

**Oracle Utilities Smart Grid Gateway
Adapter Development Kit**

Configuration Guide

Release 2.1.0 Service Pack 3

E41193-03

May 2015

Oracle Utilities Smart Grid Gateway Adapter Development Kit Configuration Guide

E41193-03

Copyright © 2012, 2015 Oracle and/or its affiliates. All rights reserved.

Primary Author: UGBU Documentation

Contributing Author: UGBU Documentation

Contributor: UGBU Documentation

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.

Contents

Chapter 1

Overview	1-1
What is the Adapter Development Kit?	1-2
What Is This Book?.....	1-3
Other Documentation.....	1-3
Oracle Utilities Application Framework Configuration Tools.....	1-5

Chapter 2

Implementing OSB Processing for One-Way Commands	2-1
Overview.....	2-2
Prerequisites	2-2
Related Documentation	2-2
Oracle Utilities Smart Grid Gateway Adapters.....	2-4
Adapter Components	2-4
Processing Life Cycle.....	2-5
The Java Project	2-6
OSB Configuration.....	2-17
Setting Up the Adapter Environment.....	2-28
Additional Information	2-31
OSB Processing Terms and Acronyms.....	2-36

Chapter 3

Configuring BPEL Processing for Two-Way Commands	3-1
WSDLs, Endpoints, and XAL.....	3-2
How to locate WSDLs and Endpoints.....	3-2
Components of the Composites	3-4
Composite Properties	3-4
Proxies	3-6
Process Callouts	3-6
Configuring and Customizing the Adapter BPEL Processes	3-7
Editing Configuration Files	3-7
Customizing the BPEL Processes	3-8
Commands.....	3-9
Common.....	3-9
Commission/Decommission	3-11
Connect/Disconnect.....	3-12
Device Status Check	3-15
On Demand Read	3-18
Working with Enterprise Manager	3-21
MultiSpeak Implementation.....	3-23
MultiSpeak Head End System Service Definitions.....	3-23
MultiSpeak Head End XML Schemas	3-23
The Test Harness.....	3-24
Locating the WSDL for the Test Harness	3-24
Web Services.....	3-26

The Sample Meters File.....	3-40
-----------------------------	------

Chapter 4

Configuring Oracle Utilities Smart Grid Gateway.....	4-1
Understanding the Adapter.....	4-2
Understanding the Adapter Processing.....	4-3
Initial Measurement Data Load and Device Events.....	4-3
Device Communication	4-4
Configuring a Head-End System	4-10
XAI Inbound Services.....	4-10
XAI Senders.....	4-11
Outbound Message Types	4-12
External System.....	4-12
Service Provider	4-12
Processing Methods for Service Provider	4-13
Configuring Extendable Lookups.....	4-16
Device Event Mapping	4-16
Head-End UOM Code to Standard UOM Mapping	4-16
Interval Status Code to Condition Mapping.....	4-17
Extending the Adapter Development Kit	4-18

Appendix

Glossary	5-1
----------------	-----

Index

Chapter 1

Overview

This chapter provides an overview of this configuration guide and an introduction to the Oracle Utilities Smart Grid Gateway Adapter Development Kit, including the following:

- **What is the Adapter Development Kit?**
- **What Is This Book?**
- **Other Documentation**
- **Oracle Utilities Application Framework Configuration Tools**

What is the Adapter Development Kit?

The Oracle Utilities Smart Grid Gateway Adapter Development Kit provides a starting point for customers to create their own customized adapter for using Oracle Utilities Smart Grid Gateway with a specific head-end system.

The Oracle Utilities Smart Grid Gateway Adapter Development Kit includes the following:

- **Sample Oracle Service Bus Process:** A file parsing and transformation process framework, delivered as an Oracle Service Bus (OSB) configuration, that can be used as a reference for creating the functionality to import usage readings and device events from a specific head-end system.

See **Chapter 2: Implementing OSB Processing for One-Way Commands** for more information about the OSB process provided with the Adapter Development Kit.

- **Sample Oracle Business Process Execution Language Processes:** A set of sample communication processes that support Remote Connect, Remote Disconnect, Commission, Decommission, Device Status Check, and On-Demand Read commands delivered as an Oracle Business Process Execution Language (BPEL) project.

See **Chapter 3: Configuring BPEL Processing for Two-Way Commands** for more information about the BPEL processes provided with the Adapter Development Kit.

- **Oracle Utilities Smart Grid Gateway Business Objects:** Business objects to support two-way commands such Commission, Decommission, Remote Connect, Remote Disconnect, and Device Status Check.

See **Chapter 4: Configuring Oracle Utilities Smart Grid Gateway** for more information about these business objects and the general configuration options used by the adapters developed with the Oracle Utilities Smart Grid Gateway Adapter Development Kit.

- **Sample Two-Way Communication Test Harness:** A head-end system emulator delivered as a BPEL composite.
- **Demonstration Data:** A set of sample data provided in the demonstration database that can be used with the adapter processes as delivered (prior to any customization you might perform as part of your implementation).

What Is This Book?

This guide describes how to configure the Oracle Utilities Smart Grid Gateway Adapter Development Kit. It is intended for implementers and system administrators responsible for configuration and initial setup of the application.

The Oracle Utilities Smart Grid Gateway Adapter Development Kit is based on the Oracle Utilities Application Framework (OUAF). For information about using and configuring basic Framework functions, see the Oracle Utilities Application Framework documentation.

This guide includes the following chapters:

- **Chapter 1: Overview** (this chapter) provides an overview of the Oracle Utilities Smart Grid Gateway Adapter Development Kit architecture and of the configuration tools and process used in implementing the product.
- **Chapter 2: Implementing OSB Processing for One-Way Commands** provides an overview of how to use the Oracle Utilities Smart Grid Gateway Adapter Development Kit and Oracle Service Bus (OSB) to develop a customized adapter for importing usage and event information from a head-end system.
- **Chapter 3: Configuring BPEL Processing for Two-Way Commands** describes the BPEL processes included in the Oracle Utilities Smart Grid Gateway Adapter Development Kit that support two-way communications.
- **Chapter 4: Configuring Oracle Utilities Smart Grid Gateway** provides an overview of some general configuration options used by the adapters developed with the Oracle Utilities Smart Grid Gateway Adapter Development Kit.

Other Documentation

This section describes other documentation provided with the Oracle Utilities Smart Grid Gateway Adapter Development Kit.

Installation Documentation

Installation documentation describes the steps involved in the installation and initial set up of the system, and includes the following documents:

- Oracle Utilities Smart Grid Gateway Quick Install Guide
- Oracle Utilities Smart Grid Gateway DBA Guide
- Oracle Utilities Smart Grid Gateway Installation Guide

User Documentation

User documentation provides conceptual information and procedures related to working with the various objects used in the system, and includes the following documents:

- Oracle Utilities Application Framework Business Process Guide
- Oracle Utilities Application Framework Administration Guide
- Oracle Utilities Service and Measurement Data Foundation User's Guide
- Oracle Utilities Smart Grid Gateway User's Guide

Supplemental Documentation

Supplemental documentation provides technical information related to system administration tasks and include the following documents:

- Oracle Utilities Smart Grid Gateway Server Administration Guide
- Oracle Utilities Smart Grid Gateway Batch Server Administration Guide

- Oracle Utilities Smart Grid Gateway Configuration Guide

The Oracle Utilities Smart Grid Gateway Adapter Development Kit uses Oracle Service Bus (OSB) and Oracle Business Process Execution Language (BPEL) as middleware components. These tools are part of the Oracle SOA Suite. See the Oracle SOA Suite Documentation library (<http://www.oracle.com/technetwork/middleware/soasuite/documentation/index.html>) for more information about using these tools.

Embedded Help

Oracle Utilities Smart Grid Gateway, like all Oracle Utilities Application Framework applications, provides extensive internal documentation. For example, detailed descriptions of system objects are included in the objects' maintenance portals. The life cycle of each business object is described on the Lifecycle tab and depicted in flow diagrams on the Summary tab. This information is extremely useful for implementers and system administrators.

Embedded help is provided for all non-obvious fields in most portals and zones. If a field has associated help text, a ? icon appears next to the field when the zone is displayed.

Online Help

Oracle Utilities Smart Grid Gateway also includes context-sensitive help for all the user interface screens users will typically work with as they use the system. Online help contains conceptual information and procedures related to working with the various objects used in the system.

The online help is divided into the following three sections:

- Oracle Utilities Application Framework: Describes the features and functions of the application framework (F1)
- Oracle Utilities Service and Measurement Data Foundation: Describes the features and functions provided in the service and measurement data foundation (D1)
- Oracle Utilities Smart Grid Gateway: Describes the features and functions provided in the adapter development kit application (DG)

Oracle Utilities Application Framework Configuration Tools

Please refer to the general configuration guide for information on the Oracle Utilities Application Framework (OUAF) configuration tools that can be used to create and customize system entities, such as business objects, portals, zones, and UI maps. Refer to the Oracle Utilities Application Framework configuration tools documentation for instructions on using tools such as:

- **Configuration Process Overview**
- **Data Areas**
- **Algorithms**
- **Entity Naming Conventions**

This configuration guide does not duplicate the concepts and procedures presented in the Oracle Utilities Application Framework configuration tools documentation; rather, it will identify the specific objects used by the Oracle Utilities Smart Grid Gateway Adapter Development Kit that can be configured and customized using the configuration tools, as well as application parameters and objects that can be managed within the application components themselves.

This guide assumes that all individuals responsible for system configuration and implementation will be familiar with the Oracle Utilities Application Framework and will have completed training on the Oracle Utilities Application Framework Configuration Tools.

Chapter 2

Implementing OSB Processing for One-Way Commands

This chapter describes how to use Oracle Service Bus (OSB) and the components of the Oracle Utilities Smart Grid Gateway Adapter Development Kit to develop an adapter for importing usage reading and device events. It includes the following sections:

- **Overview**
- **Oracle Utilities Smart Grid Gateway Adapters**
- **OSB Processing Terms and Acronyms**

Overview

The examples in this chapter provide an implementer with the information required to create a customized adapter using the Oracle Utilities Smart Grid Gateway Adapter Development Kit. Customized adapters are required in situations where a productized Smart Grid Gateway adapter is not available for a particular head-end system. The examples below explain how to receive and process inbound meter usage and events. The examples contain two parts: the incoming file parsing functionality implemented in Java, and an easy, configurable and customizable Oracle Service Bus (OSB) configuration, including an implementation of post-parsing functionality.

Prerequisites

Development of an Oracle Utilities Smart Grid Gateway adapter requires a number of technical skills as well as a development environment.

Required Technical Skills

The following technical skills are required for developing an adapter using the development kit:

- Experience in Java programming
- Experience in development using XQuery and XPath
- Experience in XML programming
- Experience in OSB development and administration
- Understanding of Oracle Utilities Service and Measurement Data Foundation (SMDF) architecture

Environment Requirements

Development of the Oracle Utilities Smart Grid Gateway adapter requires an environment configured as follows:

- The Oracle WebLogic server must be installed.
- The OSB must be installed.
- An OSB domain must be created.
- Java Messaging Service (JMS) queues for Initial Measurement Data (IMD) and Notification messages must be created.
- The `spl-d1-osb-2.0.1.jar` containing the `com.splwg.d1.sgg.osb.common` Java package must be available. This is a single jar for all vendor-specific adapters.
- The archive `sgg-osb-generic-adapter.zip` must be available. This file is downloaded as a part of adapter development kit deployment. It contains the `FileParser` folder with sample source code and the `com.splwg.dg.sgg.osb.configuration.jar` file with sample OSB projects.

