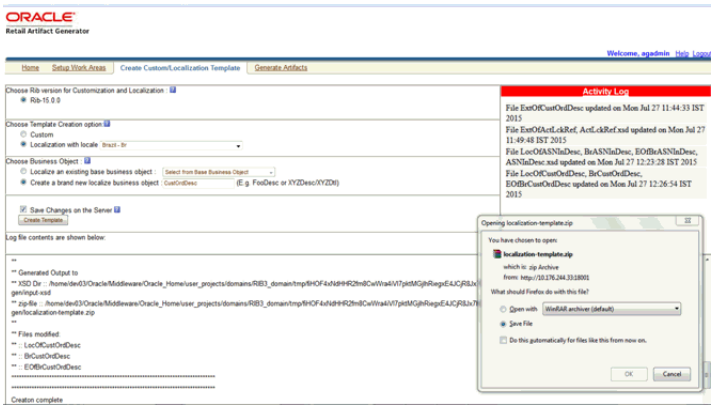
In Panel 1, select the version (such as RIB 16.0.0). In Panel 2, select the localization option from the drop down list (such as Brazil-Br). Select the option to localize an existing base business object. Input the complex type (such as ASNInDesc). You have the option of saving the changes to the server.



When you click **Create Template**, the localization-template.zip is created. It is available for download, and it can be saved on your system. The zip folder includes ASNInDesc.xsd, LocOfASNInDesc.xsd, BrASNInDesc.xsd, and EOfBrASNInDesc.xsd. If you select Save Changes on the Server option, the changes are copied to the server, and the Activity Log is updated.

Create a localization template for new business object (no existing base retail business object)

In Panel 1, select the version (such as RIB 16.0.0). In Panel 2, select a locale from the drop down list (such as Brazil-Br). Select the option to create a brand new localized business object. Input the complex type (such as eCustOrderDesc). You have the option of saving the changes to the server.



When you click **Create Template**, the localization-template.zip is created. It is available for download, and it can be saved on your system. The zip file includes LocOfCustOrderDesc.xsd, BrCustOrderDesc.xsd, and EOfBrCustOrderDesc.xsd. If you select Save Changes on the Server option, the changes are copied back to the server, and Activity Log is updated.

Adding Optional Elements

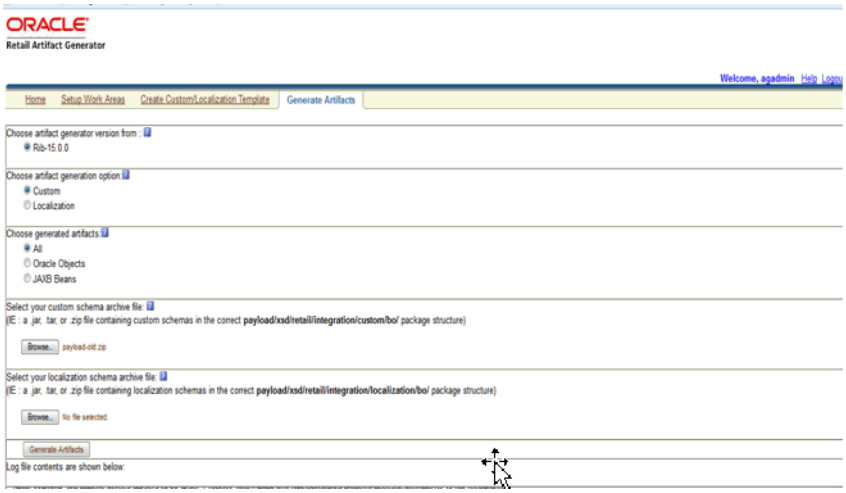
After you create the template, save the generated zip file to the your system.

This section explains how to add an optional element (simple type or complex type) to generated message payloads. Edit the desired payload XSDs of zip file in the following directory “payload/xsd/retail/integration/custom/bo”. Add the optional simple or complex element to the particular message family xsd. If needed, define the type it belongs to if it does not exist.

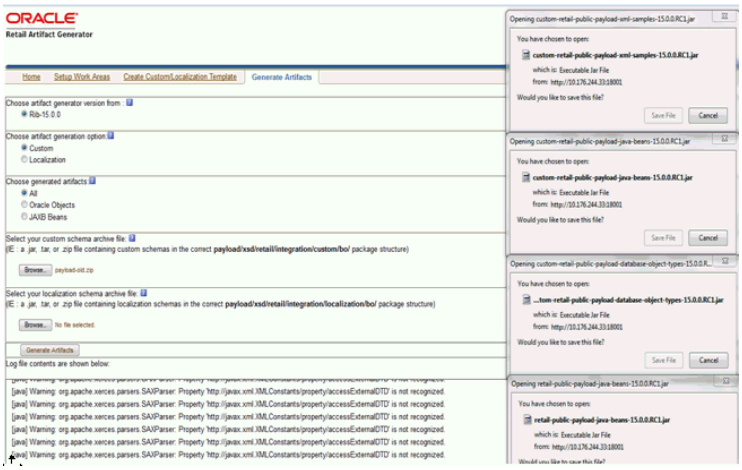
After modifying the zip file follow the steps to generate functional artifacts using GUI Generate Artifacts tab.

Choose artifact generator version and artifact generation option (custom/localization).

Select the saved schema archive file (custom template or localization template) by clicking on the Browse button.



Click on Generate Artifacts button to generate functional artifacts.



As the process runs, the status of its progress is displayed in the log window. When the process completes, the save file dialog windows will display for retail-public-payload-java-beans-<version>.jar, custom-retail-payload-java-beans-<version>.jar, custom-retail-public-payload-xml-samples-<version>.jar, and the custom-retail-public-payload-database-object-types-<version>.jar with options to save the archive files or open them for review.

A

Appendix: Install the Artifact Generator

This appendix provides step-by-step instruction (with illustrations) for installing the Artifact Generator as a Web application in Oracle WebLogic.

Creating the Role/User in the Artifact Generator

To create a role/user in the Artifact Generator, do the following:

1. Log in to the WLS. In the **Domain Structure** menu, click **Security Realms**.
2. Click **myrealm** and then on **Users and Groups**.
3. Click **Groups**. Create a new group called **agAdminGroup**.
4. Click **Users**. Add a new user.
5. Under settings for the new user, click the **Groups** tab. Add the agAdminGroup to the new user.