Related Documentation

For more information about using the adapter development kit and Oracle Service Bus, see the following Oracle documents:

- *Oracle Fusion Middleware User's Guide for Technology Adapters 11g Release 1*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus 11g Release 1*
- *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*
- *Oracle Fusion Middleware User's Guide for Oracle JDeveloper*
- *Oracle Utilities Smart Grid Gateway Configuration Guide*

- *Oracle Utilities Service and Measurement Data Foundation User's Guide*

Oracle Utilities Smart Grid Gateway Adapters

This section contains detailed information on the different components that make up an Oracle Utilities Smart Grid Gateway adapter and the application logic that is needed to implement one. It includes the following sections:

- **Adapter Components**
- **Processing Life Cycle**
- **The Java Project**
- **OSB Configuration**
- **Setting Up the Adapter Environment**
- **Additional Information**

Adapter Components

The following table lists the adapter components:

Component	Description
JCA File Adapter (FA)	Technology adapter for reading and writing files on the local file system. It is responsible for polling files from incoming folder and passing it to the Generic File Processor.
Generic File Processor (GFP)	Framework component that is implemented in Java. It is responsible for instantiation of the File Parser, getting Plain XML from it, and passing it to OSB message Flow. It also performs payload statistics related functionality such as gathering data and generating notification messages.
File Parser (FP)	<p>Component implemented in Java. It is specific to every different incoming format. It is responsible for parsing the incoming file, breaking the payload into logical parts (debatching), generating Plain XML for every logical part, and returning XML back to invoking GFP.</p> <p>Plain XML could be any logical subset of data that can be mapped to IMD. It is an intermediate format/schema between the raw data and IMD XML. The Plain XML schema must be defined by the file parser developer.</p> <p>Raw data is read in portions from the input file and converted to Plain XML before passing to OSB message flow. The reason for data being read in portions is as follows:</p> <p>Assume that an input file contains readings for 100 Measuring Components. You would want to read the input file in portions for two particular reasons.</p> <ol style="list-style-type: none"> 1. Reading data of exactly one measuring component ensures that it is mapped appropriately to one IMD at a time. 2. Reading data in portions ensures that the entire file is not loaded into memory, which could cause resource issues.
Inbound Proxy Service	OSB proxy service that contains FA-related configuration settings. It statically routes all messages to Processing proxy Service.

Component	Description
Processing Proxy Service (PPS)	OSB proxy service that validates and transforms Plain XML. This service is necessary for catching errors that occur before Plain XML is transformed to IMD or Device Event seeder structures.
Result Processing Proxy Service (RPPS)	OSB proxy service that processes transformed data. This service is necessary for catching errors that occur after Plain XML is transformed to IMD or Device Event seeder structures.

Processing Life Cycle

See the **Logic Sequence Diagram** on page 2-6 for general description of how the components interact. The life cycle from the initial input file to the IMD output can be traced as follows:

OSB Processes

1. The JCA File Adapter starts reading the file.
2. FA instantiates and initializes the GFP.
3. FA invokes GFP passing an open stream to the incoming file.
4. GFP sends **D1-PayloadStatistics** notification message to FA.
5. GFP instantiates and initializes (if it is not done yet) the FP that is defined in the Inbound Proxy properties for parsing the file.
6. GFP invokes the FP (See **File Parser Processes** below).
7. In case of error in FP it invokes GFP. GFP sends **D1-PayloadErrorNotif** notification message to FA.
8. GFP gets the Plain XML as a return from FP.
9. GFP returns the Plain XML to the FA.
10. The Plain XML is passed via InboundProxyService to Processing Proxy service (PPS). In PPS's message flow, the Plain XML is validated and transformed to IMD XML.
11. PPS passes the IMD to Result Processing Proxy service (RPPS). RPPS publishes the IMD XML to a JMS queue, which is then picked up by Message Driven Bean (MDB).
12. OSB processes 3 through 9 are repeated until FP returns NULL on process 9.
13. GFP sends **D1-PayloadSummary** notification message to FA.

File Parser Processes

1. FP starts reading the input stream
2. FP instantiates a JAXB object of the type of Plain XML. The JAXB related interfaces and implementation classes are generated by using JDeveloper (See *Oracle Fusion Middleware User's Guide for Oracle JDeveloper* or **Generating Java Classes from XML Schemas with JAXB** on page 2-35).
3. FP reads the input stream in segments until it hits a logical end of data with which it could create a Plain XML.
4. FP populates the data that is read into the Plain XML JAXB object.

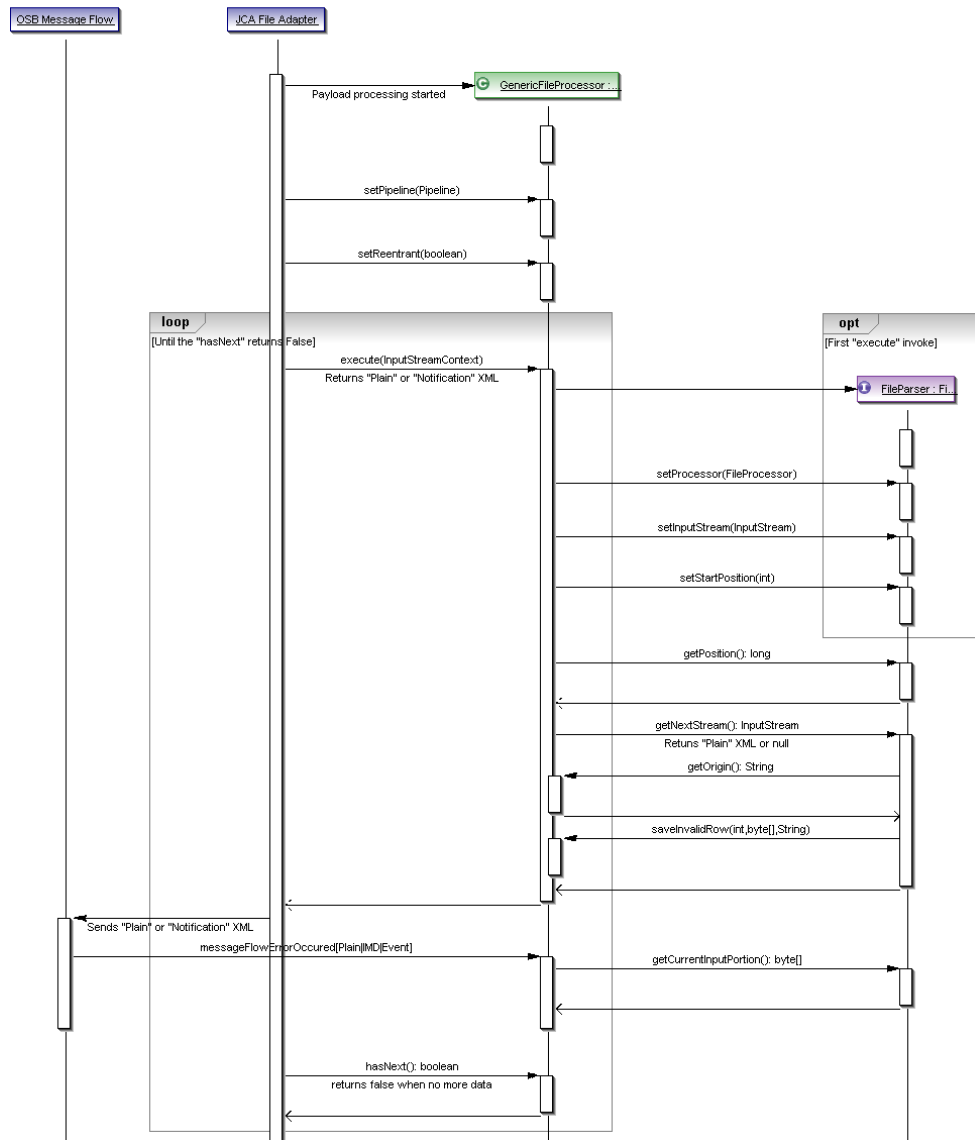
5. FP marshals the JAXB object into Plain XML and passes back GFP. If the end of file (EOF) is reached FP passes back NULL.
6. File parser processes 2 through 5 are performed for every invocation by GFP.

The Java Project

The source code in the example files contains an implementation of functionality to parse an incoming payload in Comma Separated Values (CSV) and XML formats. This implementation can be used as a reference for creating a project that is specific to your head-end system. The descriptions below are mostly related to CSV parsing functionality as it is a more complex and interesting example. The XML related parsing example is provided only to illustrate how XML-based input formats may also be processed.

Logic Sequence Diagram

The sequence diagram below is helpful for understanding the File Parser functionality. The custom class must implement the FileParser and FileParser2 Java interfaces described in the SGG D1 jar to be qualify as a file parser. For more information, see **FileParser Interface** on page 2-31 and **FileParser2 Interface** on page 2-34.



When FA finds a file in the input directory, it instantiates and calls GFP which in turn starts an interaction with the FP. The interaction can be categorized into the following three phases:

1. Instantiation and Initialization
 - `setProcessor()`
 - `setInputStream()`
 - `setStartPosition()`
2. Transaction
 - `getPosition()`
 - `getNextStream()`
3. Exception Handling
 - `getCurrentInputPortion()`

Instantiation and Initialization Phase

The GFP is instantiated and receives an Input Stream when a file gets into the input directory. At that point an instance of the FP is instantiated. Certain methods, as specified above, are invoked on the FP to initialize.

Transaction Phase

Once the FP is initialized, GFP uses the FP instance to generate the Plain XML structure. This happens with the invocation of `getNextStream()` method on the FP.

On every invocation, the FP returns exactly one Plain XML. The FP invocations continues until NULL is returned from the FP. NULL indicates the end of file and an indication that parsing is complete for a specific file. GFP stops invoking `getNextStream()` for a particular input file when it receives a NULL from the FP.

FP uses the file input stream to read the file in portions. It could read byte-by-byte or line-by-line, but at the end of each `getNextStream()` method call it would have read only so much that is sufficient to create one Plain XML structure. For example, in an input file, if there is sufficient data for only one Plain XML then the first call to `getNextStream()` would return a Plain XML and the next one would return NULL. But if there is more than one, `getNextStream()` calls would continue to return Plain XML structures for every incoming portion until end of file is reached.

Exception Handling Phase

Exception handling inside the FP can be categorized into two phases: **Reactive** and **Recovery**:

Reactive Phase

This phase involves catching an exception and reporting it to the GFP. It is achieved by invoking `saveInvalidRow()` method on the GFP (`this.fileProcessor`).

```

} catch (Exception ex) {
    this.fileProcessor.saveInvalidRow(this.curPortionStart,
                                    this.getCurrentInputPortion(),
                                    ex.toString());
}

```

Note that when an exception occurs, no further parsing is done and NULL is passed back from `getNextStream()`.

If one fails to return NULL, errors such as “Simultaneous good and bad result returned by the File Parser” can be found in the Weblogic server log.

When an exception is reported to the GFP, it does three things:

1. Creates an XML payload D1-PayloadErrorNotif and passes it to the OSB message flow. The message is published to the Notification Queue from inside PPS of the BASE OSB project.

2. It increments the “transaction error occurred” variable by invoking the utility method inside the D1 jar. All the errors occurring during the lifetime of a file parsing are captured and are reported through the message D1-PayloadSummary - which is again published to Notification Queue.
3. Finally, it creates a data file in the error directory with the portion of raw data that is read when the error occurred. It also creates a Rejected File Descriptor (RFD) file. The raw input portion is available by utilizing `getCurrentInputPortion()` method, which is discussed below.

As noted in the above image, the `saveInvalidRow()` method takes three input parameters:

1. The position at which the parsing started. It will be written to an RFD file. This is helpful in identifying the location of an error in the input file.
2. The raw data that will be written into error data file. It should be in the same to the incoming file format and it should contain data that is read since current invocation of `getNextStream()` method is started. It will allow correcting troubled data and place “fixed” file into an incoming folder for further re-processing.
3. The error message that is reported in D1-PayloadErrorNotif message. Note, the content of the error message has to be defined by the file parser developer.

Recovery Phase

This phase must not be confused with recovering from a handled exception discussed above. This case is when file parsing gets completely interrupted by power failure or network interruption.

Preparation for recovery: GFP maintains an internal index to the current read position of the file. This is achieved by GFP by invoking `getPosition()` method on the FP before every `getNextStream()` method call.

```
public long getPosition() {
    long retVal = this.fileProcessor.getCurrentInputPosition();
    return retVal;
}
```

Recovery: In case of an interruption and subsequent restore, GFP sets the start position on the File Parser. For instance, if the failure had occurred at byte 415, the start position would be set to 415 so that FP starts to read from that point and not from 0. The value 415 can be referred to as the recovery point.

```
public void setStartPosition(long position) {
    this.startPosition = position;
}
```

When the file parser starts parsing again, two things must occur to ensure that it starts parsing from the recovery point:

1. Store the recovery point to the `startPosition` field.
2. Incorporate skip logic to start the pointer from the recovery point.

Java Implementation

The Java component of the project includes the following classes and packages:

1. **com.splwg.dg.osb.common.FileParserGenGeneric** class is a super class with members that are common for CSV and XML related functionality. It implements the `com.splwg.d1.sgg.osb.common.FileParser` interface (see **FileParser Interface** on page 2-31). The implementation of `com.splwg.d1.sgg.osb.common.FileParser2` (see **FileParser2 Interface** on page 2-34) has been added to define the IMD & Event Upload Statistics related functionality.
2. **com.splwg.dg.osb.csv.FileParserCSV** class contains the implementation of the parsing functionality for CSV formats. It performs the following logic:

- Parses the incoming payload.
 - Transforms incoming data to Usage or Event related structures based on incoming data type.
 - Extends those structures with file information and external service provider identifier.
 - Returns one by one that structures as a stream to caller.
3. **com.splwg.dg.osb.plain** package contains JAXB related interfaces and implementation classes corresponding to the Plain XML Schema
 4. **com.splwg.dg.osb.xml.FileParserXML** class contains the implementation of parsing functionality for the “IMD and Event Online Upload” XML format (See the *Oracle Utilities Service and Measurement Data Foundation User's Guide*). It performs the following logic:
 - Parses the incoming payload.
 - Extracts information about current device.
 - Breaks the incoming data to separate initial measurement data structures.
 - Extends those structures with device information, file information and external service provider identifier.
 - Returns one by one the structures as a stream to caller.

Implementation Details

This section provides an overview of the implementation details. See the comments in the source code for more detailed information. It includes the following sections:

- **Parsing CSV Files**
- **Parsing XML Files**

Parsing CSV Files

This section describes the file parsing functionality for CSV format files.

Input File Format

The data is stored as a CSV file. Note that values in this example cannot contain a literal comma character. This restriction is added for the simplicity of file parser for the example implementation only. The parsing logic can be re-implemented to avoid this restriction as needed. It will not affect the rest of functionality.

Input file contains the following record types:

- Interval usage
- Device Events
- Trail record

Interval Usage Format

#	Field Name	Type	Definition
1	Record Type ID	Constant value “U”	Record type identifier
2	Start Date/Time	Time in the Unix Time format.	Meter read's start date/time
3	End Date/Time	Time in the Unix Time format.	Meter read's end date/time
4	Device Id	Arbitrary Text	Device identifier

#	Field Name	Type	Definition
5	Interval Duration	Numeric Integer	The time interval between readings in seconds.
6	UOM	Arbitrary Text	Describes the units of the data values
7	Data Entry(s)	Numeric Floating-Point and optional Arbitrary Text separated by “:”	Each record can contain unlimited number of data entry fields. Each data entry is a set of two fields - a reading value and an optional reading status flag. Fields are separated by “:” character. The separator does not present when the reading status flag is not provided.

Device Event Format

#	Field Name	Type	Definition
1	Record Type ID	Constant value “E”	Record type identifier
2	Event Date/Time	Time in the Unix Time format.	The date and time at which the event occurred
3	Device Id	Arbitrary Text	Device identifier
4	Event Name	Arbitrary Text	The primary identification name of the event

Trail Record Format

#	Field Name	Type	Definition
1	Record Type ID	Constant value “T”	Record type identifier
2	Creation Date/Time	Time in the Unix Time format.	Meter read's start date/time
3	Total Records	Numeric Integer	Displays number of records in file

Sample File

```

U,1,86400,DEVICE_DG_0,900,KWH,1:S,2.5,-3.99:R,4:B
E,86400,DEVICE_DG_1,Power Outage
U,2,86400,DEVICE_DG_2,900,KWH,1:S,2,3:R,4:B,5,6,7:A
E,86402,DEVICE_DG_3,Tamper attempt suspected
U,3,86400,DEVICE_DG_4,900,KWH,1:S
T,86401,5

```

XML Schema of the Plain Format

Plain XML consists of both the Interval Usage and Device Event related elements:

```

<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://xmlns.oracle.com/GenericAdapter"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ga="http://xmlns.oracle.com/GenericAdapter">
  <xs:element name="Payload" type="ga:PayloadType"/>
  <xs:complexType name="PayloadType">
    <xs:sequence>
      <xs:element name="Origin" type="ga:NonEmptyString"/>
      <xs:element name="ServProvExtRefId" type="ga:NonEmptyString"/>
      <xs:choice>
        <xs:element name="Usage" type="ga:UsageType"/>
        <xs:element name="Event" type="ga:EventType"/>
      </xs:choice>
      <xs:element name="RawData" type="ga:NonEmptyString"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="UsageType">
    <xs:sequence>
      <xs:element name="RecordType" type="ga:RecordTypeUsageType"/>
      <xs:element name="StartDateTime" type="xs:integer"/>
      <xs:element name="EndDateTime" type="xs:integer"/>
      <xs:element name="DeviceId" type="ga:NonEmptyString"/>
      <xs:element name="IntervalDuration" type="xs:integer"/>
      <xs:element name="UOM" type="ga:NonEmptyString"/>
      <xs:element name="Intervals" type="ga:IntervalList"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="RecordTypeUsageType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="U"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="IntervalList">
    <xs:sequence maxOccurs="unbounded">
      <xs:element name="Interval" type="ga:IntervalType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="IntervalType">
    <xs:sequence>
      <xs:element name="Value" type="xs:float"/>
      <xs:element name="Status" type="ga:NonEmptyString"
minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="EventType">
    <xs:sequence>
      <xs:element name="RecordType" type="ga:RecordTypeEventType"/>
      <xs:element name="DateTime" type="xs:integer"/>
      <xs:element name="DeviceId" type="ga:NonEmptyString"/>
      <xs:element name="Name" type="ga:NonEmptyString"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="RecordTypeEventType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="E"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NonEmptyString">

```

```

    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>

```

Incoming file information

The name of the incoming file will be placed into the Payload/Origin element.

Service Provider information

The Service Provider External Reference ID will be placed into the Payload/ServProvExtRefId element. It is accessed by calling the FileProcessor2.getServiceProviderExternalReferenceId method.

Raw Data

A part of incoming file that been used to generate current Plain XML structure will be placed into the Payload/RawData element.

Interval Usage to Plain XML Mapping

The following table shows the mapping between fields in the incoming interval data and child elements of Payload/Usage element in the Plain XML format:

Interval Usage Field	Plain XML element
Record Type ID	RecordType
Start Date/Time	StartDateTime
End Date/Time	EndDateTime
Device Id	DeviceId
Interval Duration	IntervalDuration
UOM	UOM
Data Entry(s)	Intervals/ Interval/Value [Status]
Record content from incoming file.	RawData

Device Event to Plain XML Mapping

The following table shows the mapping between fields in the incoming interval data and child elements of Payload/Event element in the Plain XML format:

Interval Usage Field	Plain XML element
Record Type ID	RecordType
Event Date/Time	DateTime
Device Id	DeviceId
Event Name	Name
Record content from incoming file.	RawData

Java Code

The following methods are the main elements of the implementation for parsing files in CSV format:

Method	Description
getNextStream()	The tasks that need to be performed by getNextStream() method can be broken down into a two distinct steps: <ul style="list-style-type: none"> • Read the next portion of data from the input file. • Parse the read data and transform it into Plain XML Each of the above steps translates to methods in FP: <ul style="list-style-type: none"> • readFields(). • processData()
readFields()	This method is called from loadNextPortion() method. It reads the raw data, skips empty lines and splits raw data into separate strings.
processData()	This method is called from loadNextPortion() method for the read data. It analyzes the record type and calls corresponding methods to parse usage, device event or trailing information: <ul style="list-style-type: none"> • processUsageData() • processEventData() • processTrailData()
processUsageData()	This method transforms the usage data into Plain XML.
processEventData()	This method transforms the device event data into Plain XML.
processTrailData()	This method stores fields from the trail record to be used later when GFP calls FP for the Upload Stats related information.

Parsing XML Files

This section describes the file parsing functionality for XML format files.

Input File Format

The file format is described in the *Oracle Utilities Smart Grid Gateway Configuration Guide Release 2.0.1.8*.

Sample File

```
<deviceList>
  <device>
    <headEnd>L&G</headEnd>
    <headEndExternalId>L+G</headEndExternalId>
    <deviceId></deviceId>
    <deviceIdentifierNumber>DEV-OUSGG-OSB-DEMO-001</
deviceIdentifierNumber>
    <initialMeasurementDataList>
      <initialMeasurementData>
        <preVEE>
          <mcIdN></mcIdN>
          <uom>KWH</uom>
          <stDt>2010-05-19-00.00.00</stDt>
          <enDt>2010-05-19-00.30.00</enDt>
          <spi>900</spi>
          <msrs>
            <mL>
```

```

                <s>1</s>
                <q>0.2316</q>
            </mL>
            <mL>
                <s>2</s>
                <q>0.1416</q>
            </mL>
        </msrs>
    </preVEE>
</initialMeasurementData>
<initialMeasurementData>
    <initialMeasurementDataId>testValue</
initialMeasurementDataId>
    <preVEE>
        <mcIdN></mcIdN>
        <uom>KWH2</uom>
        <stDt>2010-05-19-00.30.00</stDt>
        <enDt>2010-05-19-01.00.00</enDt>
        <spi>900</spi>
        <msrs>
            <mL>
                <s>1</s>
                <q>1.2316</q>
            </mL>
            <mL>
                <s>2</s>
                <q>2.1416</q>
            </mL>
        </msrs>
    </preVEE>
</initialMeasurementData>
</initialMeasurementDataList>
</device>
</deviceList>

```

XML Schema of Plain Format

The Plain XML consists of the IMD related elements.