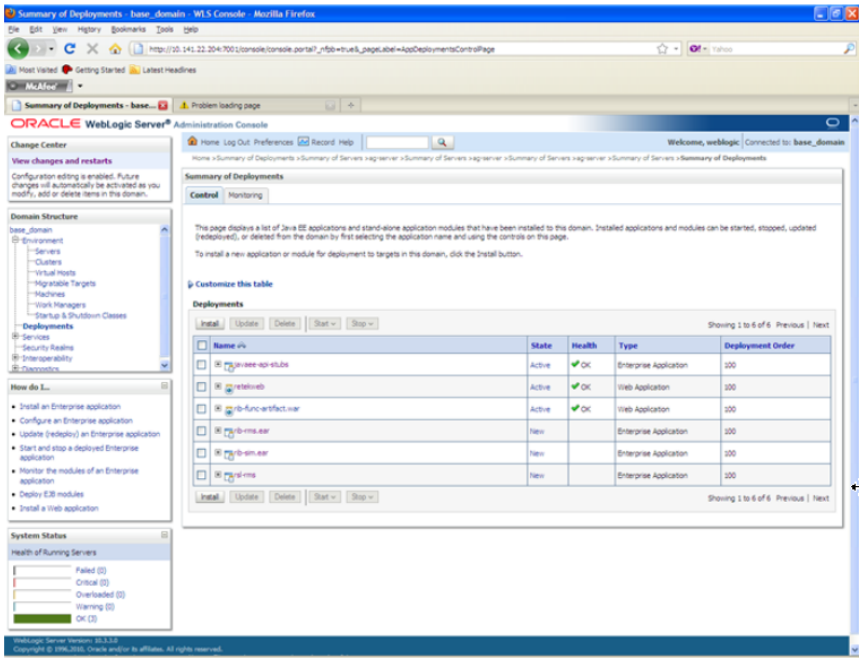
Installing as a Web Application in Oracle WebLogic

This section describes the steps you need to complete to install the Artifact Generator as a Web application in Oracle WebLogic.

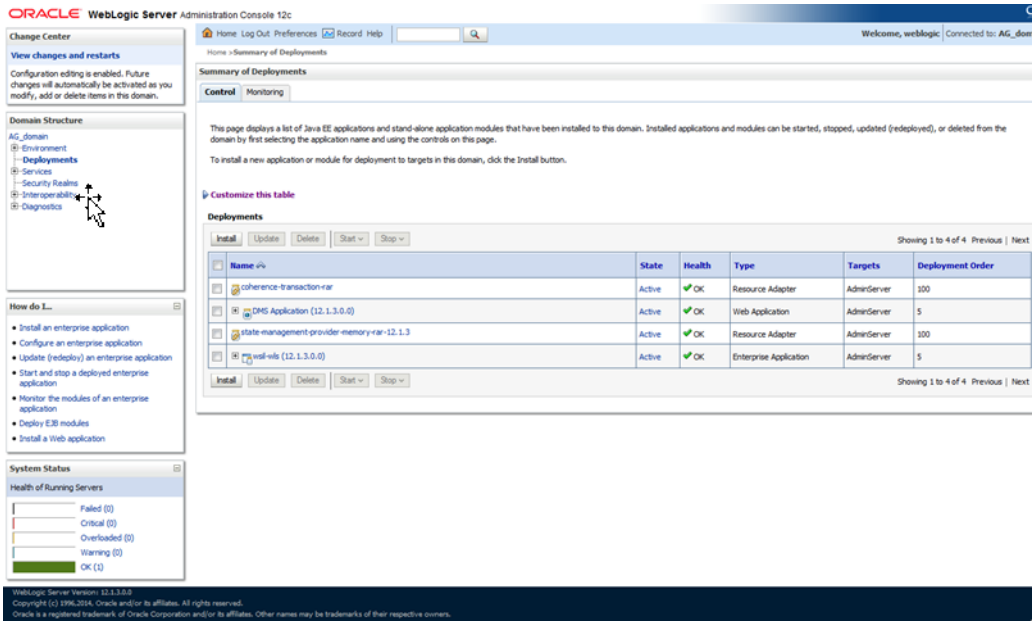
Deploying the Artifact Generator Application

Using the WebLogic Server Administration Console, complete the following steps:

1. Navigate to the Deployments page.



2. Click **Lock & Edit.**
3. Click **Install**.

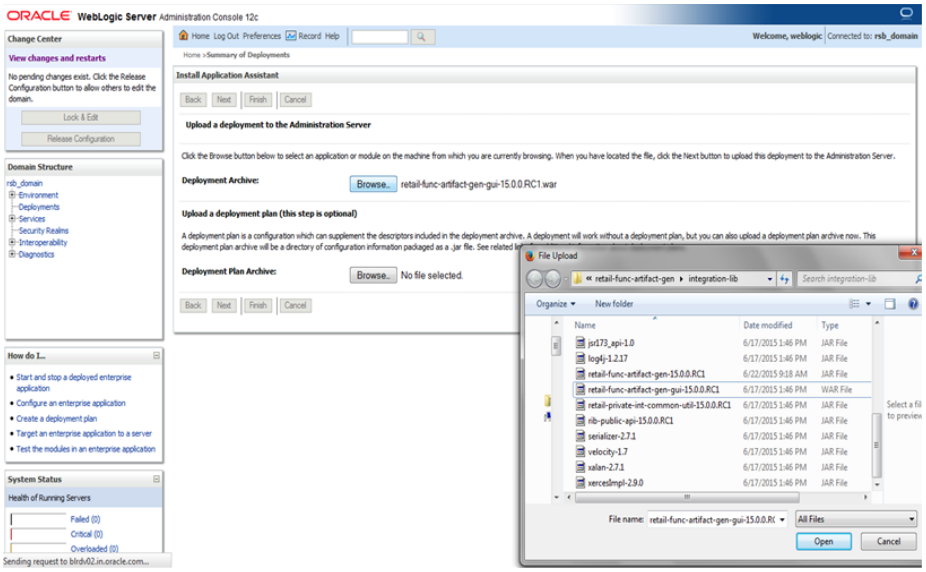


Note: If the application has already been installed, see the Redeploying the Application section.

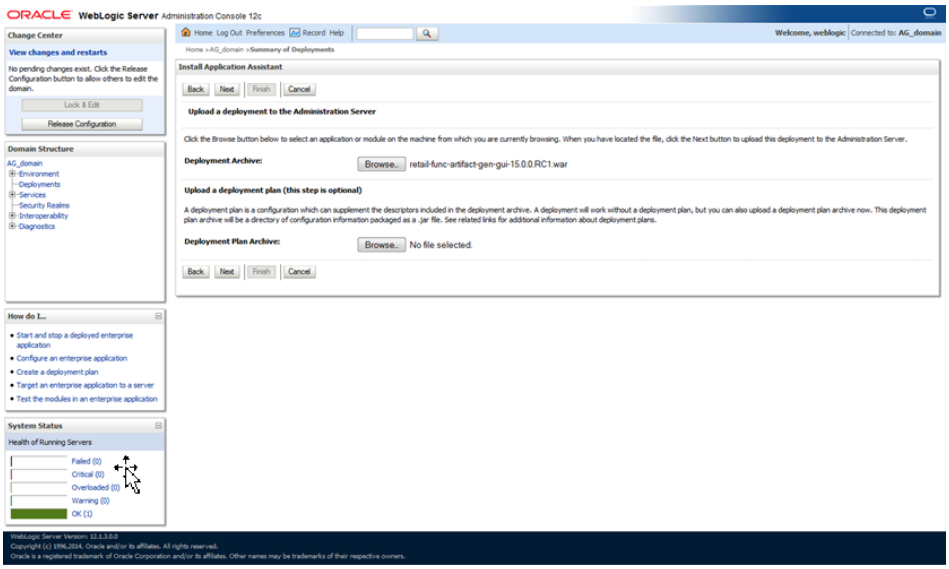
The **Locate deployment to install and prepare for deployment** page is displayed. Follow the instructions to locate the retail-func-artifact-gen-gui-<version>.war file.