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
    <xsd:element name="initialMeasurementData">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="initialMeasurementDataId" type="xsd:string"
minOccurs="0"/>
                <xsd:element name="preVEE" minOccurs="0">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="imdType" minOccurs="0">
                                <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                        <xsd:enumeration value="D1IL"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:element>
                            <xsd:element name="externalId" type="xsd:string"
minOccurs="0"/>
                            <xsd:element name="simdId" type="xsd:string"
minOccurs="0"/>
                            <xsd:element name="dvcIdN" type="xsd:string"
minOccurs="0"/>

```



```

        <xsd:element name="mcId" type="xsd:string" minOccurs="0"/>
    >
        <xsd:element name="mcIdN" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="uom" type="xsd:string" minOccurs="0"/>
        <xsd:element name="externalUOM" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="stDt" type="ouafDateTime"
minOccurs="0"/>
        <xsd:element name="stQty" type="xsd:decimal"
minOccurs="0"/>
        <xsd:element name="enDt" type="ouafDateTime"
minOccurs="0"/>
        <xsd:element name="enQty" type="xsd:decimal"
minOccurs="0"/>
        <xsd:element name="inShift" minOccurs="0">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="notShifted"/>
                    <xsd:enumeration value="shifted"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element name="mcm" type="xsd:decimal" minOccurs="0"/>
    >
        <xsd:element name="nd" type="xsd:decimal" minOccurs="0"/>
        <xsd:element name="tz" type="xsd:string" minOccurs="0"/>
        <xsd:element name="spi" type="xsd:int" minOccurs="0"/>
        <xsd:element name="ccond" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="sts" minOccurs="0">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="stsL" minOccurs="0"
maxOccurs="unbounded">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element name="s" type="xsd:decimal"
minOccurs="0"/>
                                <xsd:element name="st" type="xsd:string"
minOccurs="0"/>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="msrs" minOccurs="0">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="mL" minOccurs="0"
maxOccurs="unbounded">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element name="s" type="xsd:decimal"/>
                                <xsd:element name="dt" type="ouafDateTime"
minOccurs="0"/>
                                <xsd:element name="q" type="xsd:decimal"
minOccurs="0"/>
                                <xsd:element name="ue" type="xsd:string"
minOccurs="0"/>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>

```

```

                                <xsd:element name="fc" type="xsd:string"
minOccurs="0"/>
                                <xsd:element name="sts" minOccurs="0">
                                    <xsd:complexType>
                                        <xsd:sequence>
                                            <xsd:element name="stsL" minOccurs="0"
maxOccurs="unbounded">
                                                <xsd:complexType>
                                                    <xsd:sequence>
                                                        <xsd:element name="s"
type="xsd:decimal" minOccurs="0"/>
                                                            <xsd:element name="st"
type="xsd:string" minOccurs="0"/>
                                                                </xsd:sequence>
                                                            </xsd:complexType>
                                                        </xsd:element>
                                                    </xsd:sequence>
                                                </xsd:complexType>
                                            </xsd:element>
                                        </xsd:sequence>
                                    </xsd:complexType>
                                </xsd:element>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="rawIMD" type="xsd:anyType" minOccurs="0"/>
        <xsd:element name="boStatus" type="xsd:string" minOccurs="0"/>
        <xsd:element name="statusReason" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="bo" type="xsd:string" minOccurs="0"/>
        <xsd:element name="creationDateTime" type="ouafDateTime"
minOccurs="0"/>
        <xsd:element name="boStatusDateTime" type="ouafDateTime"
minOccurs="0"/>
        <xsd:element name="serviceProvider" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="serviceProviderExternalId" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="fromDateTime" type="ouafDateTime"
minOccurs="0"/>
        <xsd:element name="toDateTime" type="ouafDateTime"
minOccurs="0"/>
        <xsd:element name="timeZone" type="xsd:string" minOccurs="0"/>
        <xsd:element name="version" type="xsd:decimal" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:simpleType name="ouafDateTime">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="\d{4}-\d{2}-\d{2}-\d{2}\.\d{2}\.\d{2}"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

Java Code

The following methods are the main elements of the implementation of parsing files in XML format:

Method	Description
loadNextPortion()	This method gets called from getNextStream(). It navigates through entire incoming XML structure and calls processDevice() or processData() for every “device” or “initialMeasurementData” elements.
processDevice()	This method extracts the device related information and stores to be used later when Plain XML is generated.
processData()	This method navigates through entire content of “initialMeasurementData” element and generates Plain XML by using XMLEventWriter.

File Parser testing

The testing process of file parser functionality depends on the development environment that is used for particular implementation of a customer specific file format parser. The key points about FP testing logic are as follows:

- File parser is a Java class.
- It can be tested by running any java code for creating an instance of FP and invoking its methods.
- A “mock-up” class for simulating of some GFP logic that is involved in communication between GFP and FP should be implemented. See **FileProcessor Interface** on page 2-34 for more information.

OSB Configuration

The OSB Configuration consists of four projects. These projects can be categorized by either functionality or content:

Functionality by Incoming File Format

CSV format processing:

- SGG-DG-CSV-BASE
- SGG-DG-CSV-CM

XML format processing:

- SGG-DG-XML-BASE
- SGG-DG-XML-CM

Content (by purpose of content)

Content containing business logic implementation:

- SGG-DG-CSV-BASE
- SGG-DG-XML-BASE

Content containing configuration settings related to the objects and variables required during the processing of payloads:

- SGG-DG-CSV-CM
- SGG-DG-XML-CM

The following table describes the contents of each of these projects:

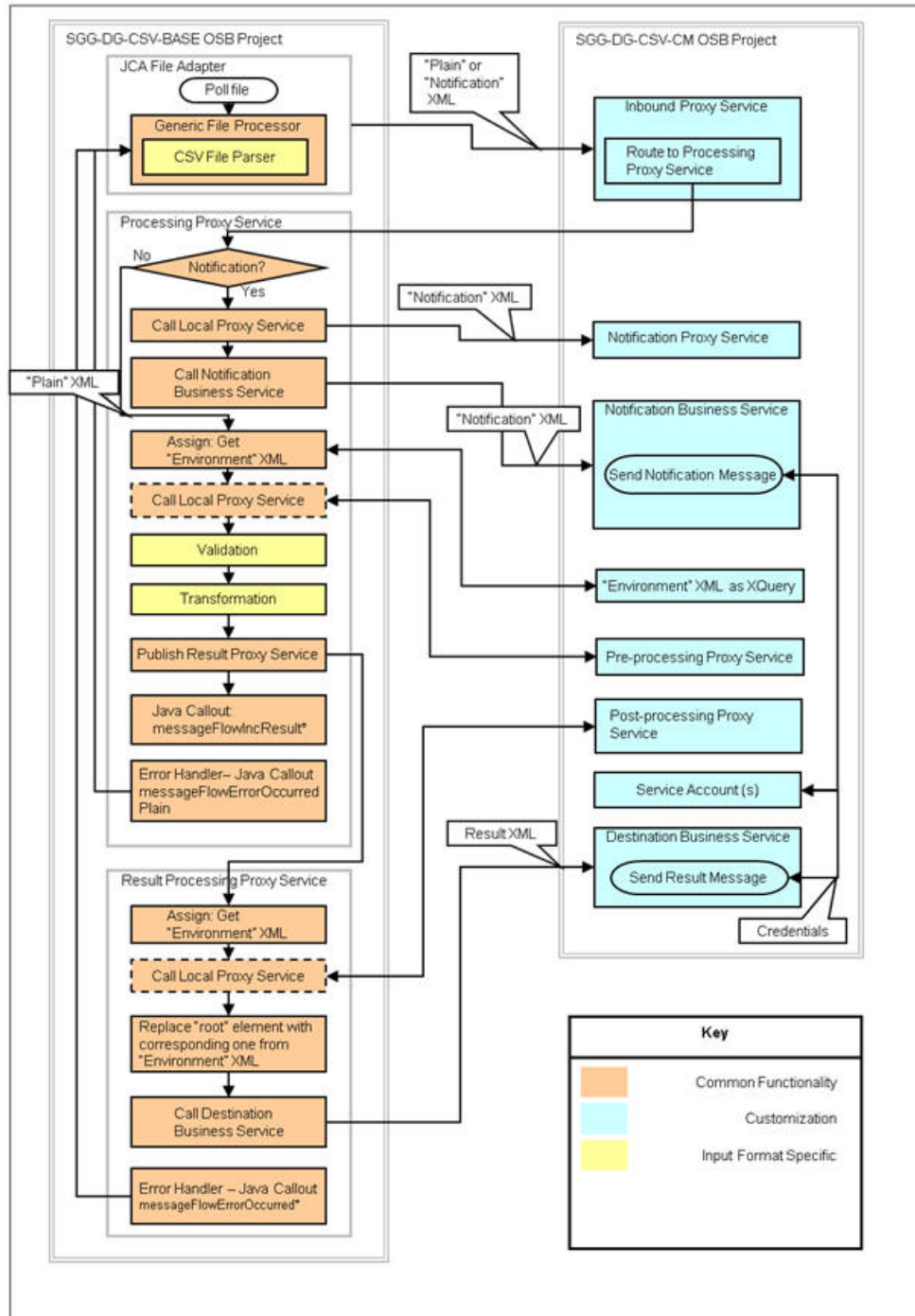
Project	Description
SGG-DG-CSV-BASE	<p>Contains the components responsible for “actual” processing of data coming in CSV format. It can be upgraded in future releases without affecting the customization and environment settings that are stored in the SGG-DG-CSV-CM project. This project performs the following functions:</p> <ul style="list-style-type: none"> • Processes notification messages that are sent from the GFP. • Calls the customizable local service proxies for pre- and post-processing of passed data • Validates passed data against the XSD schema for Plain XML format • Transforms passed Plain XML into D1-InitialLoadIMD or D1-DeviceEventSeeder formats according to type of incoming data. • Sends result structures to destination JMS queue. • Updates statistic information via calling the <code>messageFlowIncResultIMD</code> or <code>messageFlowIncResultEvent</code> method accordingly. • Handles errors that occur to provide information to the File Processing component. Error handling contains separate handlers for two stages: before and after the Plain XML is transformed to the final XML structure. The separation is necessary to make Java calls to appropriate methods (<code>messageFlowErrorOccurredPlain</code>, <code>messageFlowErrorOccurredIMD</code> or <code>messageFlowErrorOccurredEvent</code>) according to the current processing stage and data type.
SGG-DG-CSV-CM	<p>Contains environment related configuration settings. It allows the customization and simplifies the future upgrades as well (See Configuring the OSB Project on page 2-28 for example values).</p>

Project	Description
SGG-DG-XML-BASE	<p data-bbox="824 216 1523 373">Contains the components responsible for “actual” processing of data coming in XML format. It can be upgraded in future releases without affecting the customization and environment settings that are in SGG-DG-XML-CM project. This project performs the following functions:</p> <ul data-bbox="824 394 1523 1056" style="list-style-type: none"><li data-bbox="824 394 1523 422">• Processes notification messages that are sent from GFP.<li data-bbox="824 443 1523 499">• Calls the customizable local service proxies for pre- and post-processing of passed data.<li data-bbox="824 520 1523 548">• Validates passed data.<li data-bbox="824 569 1523 657">• Transform passed data. In current tutorial a simple XQuery code is used just to show the functionality available for transformation.<li data-bbox="824 678 1523 735">• Sends result structures to destination JMS queue in form of acceptable by the D1-InitialLoadIMD XAI inbound service.<li data-bbox="824 756 1523 812">• Updates statistic information via calling messageFlowIncResultIMD method.<li data-bbox="824 833 1523 1056">• Handles errors to provides information to GFP. Error handling contains separate handlers for two stages: before and after the Plain XML is transformed to the final XML structure. The separation is necessary to make a Java calls to appropriate methods (messageFlowErrorOccurred or messageFlowErrorOccurredIMD) according to the current processing stage.
SGG-DG-XML-CM	<p data-bbox="824 1077 1523 1134">Contains environment related configuration settings. It allows the customization and simplifies future upgrades as well.</p>

Implementation details

This section provides an overview of the implementation details. For more detailed information see the comments contained in the source code.

The following diagram shows the major processes and the data flow implemented in the OSB Configuration projects. The implementation description that follows is related to CSV parsing.



SGG-DG-CSV-BASE OSB Project

The SGG-DG-CSV-BASE OSB Project contains components that are not meant to be changed on customer site. They implement logic specific to the CSV format processing steps: validation, transformation and so on. Project consists of the following items in the related folders:

- JCA Bindings folder:
 - GenericFileProcessor.jca
 - GenericFileProcessor.wsdl
- JARs folder:
 - com.splwg.d1.sgg.osb.messageFlow.jar
- Proxy Services folder:
 - ProcessingProxyService
 - ResultProcessingProxyService
- XML Schemas folder:
 - CSV-Common.xsd
 - ProcessingNotifications.xsd
 - Result.xsd
- XQueries folder:
 - CSV2ResultXQuery.xq

JCA Bindings

A JCA file and WSDL are required to create a JCA Transport based proxy in OSB. The following files are generated by using JDeveloper. They can be used in all vendor specific projects as the final configuration is done in the “Inbound Proxy Service” by setting to false the “Always use configuration from JCA file” option on JCA Transport screen and overriding all necessary parameters. The WSDL is created in the vendor-agnostic way as well.

- **GenericFileProcessor.jca:** The GenericFileProcessor.jca file contains the JCA File Adapter properties that are described in the “A.1 Oracle File and FTP Adapters Properties” section of the “Oracle JCA Adapter Properties” book in the *Oracle Fusion Middleware User's Guide for Technology Adapters 11g Release 1*. All properties can be overridden in the InboundProxyService JCA Transport based proxy service that is located in the SGG-DG-CSV-CM where this JCA file is used. See the source code for details.
- **GenericFileProcessor.wsdl:** The interface to the JCA File Adapter service is described through a WSDL file. See source code for details.

JARs

com.splwg.d1.sgg.osb.messageFlow: The com.splwg.d1.sgg.osb.messageFlow jar contains the com.splwg.d1.sgg.osb.messageFlow.FileProcessorUtil class. The FileProcessorUtil class provides access from the Service Proxy Message Flow to the static methods of GenericFileProcessor class via the following methods that are called from the proxy services message flows and corresponding error handles (see message flow logic descriptions and source code for details when those methods are used):

Method	Description
public static void messageFlowErrorOccurred Plain(XmlObject fault)	Notifies GFP that error occurred in OSB message flow before the Plain XML is transformed to IMD or Device Event structure.

Method	Description
public static void messageFlowErrorOccurred Event(XmlObject fault)	Notifies GFP that error occurred in OSB message flow after the Plain XML is transformed to Device Event structure.
public static void messageFlowErrorOccurred IMD(XmlObject fault)	Notifies GFP that error occurred in OSB message flow after the Plain XML is transformed to IMD structure.
public static void messageFlowIncResultEvent (int count)	Notifies GFP that one or more Device Event structures are successfully sent to JMS queue.
public static void messageFlowIncResultIMD(int count)	Notifies GFP that one or more IMD structures are successfully sent to JMS queue.

Proxy Services

ProcessingProxyService Proxy Service: The ProcessingProxyService is a local transport based proxy service. It contains the message flow where the validation and transformation of Plain XML that was generated for usage and device event data in the CSV format are implemented. The publishing of notification messages is implemented in ProcessingproxyService as well.

Message Flow: The ProcessingProxyService message flow consists of the following actions that are responsible for the processing of the Usage and Event data coming as Plain XML for incoming data in CSV format. The result of this processing will be one of the following:

- XML containing an Initial IMD structure according to the IMD Seeder BO's schema (see the Initial Measurement Data XML Format section in Oracle Utilities Smart Grid Gateway Configuration Guide)
- Device Event Seeder XML (see the Device Event XML Format section in Oracle Utilities Smart Grid Gateway Configuration Guide)

The publishing of notification messages is also implemented in this flow.

The message flow logic is as follows:

- If fn:count(\$body/ga:Payload) is equal to '0' (the current message is a "Notification" message)
 - "Publish to the SGG-DG-CSV-CM/Proxy Services/NotificationProxyService local proxy service.
 - Publish to the SGG-DG-CSV-CM/Proxy Services/NotificationBusinessService business service.
- Else
 - Assign to the envSettings variable the content of the SGG-DG-CSV-CM/ XQueries/ EnvironmentSettings resource. It contains the run-time configuration settings (see EnvironmentSettings.xq file below)
 - If \$envSettings/callPreProcessing is equal to 'true'
 - Call the PreProcessingProxyService service proxy from SGG-DG-CSV-CM project passing the \$body variable as an input and store the response back to the \$body variable
- Validate \$body variable against XML schema for Plain format
- Transform the Plain XML into a result element with children elements InitialLoadIMD or DeviceEventSeeder according to the name Payload/* node and assign result to the \$curResult variable

- If name of Payload/* node is equal to 'Usage' (interval usage)
 - Generate a result/InitialLoadIMD structure according to the following mapping table:

Plain XML Element	Initial Load IMD Element	Notes
RecordType	N/A	
DeviceId	dvcIdN	
UOM	externalUOM	
IntervalDuration	spi	
StartDateTime	stDt	Convert to XSD "dateTime" type
EndDateTime	enDt	Convert to XSD "dateTime" type
Intervals/		
Interval/		
Value [Status]	msrs	Value ->
msrs/mL/q		
Status ->		
msrs/mL/sts/stsL/ st		
N/A	imdType	Const value 'D11L'
Origin	externalId	
ServProvExtRefId	serviceProviderExternal Id	
RawData	rawData	When \$envSettings/ populateRawIMD/ text()='true'

- If name of Payload/* node is equal to 'Event' (device event data)
 - Generate a result/DeviceEventSeeder structure according to the following mapping table:

Plain XML Element	Initial Load IMD Element	Note
RecordType	N/A	
DateTime	eventDateTime	Convert to XSD "dateTime" type
DeviceId	externalUOM	
Name	externalEventName	
Origin	externalSourceIdentifier	

Plain XML Element	Initial Load IMD Element	Note
ServProvExtRefId	externalSenderId	
RawData	rawEventInformation	When \$envSettings/ populateRaw/ text()='true'

- Call SGG-DG-CSV-BASE/Proxy Services/ResultProcessingProxyService proxy service passing \$curResult variable

Error Handling: The GFP will be called in a proxy service level error handler and the “fault” context variable is provided as well. The last action in error handler is “Replay with Success”. It will allow processing the rest of incoming file.

ResultProcessingProxyService Proxy Service: The Result Processing Proxy Service will contain actions where post-transformation logic will be implemented. Also, it will have an Error Handler responsible only for catching errors occurred after a Plain XML is transformed to the result transaction. For example, such errors could be related to incorrect configuration of JMS queues in business service.

Message Flow: The message flow consists of the following actions that are responsible for the processing of the IMD data that is received from ProcessingProxyService service. The message flow logic is as follows:

- Assign to the envSettings variable the content of the SGG-DG-CSV-CM/ XQueries/ EnvironmentSettings resource. It contains the run-time configuration settings.
- Assign “true” to the isIMD variable if the name of \$body/result/*[1] node is “InitialLoadIMD” or “false” otherwise.
- If \$envSettings/callPostProcessing is equal to 'true'
 - Call the SGG-DG-CSV-CM/Proxy Services/PostProcessingProxyService local proxy service passing \$body and assign response to \$body variable
- Assign to \$finalResult an XML containing only root element with name taken from \$envSettings variable according to the following table. Set root element's dateTimeTagFormat attribute to the “xsd” value.

\$envSettings Element	Condition
destinationRootElementIMD	\$isIMD is “true”
destinationRootElementEvent	\$isIMD is not “true”

- Insert all child nodes of \$body/result into \$finalResult variable as child nodes.
- Call SGG-DG-CSV-CM/Business Services/DestinationBusinessService business service passing \$finalResult variable
- If \$isIMD is “true”
 - Invoke messageFlowIncResultIMD Java method passing value “1”.
- Else
 - Invoke messageFlowIncResultEvent Java method passing value “1”.

Error Handling: The GFP will be called in a proxy service level error handler and the “fault” context variable is provided as well. The last action in error handler is “Replay with Success”.

The error handling logic is as follows:

- If \$isIMD is “true”
 - Invoke messageFlowErrorOccurredIMD Java method passing \$fault variable
- Else
 - Invoke messageFlowErrorOccurredEvent Java method passing \$fault variable
- Reply with Success

XML Schemas

CSV-Common.xsd: The CSV-Common.xsd XML schema defines the structure of message in Plain XML format that JCA File Adapter passes to the following message flow when incoming file is in the CSV format. See the source code for details.

Result.xsd: The Result.xsd schema describes the structure of message that is produced first time by a transformation action in the ProcessingProxyService and passed to ResultProcessingProxyService. See the source code for details.

ProcessingNotifications.xsd: The ProcessingNotifications.xsd XML schema describes the structure of message that JCA File Adapter passes to the following message flow to notify the OUAF about current stage of input file processing. See the Upload Statistics subsection of the Understanding Device Communication section in the Oracle Utilities Smart Grid Gateway Configuration Guide for details.

XQueries

CSV2ResultXQuery.xq: The CSV2ResultXQuery.xq XQuery is used to transform incoming message in the Plain XML to the result structure described by Result.xsd schema. See mapping tables in the ProcessingproxyService proxy service's message flow description and source code for details.

SGG-DG-CSV-CM OSB Project

The SGG-DG-CSV-CM OSB project contains components meant to be changed on customer site. Such separation allows future upgrades of base functionality implemented in SGG-DG-CSV-BASE project without touching the customization and environment settings.

Project consists of the following items in the related folders:

- Business Services folder:
 - DestinationBusinessService
 - NotificationBusinessService
- Proxy Services folder:
 - InboundProxyService
 - InboundProxyService.wsdl
 - NotificationProxyService
 - PreProcessingProxyService
 - PostProcessingProxyService
- Service Accounts folder:
 - DestinationServiceAccount
- XQueries folder:
 - EnvironmentSettings.xq

Business Services

Destination Business Service: The DestinationBusinessService is a JMS transport based business service containing all settings about destination queue where a device event seeder XML will be sent. Those settings can be changed on customer site according to the deployment environment.

Notification Business Service: The NotificationBusinessService is a JMS transport based business service containing all settings about destination queue where the notification messages will be sent. Those settings can be changed on customer site according to the deployment environment.

Proxy Services

Inbound Proxy Service: The InboundProxyService is a JCA-transport based proxy service. It receives vendor-specific messages in Plain XML format from JCA File Adapter and routes them to the ProcessingProxyService. The proxy service is generated from SGG-DG-CSV-BASE/JCA Bindings/GenericFileProcessor.jca file.

Configuration Details

The following table contains regular JCA Transport properties along with the Dynamic Endpoint Properties that are related only to SGG Adapter functionality.