4. Select **Upload your File(s)**.

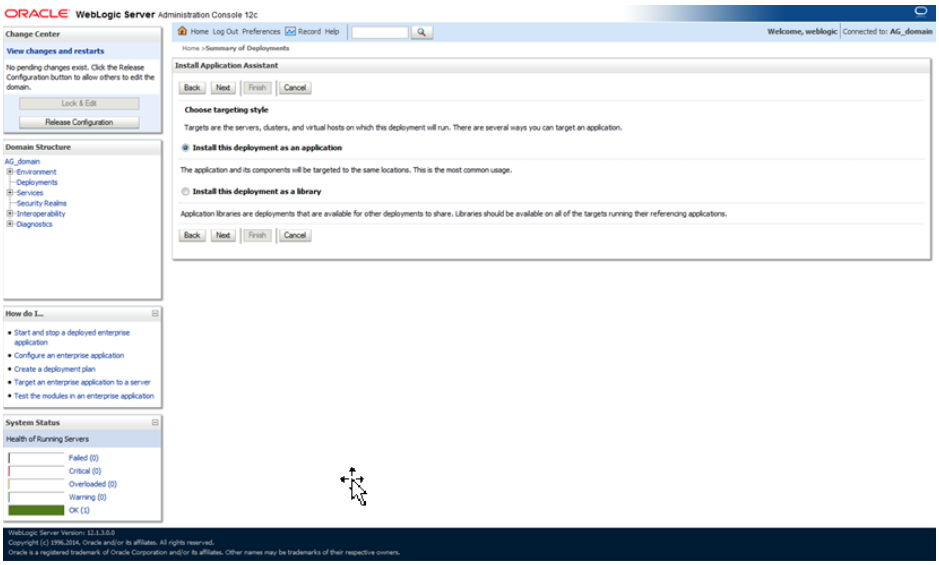
5. On the **Upload a Deployment to the administration server** page, use the Browse button to locate the retail-func-artifact-gen-gui-<version>.war file in the Deployment Archive.



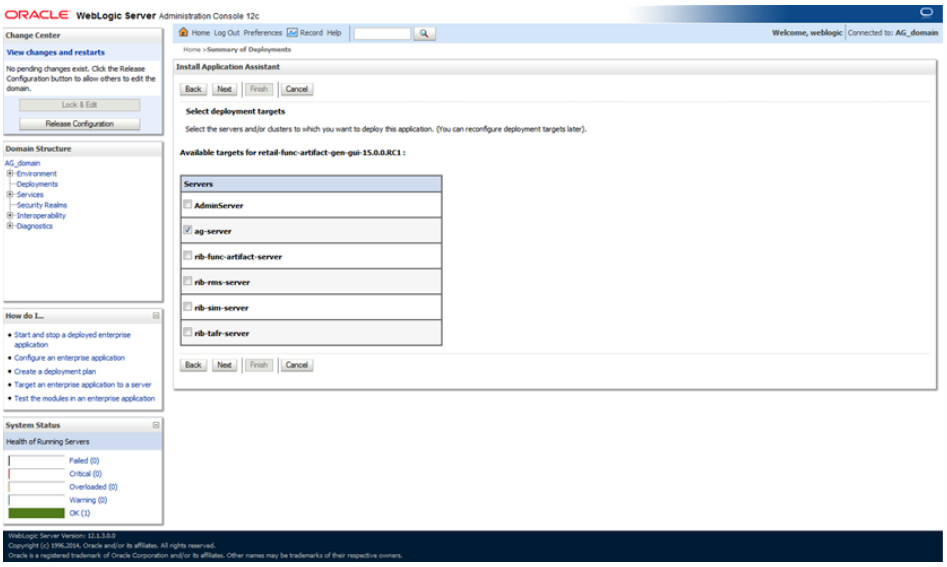
6. Select **retail-func-artifact-gen-gui**-<*version*>**.war**.



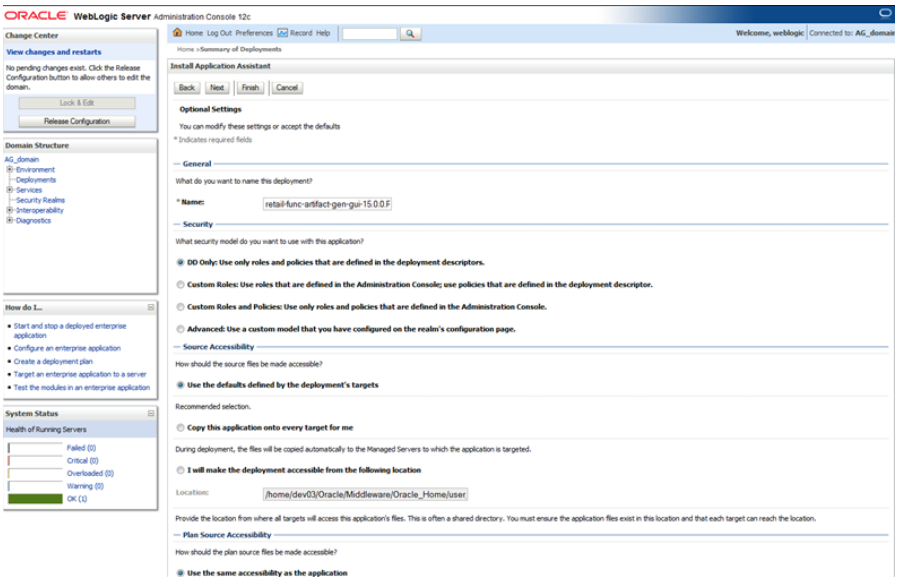
7. Click **Next** and move to **Choose targeting style**.



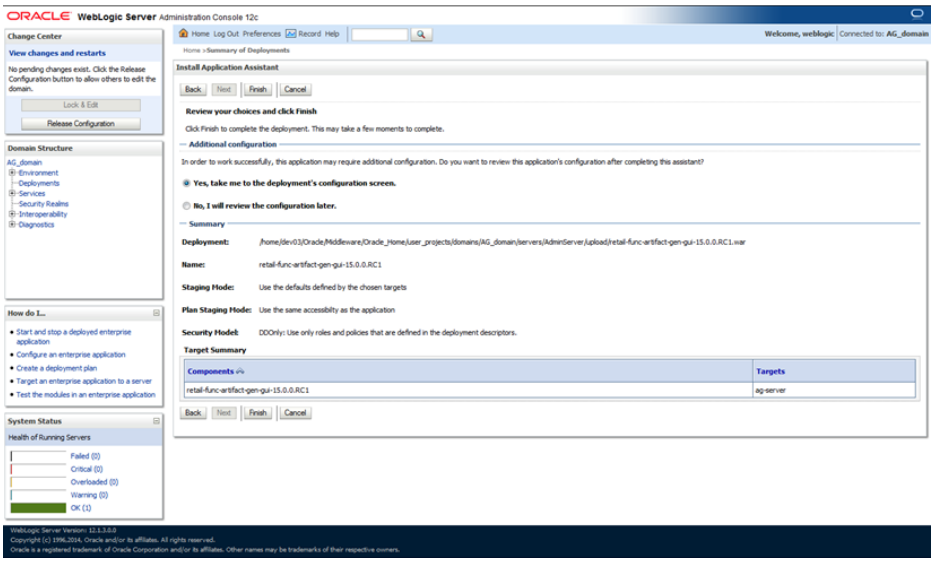
8. Select **Install this deployment as an application**.
9. Click **Next** and move to **Select deployment targets**.