Property	Value
JCA File	SGG-DG-CSV-BASE/JCA Bindings/ GenericFileProcesso
Adapter Name	File Adapter
Adapter Type	FILE
Dispatch Policy	default
JNDI Service Account	Empty
Endpoint Properties	Empty
Dynamic Endpoint Properties:	
SGG_INPUT_PARSER	com.splwg.dg.osb.csv.FileParserCSV
SGG_ERROR_FOLDER	ErrorDirectoryPath
SGG_SP_EXT_REF_ID	ServiceProviderExternalReferenceId

Where:

- SGG_INPUT_PARSER - the name of class where the File Parser functionality is implemented
- SGG_ERROR_FOLDER - the directory where in case of error will be created files with the portion of raw data that is read when the error occurred. Also RFD files will be created here as well.
- SGG_SP_EXT_REF_ID - the value that will be sent to OUAF as a Service Provider External Reference ID for Initial Measurement Data or a External Sender ID for Device Event Data.

The following table contains the common advanced settings for Oracle JCA File Adapter properties. All Activation Spec Properties except the PipelineValves value are provided as a sample

and can be changed on customer site. For details please, see *Oracle Fusion Middleware User's Guide for Technology Adapters 11g Release 1 (11.1.1)* for Oracle JCA Adapter for Files/FTP.

Property	Value
Always use configuration from JCA File	Unchecked
Operation Name	Read
Activation Spec Properties:	
MaxRaiseSize	5
PollingFrequency	30
UseHeaders	False
DeleteFile	True
PhysicalArchiveDirectory	PhysicalArchiveDirectoryPath
Requursive	False
IncludeFiles	.**.csv
ThreadCount	10
PublishSize	1
PhysicalDirectory	PhysicalDirectoryPath
PipelineValves	com.splwg.d1.sgg.osb.common.GenericFileProcessor
MinimumAge	0

Message Flow: As InboundProxyService is meant just to route incoming messages to the ProcessingProxyService its message flow consists of a RouteNode containing single Routing action.

Pre-processing Proxy Service: The PreProcessingProxyService is a local-transport based proxy service that meant to allow the customizable pre-processing of Plain XML like some value changes the result of which can be used in post-processing. It is called by ProcessingProxyService right at the beginning of message flow.

Message Flow: The message flow is left empty. The messages will be echoed from the request pipeline to the response pipeline until required customization will be done by adding extra nodes.

Post-processing Proxy Service: The PostProcessingProxyService is a local-transport based proxy service that meant to allow the customizable post-processing of Device Event Seeder structure. It is called from ProcessingProxyService at the end of message flow just before sending a result structure to the DestinationBusinessService.

Message Flow: The message flow is left empty. The messages will be echoed from the request pipeline to the response pipeline until required customization will be done by adding extra nodes.

Notification Proxy Service: The NotificationProxyService is a local-transport based proxy service that meant to allow the customizable publishing of notification messages. It is called by ProcessingProxyService right at the beginning of message flow.

Message Flow: The message flow is left empty. Some action like a sending email can be added during customization.

Service Account(s)

Destination Service Account: The DestinationServiceAccount contains credentials to access the destination JMS Queues. The User Name and Password should be changed during deployment.

XQueries

EnvironmentSettings.xq: The EnvironmentSettings.xq xquery contains the runtime configuration settings. It allows an adjustment of result XML processing like to populate it optionally with “raw” data or specify an XAI inbound service names. It contains the following schemes:

Element Name	XML Data Type	Comments
populateRaw	boolean	Enables adding “raw” data into “result” XML.
callPreProcessing	boolean	Enables calling to pre-processing proxy service.
callPostProcessing	boolean	Enables calling to post-processing proxy service.
destinationRootElement IMD	string	Holds the name of XAI inbound service for interval IMD seeder.
destinationRootElement Event	string	Holds the name of XAI inbound service for Device Event seeder

Setting Up the Adapter Environment

This section contains detailed information about the setting up of an environment with an example adapter, including:

- **Setting Up the Java Project**
- **Configuring the OSB Project**

Setting Up the Java Project

1. Open the sgg-osb-generic-adapter.zip file and extract the FileParser folder into a temporary location on your hard disk.
2. Create a Java project in JDeveloper and import source code into it from the folder extracted previously.
3. Add the spl-d1-osb-2.0.1.jar file to the project's Libraries and Classpath properties.
4. Deploy the project as a jar file to the destination OSB domain's lib folder.

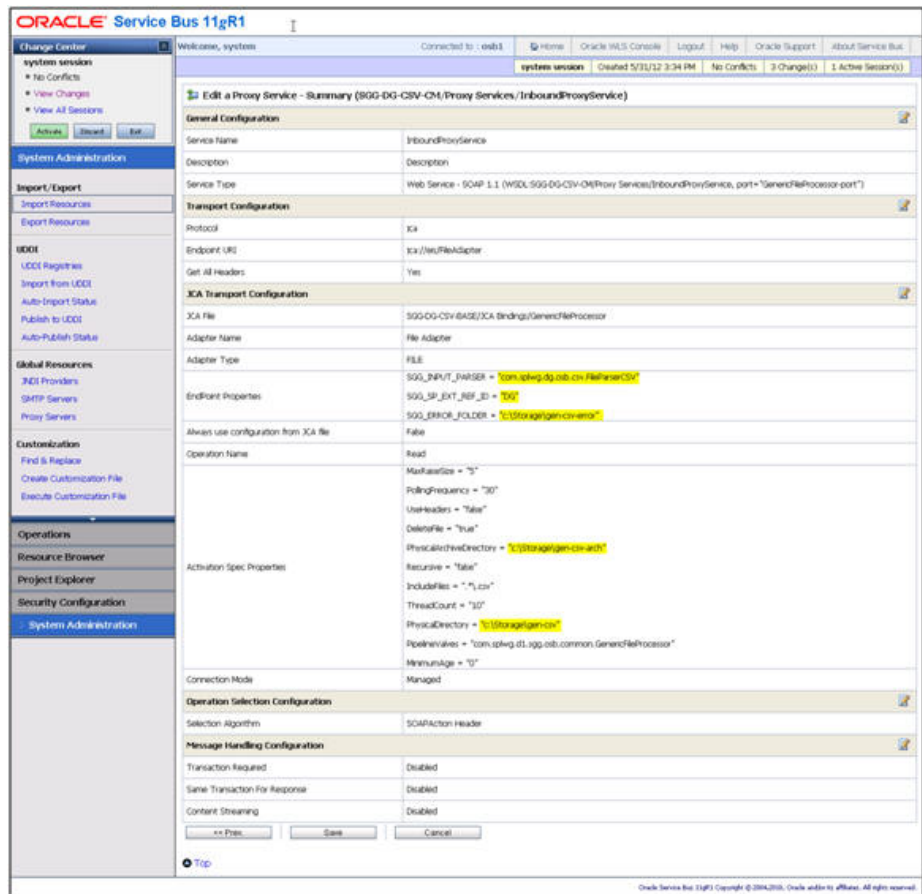
Note: The WebLogic server should be restarted after jar file is copied to lib folder. For more details, see *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*.

Configuring the OSB Project

1. Open the sgg-osb-generic-adapter.zip file and extract the com.splwg.dg.sgg.osb.configuration.jar file into a temporary location on your hard disk.
2. Ensure that OSB is running in the domain that is created for the adapter.
3. Import resources by using the OSB Administration Console via the **System Administration > Import/Export > Import Resources** menu item from the extracted

com.splwg.dg.sgg.osb.configuration.jar file. Note that conflicts may occur during import. They will be fixed in the following steps.

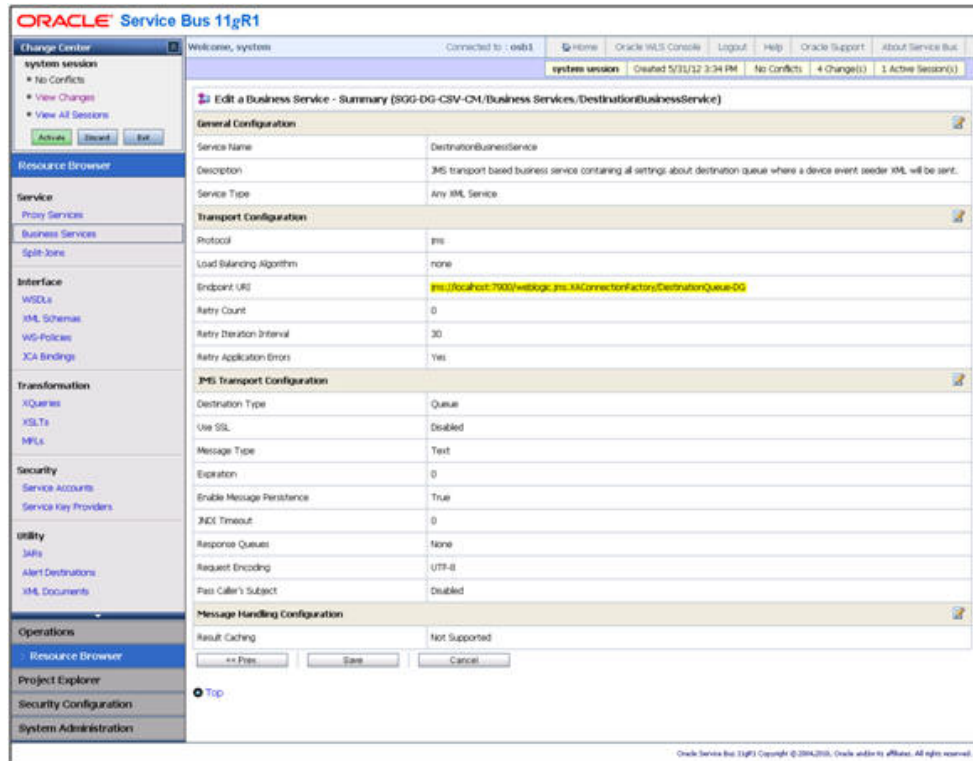
4. Check and change the JCA Transport Configuration properties of the SGG-DG-CSV-CM/Proxy Services/InboundProxyService proxy service according to the deployment environment:



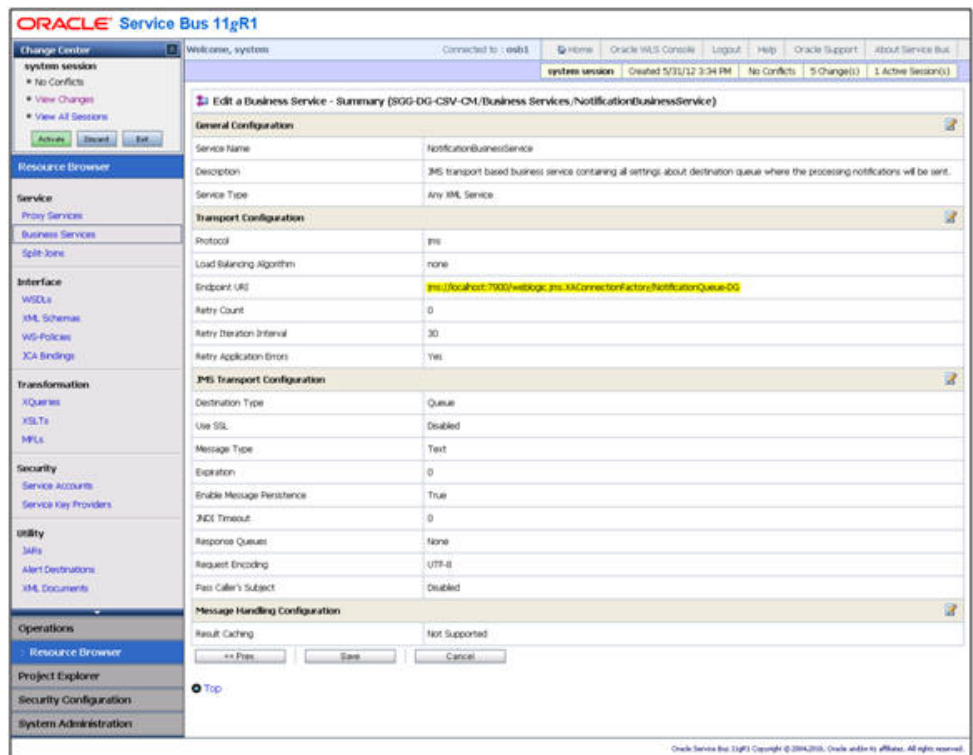
Where:

- **Endpoint Properties/SGG_INPUT_PARSER** is a name of class that implements file parsing functionality. See a Java project mentioned early.
 - **Endpoint Properties/SGG_ERROR_FOLDER** is a folder where a rejected transaction from incoming file will be placed in case when the parsing or validation is failed.
 - **Endpoint Properties/SGG_SP_EXT_REF_ID** is a value corresponding to the target Service Provider's external reference id on OUAF side. It is used in the XML structures related to the IMD & Event Upload Stats functionality. Also, this value will be placed into D1-InitialLoadIMD/serviceProviderExternalId and D1-DeviceEventSeeder/externalSenderId elements.
 - **Activation Spec Properties/IncludeFiles** is the naming convention that the Oracle File Adapter uses to poll for inbound files.
 - **Activation Spec Properties/PhysicalArchiveDirectory** is a folder where to archive successfully processed files.
 - **Activation Spec Properties/PhysicalDirectory** is an input folder to be polled.
5. Change the Endpoint URI in the Transport Configuration section of the SGG-DG-CSV-CM/Business Services/DestinationBusinessService business service according to the deployment environment with value containing URI for JMS server and queue where IMD or

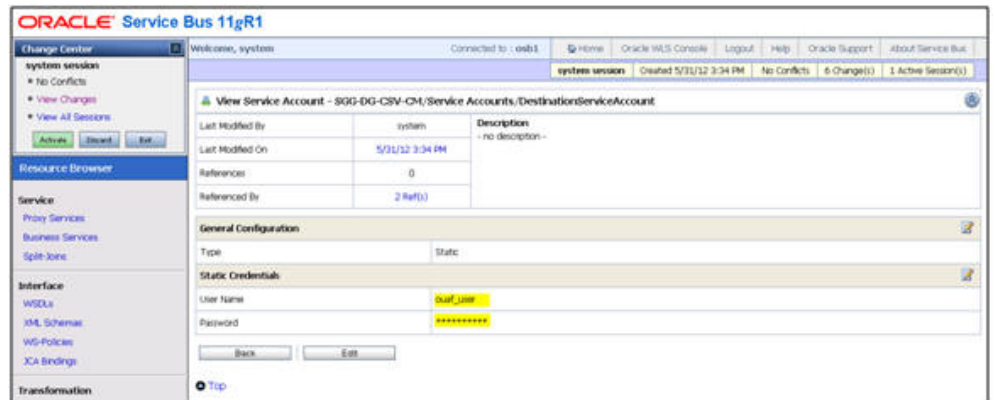
Device Event structures will be sent (see *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* for more detail):



- Change the Endpoint URI in the Transport Configuration section of the SGG-DG-CSV-CM/Business Services/NotificationBusinessService business service according to the deployment environment:



- Update the credentials stored in the SGG-DG-CSV-CM/Service Accounts/ DestinationServiceAccount service account according to the deployment environment with user ID and password required to access JMS server:



- Repeat steps 4-7 for the SGG-DG-XML-CM project

Additional Information

This section contains additional information about the adapter development kit, including:

- **Processing Large Input Files**
- **FileParser Interface**
- **FileParser2 Interface**
- **FileProcessor Interface**
- **Generating Java Classes from XML Schemas with JAXB**

Processing Large Input Files

In some environments, the OSB project may begin processing a large input file before it has been completely copied to the input directory. To prevent this, configure the `MinimumAge` property in the “`InboundProxyService`” proxy service. The `MinimumAge` property specifies the minimum age of files to be retrieved, based on the last modified time stamp. This enables large files to be completely copied to the input directory before they are retrieved for processing.

See Working with Adapter OSB Projects in the *Oracle Utilities Smart Grid Gateway Configuration Guide* for more information about setting the `MinimumAge` property.

FileParser Interface

This section provides the source code of the `FileParser` interface used by the included in the Java class delivered with the Java package described earlier in this section.

```
package com.splwg.d1.sgg.osb.common;

import java.io.IOException;
import java.io.InputStream;

/** <p>
 * FileParser component is responsible for processing the input
 * stream and returning a Plain XML containing all data from
 * incoming transaction in a shape expected by the following
```

```
* Oracle Service Bus message flow.
* </p>
*/
public interface FileParser {
    /**
     * Returns the next message in Plain XML format
     * or null if there are no more messages
     * @return next message as InputStream
     */
    InputStream getNextStream() throws IOException;

    /**
     * Returns the current position in the currently
     * parsed source input stream
     * @return the position in source input stream after
     * last incoming row was read and parsed
     */
    long getPosition();

    /**
     * Sets source stream
     * @param input the Input Stream that should be parsed
     */
    void setInputStream(InputStream input);

    /**
     * Sets "owner" FileProcessor
     * @param owner the Instance of FileProcessor that will
     * be notified about current processing via call to
     * utility methods (saveInvalidRow and setParseFinishNote)
     */
    void setProcessor(FileProcessor owner);

    /**
     * Sets the position from which the parsing of source
     * stream should be started.
     * It is necessary if last time the processing of
     * current file has been interrupted
     * @param position the starting position
     */
    void setStartPosition(long position);

    /**
     * Returns the byte array that contains a portion of an incoming
     * file that most recently has been read and converted to Plain
XML.
     * It should be in the same format as incoming file. The header and
     * tail records, if there are, should be included and updated
according
     * to the content
     */
    byte[] getCurrentInputPortion();

    /**
     * Returns type of currently generated transaction - USAGE or EVENT
     */
    FileProcessor.TransactionType getCurrentTransactionType();

    /**
     * Returns the number of successfully generated transactions of
given type
     */
}
```

```
int getTransactionCount(FileProcessor.TransactionType type);  
  
/**  
 * Returns the number of expected transactions in input stream  
 * Returns -1 if information not available  
 */  
int getTotalExpected();  
}
```

FileParser2 Interface

This section provides the source code of the FileParser2 interface used by the Java class delivered with the Java package described earlier in this section. This interface has been added to include a way for the File Processor to retrieve the file creation date which can be stored inside the incoming file.

```
package com.splwg.d1.sgg.osb.common;

import java.util.Date;

public interface FileParser2 extends FileParser {
    /**
     * FileProcessor usage method
     * Returns the File creation date and time stamp stored
     * in the file
     * Returns null if information not available
     */
    Date getCreationDateTime();
}
```

FileProcessor Interface

This section contains a portion of the FileProcessor interface related to the file parsing functionality.

```
package com.splwg.d1.sgg.osb.common;
public interface FileProcessor {
    public enum TransactionType {
        USAGE, EVENT
    }

    /**
     * FileParser usage method
     *
     * @return a string containing the name of currently processed
inbound file
     */
    String getOrigin();

    /**
     * FileParser usage method Creates a file in the Error Folder and
saves
     * passed row parameter to it
     *
     * @param position
     *         the number representing the position in file where
invalid
     *         raw data is started
     * @param row
     *         the byte array containing an invalid raw data in
vendor
     *         specific format the same with format of incoming file
     * @param failCause
     *         the string showing the cause of fail It will be used
while
     *         logging the fail message.
     */
    void saveInvalidRow(long position, byte[] row, String failCause);
}
```

```
/**
 * FileParser usage method
 * Returns the current offset in input file
 */
long getCurrentInputPosition();
}
```

Generating Java Classes from XML Schemas with JAXB

In JDeveloper you can use JAXB (Java Architecture for XML Binding) to generate Java classes from XML schemas. JAXB is an easy way to incorporate XML data and processing functions in Java applications without having to know XML. You can generate a JAXB 1.0 or 2.0 content model, including the necessary annotations, from an XML schema.

When the JAXB binding compiler is run against an XML schema, JAXB packages, classes, and interfaces are generated. You can then use the generated JAXB packages and the JAXB utility packages in a binding framework to unmarshal, marshal, and validate XML content.

To generate Java classes from XML schemas with JAXB:

1. From the main menu choose **File > New > Business Tier > TopLink/JPA** and select either JAXB 1.0 or 2.0 Content Model from XML Schema to open the compilation dialog.
2. Select the schema file and optionally the JAXB customization file to use and the package to which the generated classes will be added.

The JAXB package and generated classes are added to the Application Resources folder.

OSB Processing Terms and Acronyms

Term	Definition
AMI	Advanced Metering Infrastructure
CSV	Comma-Separated Values
FA	File Adapter. Oracle JCA File Adapter.
FP	File Parser, Java code that parses input file and generates Plain XML
GA	Smart Grid Gateway generic adapter
GFP	Generic File Processor, Java code that is invoked by FA. It communicates with FP
IMD	Initial Measurement Data
IMD XML	The final XML containing the IMD.
JAR	Java Archive
JAXB	Java Architecture for XML Binding
JCA	Java Connector Architecture
JDeveloper	Freeware Oracle integrated development environment for development of Java-based SOA and Java EE applications
JMS	Java Message Service
MDB	Message Driven Bean
SMDF	Service and Measurement Data Foundation
OSB	Oracle Service Bus
OOUAF	Oracle Utilities Application Framework
Plain XML	An intermediate XML containing all of the values from the inbound file. It is necessary because OSB Message Flow can handle only XML.
PPS	Processing Proxy Service, OSB proxy service where the validation and transformation of Plain XML is implemented. It is necessary to catch errors that occurred before Plain XML is transformed to IMD or Device Event seeder structures.
RFD	Rejected File Descriptor, File containing information about a portion of input file that was a cause of error.
RPPS	Result Processing Proxy Service, OSB proxy service where processing of transformation result is implemented. It is necessary to catch errors that occurred after Plain XML is transformed to IMD or Device Event seeder structures.
SGG	Smart Grid Gateway
Weblogic	Oracle J2EE Application Server
WSD	Web Session Directory
XML	eXtensible Markup Language
XPath	Programming language for selecting nodes from an XML document
XQuery	Programming language to query XML.

Chapter 3

Configuring BPEL Processing for Two-Way Commands

This section describes the BPEL processes included in the Oracle Utilities Smart Grid Gateway Adapter Development Kit that support two-way communication between the adapter and the head-end system. The ideas and general practices described in this section can be applied to any other adapter. This section includes the following topics:

- **WSDLs, Endpoints, and XAI**
- **Configuring and Customizing the Adapter BPEL Processes**
- **Components of the Composites**
- **Commands**
- **Working with Enterprise Manager**
- **MultiSpeak Implementation**
- **The Test Harness**

WSDLs, Endpoints, and XAI

WSDLs are files that describe a web service. New services must utilize the WSDLs to determine the structure of requests and responses. For remote services, they contain the locations to access the services. For both remote and local (hosted) services, the files will contain the definitions for each web service including names, arguments, exceptions, and the structure of the input and output messages.