10. Click the managed server to which the application will be deployed. Click **Next** and move to **Optional Settings**.

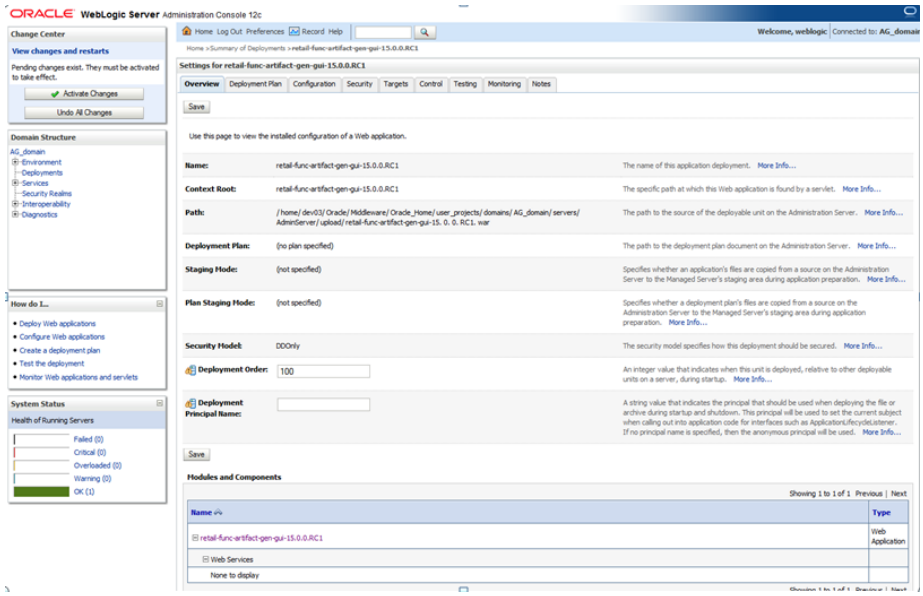


11. Select **Next** and move to **Review your choices and click Finish**.

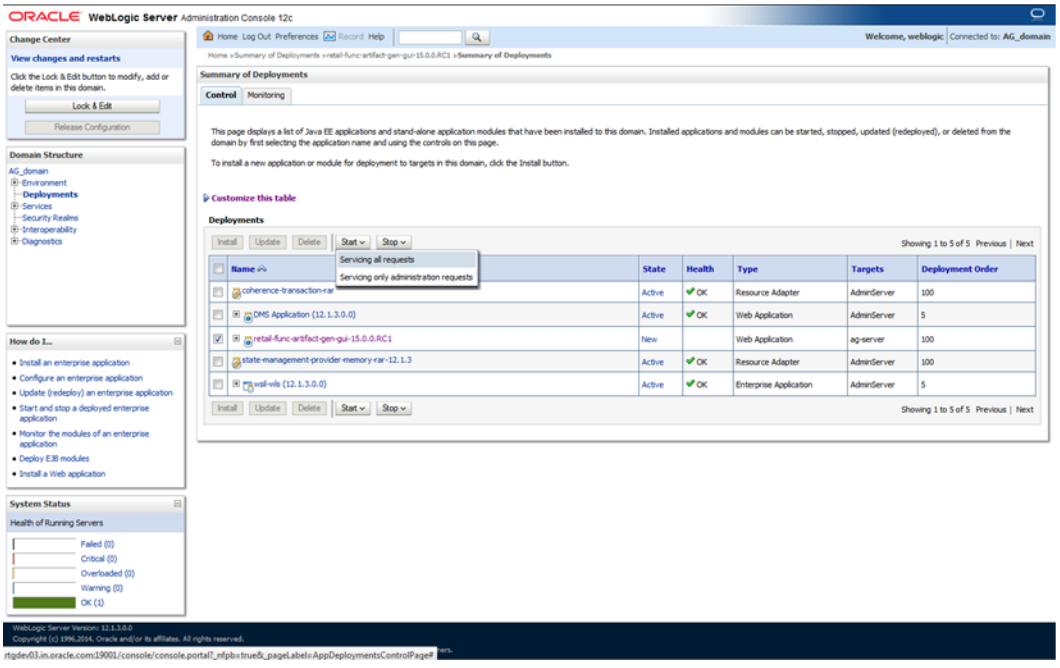


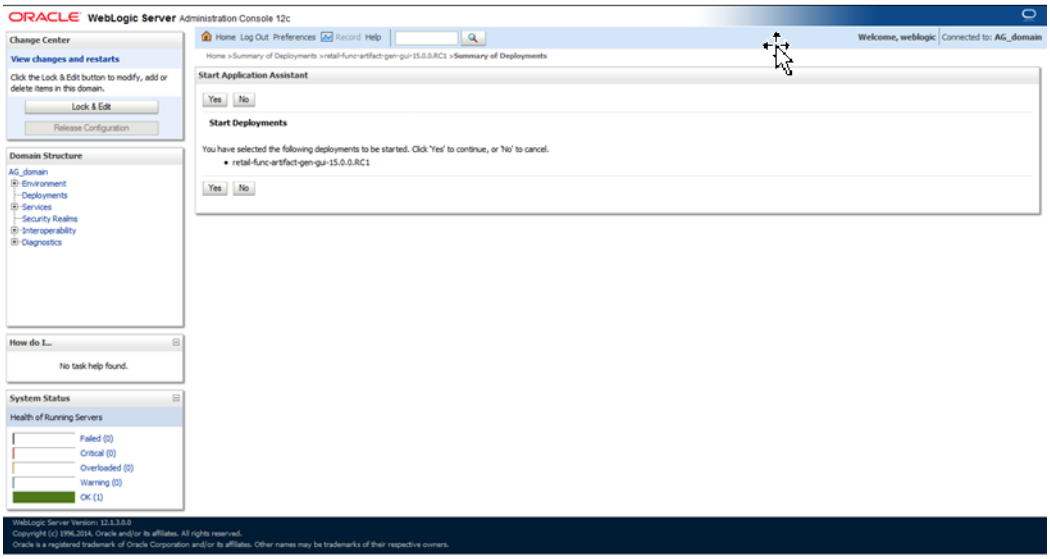
12. Select **No, I will review the configuration later**.

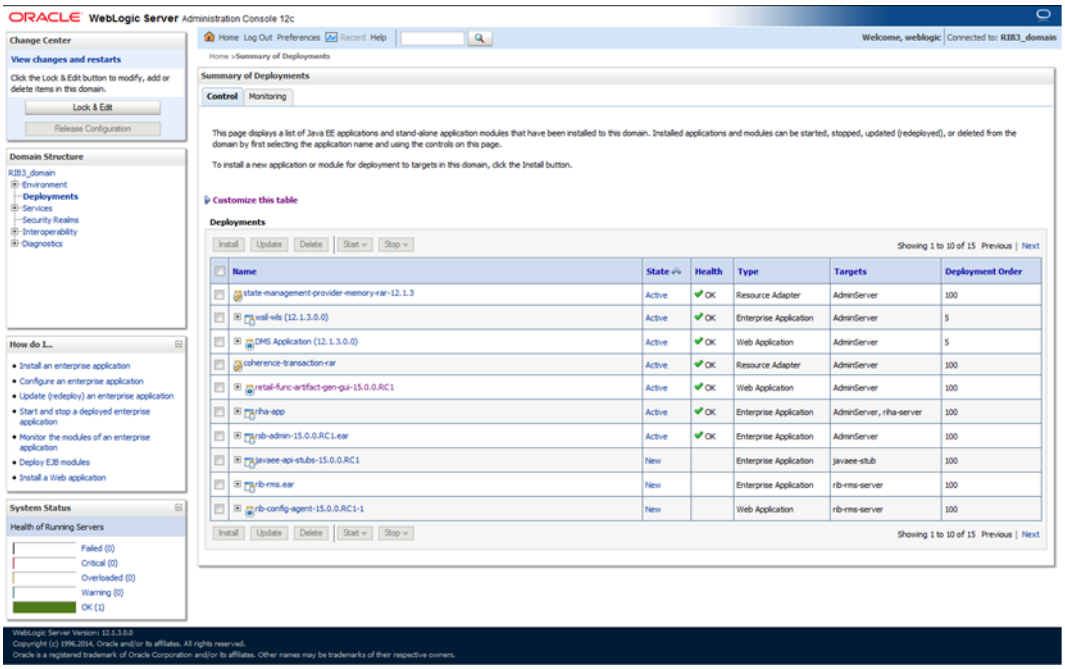
13. Click **Finish** to deploy the application.



14. Click **Activate Changes**.
15. Click **Deployment.**
16. On the right pane, check retail-func-artifact-gen-gui-<version>.
17. Start the application by choosing 'Servicing all requests' from Start dropdown.



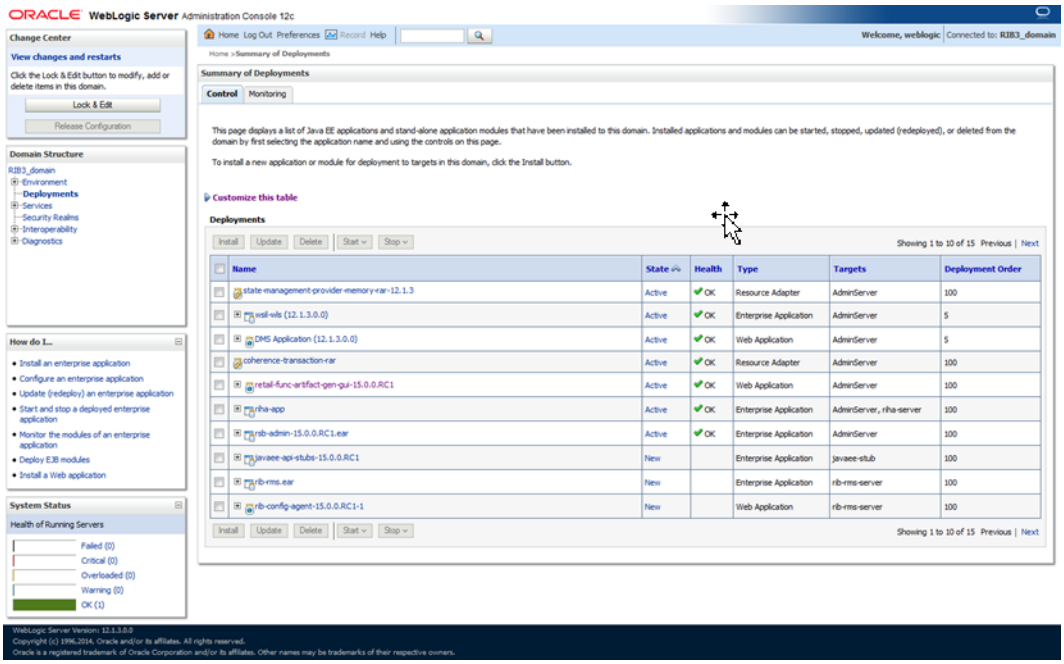




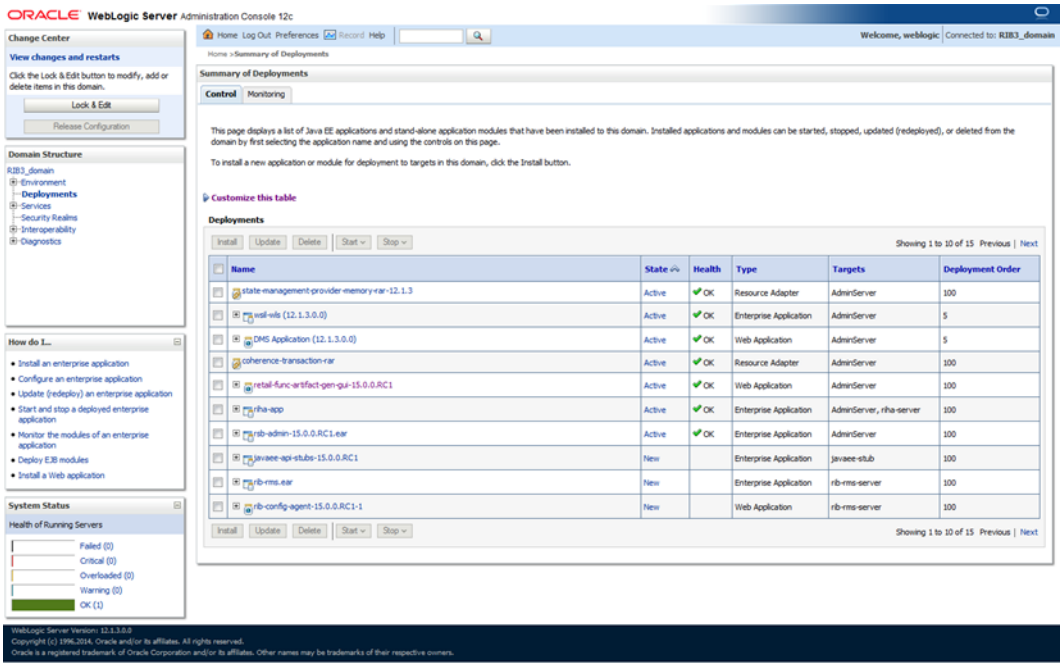
18. the application status is **Active.**

Verifying the Artifact Generator Web Application

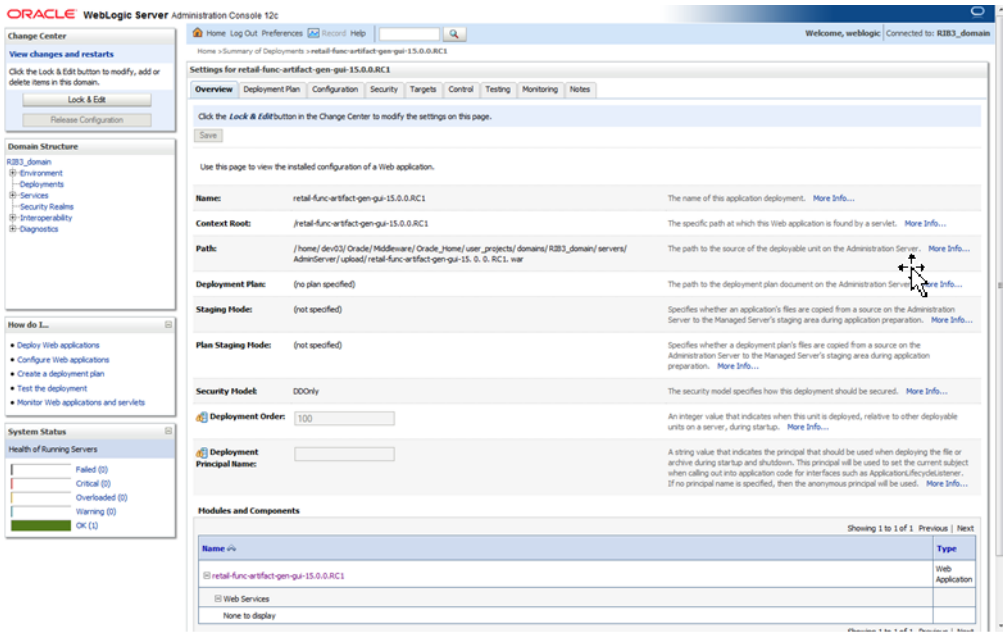
1. Navigate to the Deployments page.



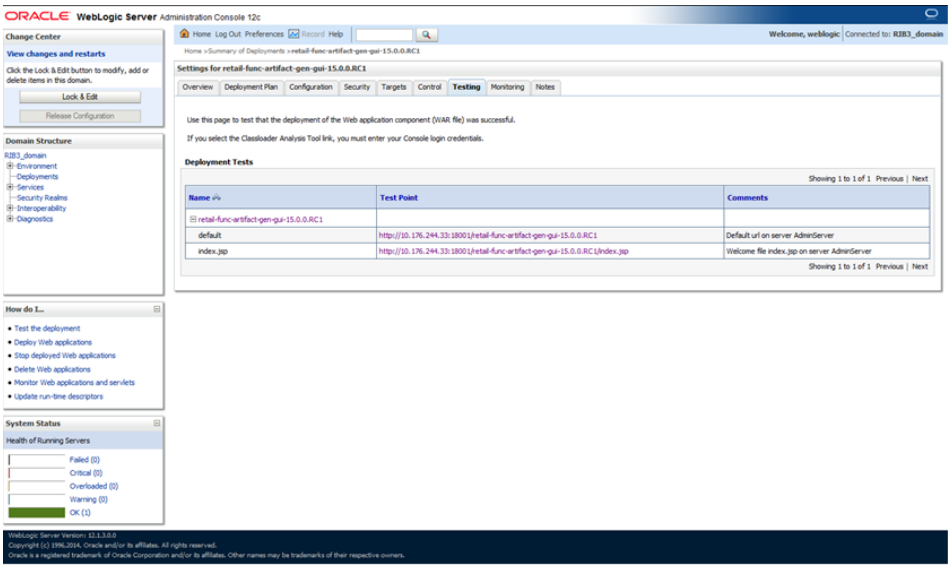
2. Locate the **retail-func-artifact-gen-gui-<version>** on the Summary of Deployments page.



3. Click the name, **retail-func-artifact-gen-gui-<version>,** to move to the **Settings for the rib-func-artifact-gen-gui-version>.**



4. Select the **Testing** tab.

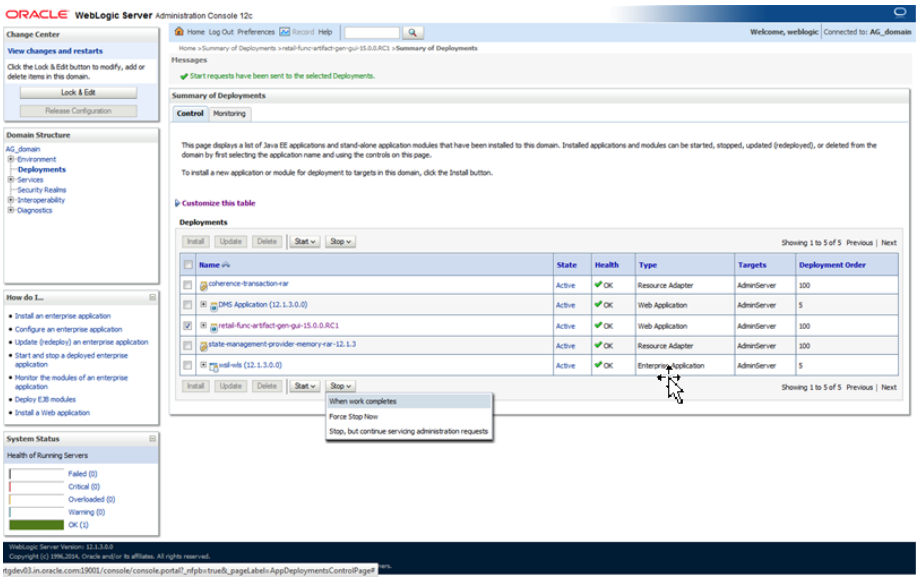


5. Click **index.jsp URL** in the Test Point. The URL should open to the Retail Artifact Generator Login page. The installation is complete.

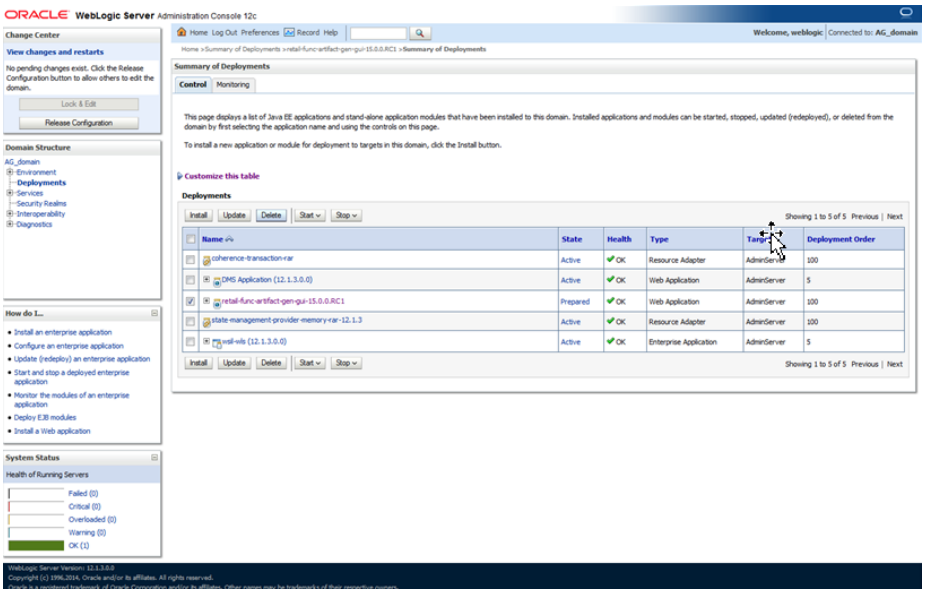
Redeploying the Application

If the retail-func-artifact-gen-gui-<version> application has already been deployed, follow these steps:

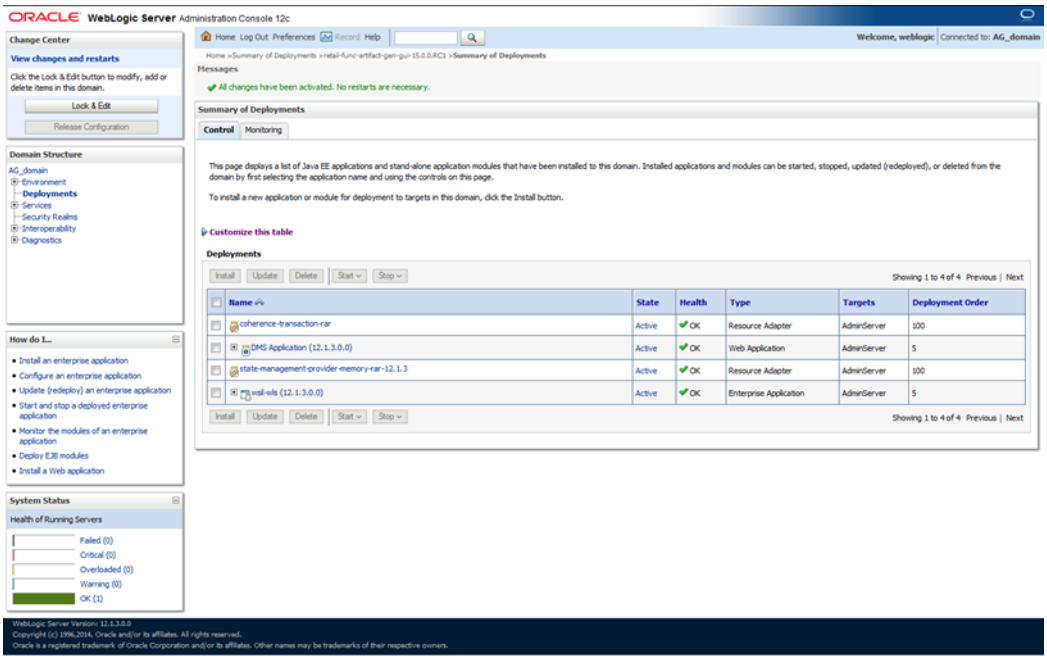
1. If the retail-func-artifact-gen-gui-<version> application is running, select **Stop** and **When Work Completes** or **Force Stop Now**, depending on the environment. The recommended option always is **When Work Completes.**



2. Click Lock & Edit. Select Delete.



3. Click **Activate Changes.**



4. Return to the appropriate step in the section, Deploying the Artifact Generator Application."

B

Appendix: Example of Customization and Localization for ASNInDesc

This appendix provides a sample customization and localization of ASNInDesc, which is used as input to create custom and localization templates in various examples in this document.

Note that the XSD contains no extension hooks (such as import statements and reference elements). After the custom template is created, the extension hook for customization is added to ASNInDesc.xsd.

The reference element is added to ASNInDesc complex type, as shown in the following screen.

Similarly, for localization (such as Brazil), the import statement and reference elements are added to ASNInDesc.xsd, as shown in the following screen.

The custom template, ExtOfASNInDesc.xsd, is created from the tool, as shown in the following screen.

```
<!-- ########################################################################
This is a generated Custom XSD template for the complex type
Please follow these steps in the same order
STEP 1: Add your custom elements at the designated location below and save the file
STEP 2: Run "GenerateArifacts" using GUI or CLI option
STEP 3: Verify the artifacts to ensure customization is as expected

For more information refer to the documentation doc
######################################################################################
-->
<xs:schema elementFormDefault="qualified"
    targetNamespace="http://www.oracle.com/retail/integration/custom/bo/ExtOfASNInDesc/v1"
    version="1.0"
    xmlns="http://www.oracle.com/retail/integration/custom/bo/ExtOfASNInDesc/v1"
    xmlns:retailDoc="http://www.w3.org/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="ExtOfASNInDesc">
        <xs:complexType>
            <xs:sequence>
                <!-- Adding Element in XSD -->
                <!-- xs:element maxOccurs="1" minOccurs="0" name="elementName" type="elementType">
                    <retailDoc:annotation>
                        <retailDoc:documentation>Description of the element</retailDoc:documentation>
                    </retailDoc:annotation>
                </xs:element>
                -->
            </xs:sequence>
        </xs:complexType>
    </xs:element>
<!-- Adding simple Type -->
<!-- <xs:simpleType name="elementType">
        <retailDoc:annotation>
            <retailDoc:documentation>Description of Simple Type</retailDoc:documentation>
        </retailDoc:annotation>
        <xs:restriction base="xs:string">
            <xs:maxLength value="maxValue"/>
        </xs:restriction>
    </xs:simpleType>
-->
</xs:schema>
```

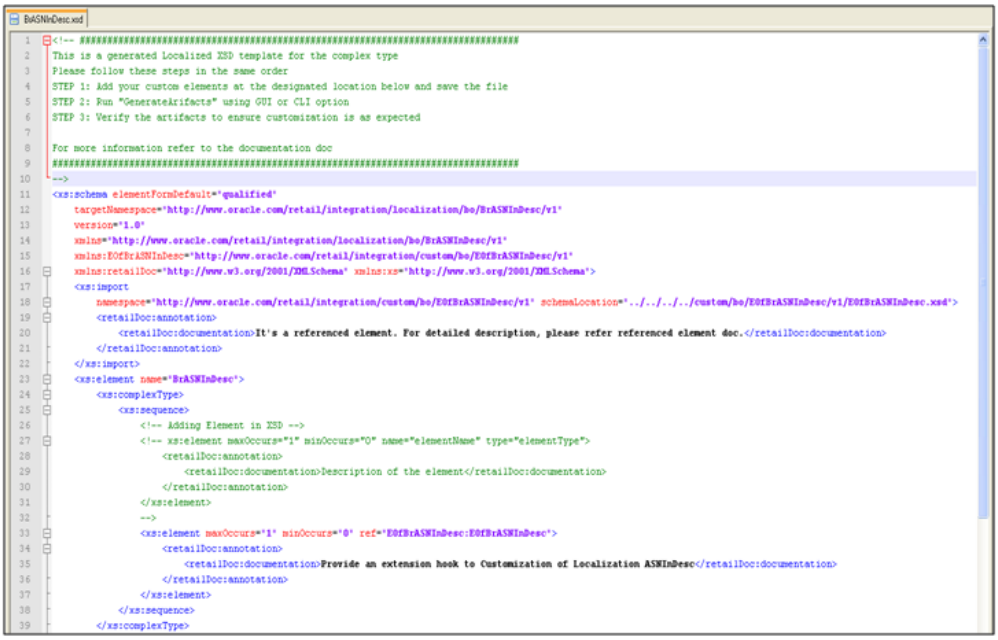
The localization template created from the tool appears as follows in the illustrations of LOCOfASNInDesc.xsd, BrASNInDesc.xsd, and EOfBrASNInDesc.xsd.

LocOfASNInDesc.xsd

```
<!-- ########################################################################
This is a generated Localization XSD template for the complex type
Please follow these steps in the same order
STEP 1: Add your custom elements at the designated location below and save the file
STEP 2: Run "GenerateArifacts" using GUI or CLI option
STEP 3: Verify the artifacts to ensure customization is as expected

For more information refer to the documentation doc
######################################################################################
-->
<xs:schema elementFormDefault="qualified"
    targetNamespace="http://www.oracle.com/retail/integration/base/bo/LocOfASNInDesc/v1"
    version="1.0"
    xmlns="http://www.oracle.com/retail/integration/base/bo/LocOfASNInDesc/v1"
    xmlns:BrASNInDesc="http://www.oracle.com/retail/integration/localization/bo/BrASNInDesc/v1"
    xmlns:retailDoc="http://www.w3.org/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <retailDoc:annotation>
        <retailDoc:documentation>This is root element of this document which contains name space definitions for the document elements.</retailDoc:documentation>
    </retailDoc:annotation>
    <xs:import
        namespace="http://www.oracle.com/retail/integration/localization/bo/BrASNInDesc/v1" schemaLocation="../../../../localization/bo/BrASNInDesc/v1/BrASNInDesc.xsd">
        <retailDoc:annotation>
            <retailDoc:documentation>It's a referenced element. For detailed description, please refer referenced element doc.</retailDoc:documentation>
        </retailDoc:annotation>
    </xs:import>
    <xs:element name="LocOfASNInDesc">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="1" minOccurs="0" ref="BrASNInDesc:BrASNInDesc">
                    <retailDoc:annotation>
                        <retailDoc:documentation>Provide an extension hook to localize ASNInDesc</retailDoc:documentation>
                    </retailDoc:annotation>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

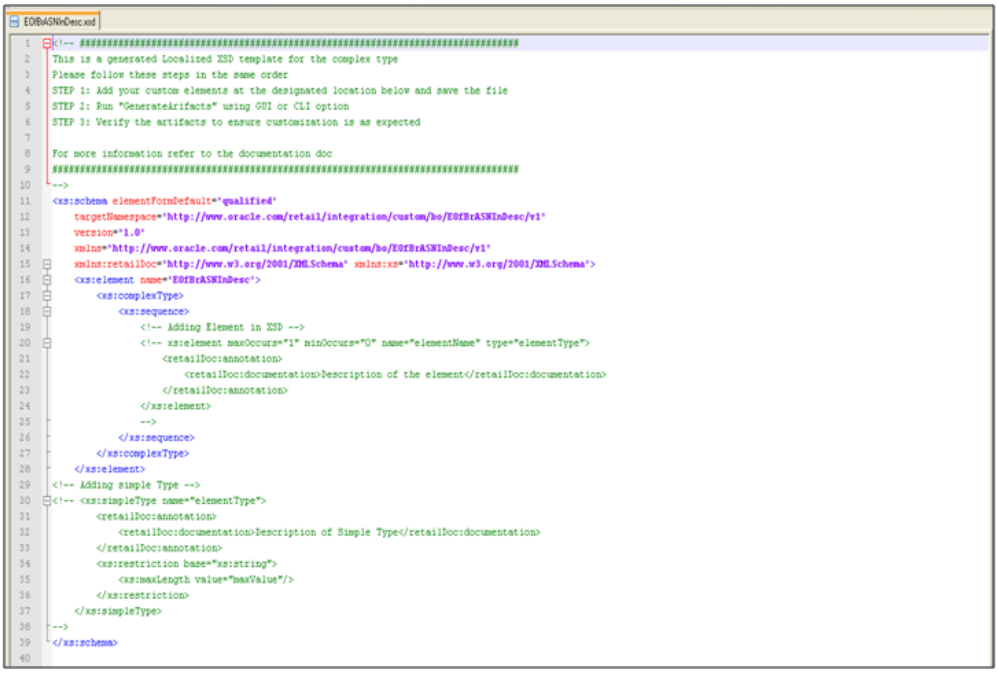
BrASNInDesc.xsd



```
<!-- ########################################################################
This is a generated Localized XSD template for the complex type
Please follow these steps in the same order
STEP 1: Add your custom elements at the designated location below and save the file
STEP 2: Run "GenerateArifacts" using GUI or CLI option
STEP 3: Verify the artifacts to ensure customization is as expected

For more information refer to the documentation doc
######################################################################################
-->
<xs:schema elementFormDefault="qualified"
    targetNamespace="http://www.oracle.com/retail/integration/localization/bo/BrASNInDesc/v1"
    version="1.0"
    xmlns="http://www.oracle.com/retail/integration/localization/bo/BrASNInDesc/v1"
    xmlns:EOfBrASNInDesc="http://www.oracle.com/retail/integration/custom/bo/EOfBrASNInDesc/v1"
    xmlns:retailDoc="http://www.w3.org/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:import
        namespace="http://www.oracle.com/retail/integration/custom/bo/EOfBrASNInDesc/v1" schemaLocation="../../../../custom/bo/EOfBrASNInDesc/v1/EOfBrASNInDesc.xsd">
        <retailDoc:annotation>
            <retailDoc:documentation>It's a referenced element. For detailed description, please refer referenced element doc.</retailDoc:documentation>
        </retailDoc:annotation>
    </xs:import>
    <xs:element name="BrASNInDesc">
        <xs:complexType>
            <xs:sequence>
                <!-- Adding Element in XSD -->
                <!-- xs:element maxOccurs="1" minOccurs="0" name="elementName" type="elementType">
                    <retailDoc:annotation>
                        <retailDoc:documentation>Description of the element</retailDoc:documentation>
                    </retailDoc:annotation>
                </xs:element>
                -->
                <xs:element maxOccurs="1" minOccurs="0" ref="EOfBrASNInDesc:EOfBrASNInDesc">
                    <retailDoc:annotation>
                        <retailDoc:documentation>Provide an extension hook to Customization of Localization ASNInDesc</retailDoc:documentation>
                    </retailDoc:annotation>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
```

EOfBrASNInDesc.xsd



```
<!-- ########################################################################
This is a generated Localized XSD template for the complex type
Please follow these steps in the same order
STEP 1: Add your custom elements at the designated location below and save the file
STEP 2: Run "GenerateArifacts" using GUI or CLI option
STEP 3: Verify the artifacts to ensure customization is as expected

For more information refer to the documentation doc
######################################################################################
-->
<xs:schema elementFormDefault="qualified"
    targetNamespace="http://www.oracle.com/retail/integration/custom/bo/EOfBrASNInDesc/v1"
    version="1.0"
    xmlns="http://www.oracle.com/retail/integration/custom/bo/EOfBrASNInDesc/v1"
    xmlns:retailDoc="http://www.w3.org/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="EOfBrASNInDesc">
        <xs:complexType>
            <xs:sequence>
                <!-- Adding Element in XSD -->
                <!-- xs:element maxOccurs="1" minOccurs="0" name="elementName" type="elementType">
                    <retailDoc:annotation>
                        <retailDoc:documentation>Description of the element</retailDoc:documentation>
                    </retailDoc:annotation>
                </xs:element>
                -->
            </xs:sequence>
        </xs:complexType>
    </xs:element>
<!-- Adding simple Type -->
<!-- <xs:simpleType name="elementType">
        <retailDoc:annotation>
            <retailDoc:documentation>Description of Simple Type</retailDoc:documentation>
        </retailDoc:annotation>
        <xs:restriction base="xs:string">
            <xs:maxLength value="maxValue"/>
        </xs:restriction>
    </xs:simpleType>
-->
</xs:schema>
```

C

Appendix: References

The following is a list of reference materials providing more information on the topics covered in this guide.

- [XML] Extensible Markup Language (XML) 1.0 (Second Edition), http://www.w3.org/TR/REC-xml
- [XMLSchema1] W3C Recommendation, XML Schema Part 1: Structures, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
- [XMLSchema2] W3C Recommendation, XML Schema Part 2: Datatypes, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/