In Oracle SOA, WSDLs can be classified as a “service” or as a “reference.” Services are hosted on the SOA server; that is, the implementation of the web service is on the local application server. Referenced services are implemented on a different server. This distinction is also relevant to the composite level. Services are implemented in the current composite. References are located elsewhere, possibly on the same app server, but in a different composite.

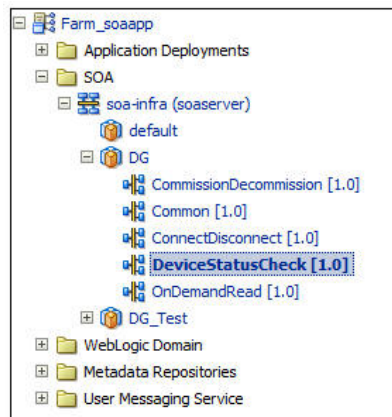
An “endpoint” is simply the URL for a web service. Since web services communicate via HTTP, each will have a unique URL. Once a service’s endpoint is known, an XAI service can be targeted to it.

How to locate WSDLs and Endpoints

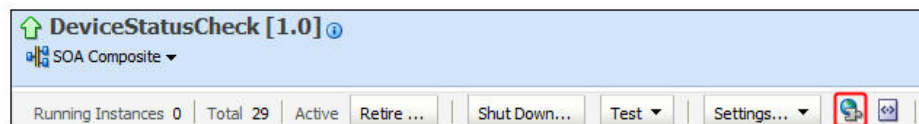
There are two ways to locate WSDLs in an installed Adapter: through Enterprise Manager and by using a direct URL. Only “hosted” WSDLs can be located in this way. Referenced (remote) WSDLs must be either located in source code or obtained from the hosted location.

How to Use Enterprise Manager to Locate WSDLs and Endpoints

1. Open Enterprise Manager and use the navigation pane to open the Dashboard of the desired composite:



2. The top bar of the Dashboard contains several buttons and icons. One of these is a “world” icon with a puzzle piece over it. Click this icon to display the WSDL and endpoint URIs for the composite:

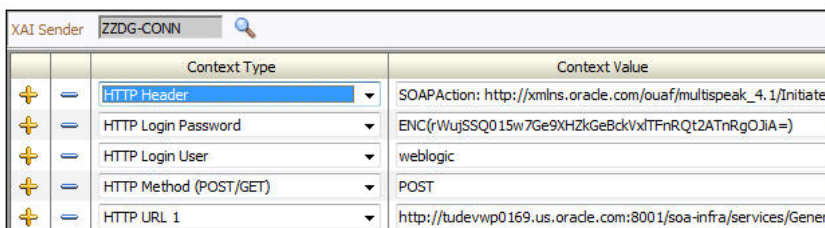


- Click the URL link to see the WSDL in the browser, or right click and save it to your machine for use in developing new services.

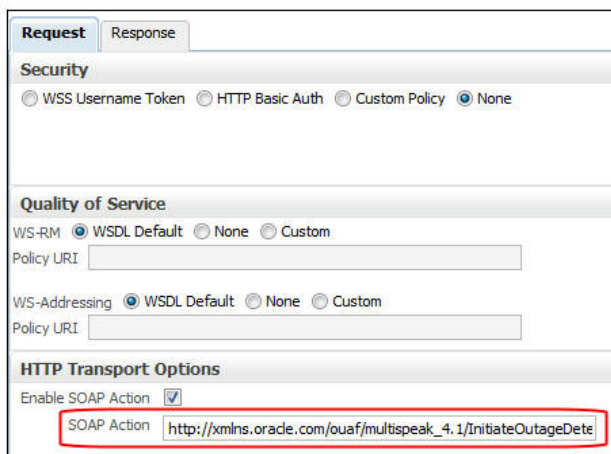


Depending on the requirements, It may be necessary to download the associated schema or WSDL file(s). Schemas are available within the WSDL's "types" element. Find associated WSDL URLs within the import element. That URL can be pasted into a browser tab.

The endpoint for the service is also visible in this window. It is this URL that should be added to XAI Sender configuration. For example:



The value HTTP Header/SOA Action is taken from the service's WSDL. Locate the `wsdl:definitions/wsdl:binding/wsdl:operation/soap:operation` element for the web method being invoked. An attribute called `soapAction` will contain the value for this field. A shortcut to this field is using Enterprise Manager's Testing framework (locate the Test tab from the dashboard view for the composite). Once the WSDL is parsed, the SOAP Action will appear in a field on the request tab:



The values **HTTP Login User** and **HTTP Login Password** should be set to a valid WebLogic user that has access to the module. The **HTTP Method** should always be set to POST and the HTTP URL 1 should be set to the value of the endpoint above.

How to Use a Direct URL to Locate WSDLs and Endpoints

The WSDLs can be accessed without Enterprise Manager by understanding the paths used on the SOA server. In general, they have the form:

```
http://{server name}:{port number}/soa-infra/services/{partition}/
{Composite}/{Web Service}
```

Components of the Composites

This section illustrates several important logical features of the BPEL composites, including:

- **Composite Properties**
- **Proxies**
- **Process Callouts**

Composite Properties

Most composites contain properties within the main file, composite.xml. They are essentially global constants that can be preconfigured with default values and changed at deployment time. They can be accessed post-deployment in Enterprise Manager.

Typically, these properties represent timeouts and boolean properties for activating or deactivating functionality. Other uses might include setting default values of constants such as URLs. They are accessed with the `ora:getPreference()` BPEL function.

They can be changed during development, at deployment time, or after deployment using different techniques for each.

Development Changes

At development time, properties can be created at will within the “component” sections of the composite.xml file. Here is an example of the properties found in DeviceStatusCheck.

```
<component name="DeviceStatusCheck">
  <implementation.bpel src="DeviceStatusCheck.bpel"/>
  <property name="bpel.preference.isExecutingInitODRequestReceivedCallout">true</property>
  <property name="bpel.preference.isExecutingInitODEventRequestCompleted">true</property>
  <property name="bpel.preference.isExecutingODEventNotificationArrived">true</property>
  <property name="bpel.preference.timeout.callback.years">0</property>
  <property name="bpel.preference.timeout.callback.months">0</property>
  <property name="bpel.preference.timeout.callback.days">0</property>
  <property name="bpel.preference.timeout.callback.hours">0</property>
  <property name="bpel.preference.timeout.callback.minutes">0</property>
  <property name="bpel.preference.timeout.callback.seconds">45</property>
  <property name="bpel.preference.OD_ServerCallbackEndpoint">http://127.0.0.1:8001/soa-infra/ser
  <property name="bpel.preference.externalSenderID">DSC Sensus</property>
  <property name="partnerLink.OD_ServerProxy.idempotent" type="xs:string"
    many="false">false</property>
</component>
```

They must be prefixed with “bpel.preference.” The BPEL process associated with them (see the `implementation.bpel` element) can use the `ora:getPreference()` method to extract the value. When accessing the values, the “bpel.preference” prefix is dropped.

Pre-Deployment Changes

While the values of properties cannot be changed at runtime, they can be altered using the configuration plan each composite uses for deployment. Most configuration plans will contain

multiple property elements within the “component” section. Modifying the value in the replace element and redeploying the composite will change the behavior of the property:

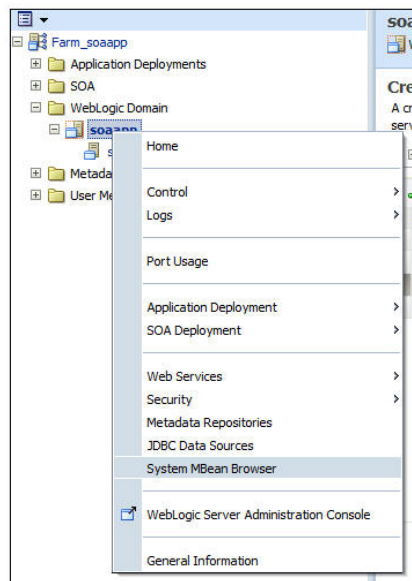
```
<component name="DeviceStatusCheck">
  <property name="bpel.preference.isExecutingInitODRequestReceivedCallout">
    <replace>true</replace>
  </property>
  <property name="bpel.preference.isExecutingInitODEventRequestCompleted">
    <replace>true</replace>
  </property>
  <property name="bpel.preference.isExecutingODEventNotificationArrived">
    <replace>true</replace>
  </property>
  <property name="bpel.preference.timeout.callback.years">
    <replace>0</replace>
  </property>
  <property name="bpel.preference.timeout.callback.months">
    <replace>0</replace>
  </property>
  <property name="bpel.preference.timeout.callback.days">
    <replace>0</replace>
  </property>
</component>
```

When the composite is deployed, the properties will contain the values in the “replace” elements.

Post-Deployment Changes

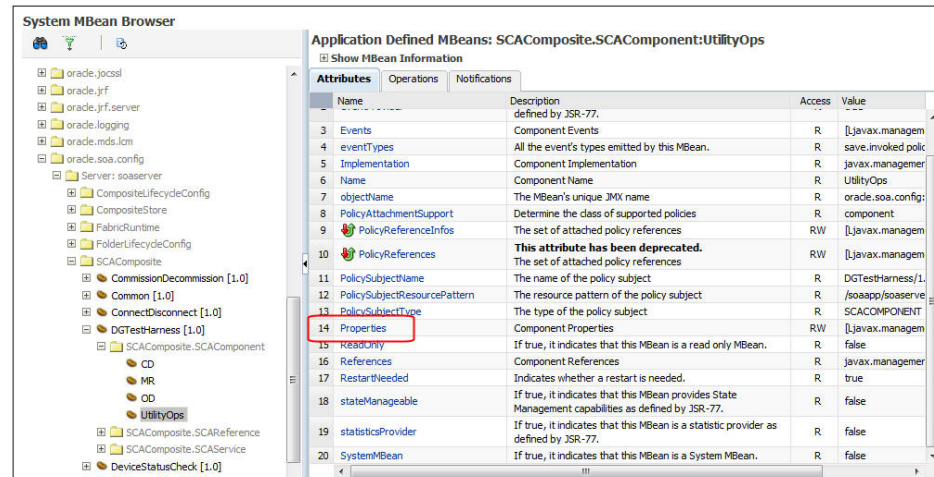
After deployment, changes can still be made to the values of the properties through Enterprise Manager.

1. In the Enterprise Manager navigation window, open WebLogic Domain/{domain name}. Right click on the domain and select **System MBean Browser**.



2. A new navigation pane opens in the window on the right. Select **Application Defined MBeans/oracle.soa.config/Server: {domain name}/SCAComposite/{Composite} [1.0]/SCAComposite.SCAComponent/{Component}** to access the controls for a

particular component. A “component” is a single BPEL or Mediator within a larger SOA composite



A list of options will open in the right pane.

3. Select **Properties**.

The Property sheet shows a list of all the properties. Opening one allows editing of the property value.

4. Click **Apply** to save any changes.

Proxies

Proxy web services make no changes to the data going through the system and are just used to centralize endpoint configuration. The message structures are identical to those of the head-end system. Data is passed through without modification. Mediator objects within the Common composite are used to move data from Oracle Utilities Smart Grid Gateway to the head-end system.

Process Callouts

Some services are designated as “Process Callouts.” These services allow integrators direct access to the XML coming to and from the head-end system. They are useful in the event the integrator needs to provide custom enrichment or transformation of the inbound or outbound data. Because process callouts are generally located before and after calls to the head-end system or when data arrives, they can also serve as an “event” trigger if the integrator requires additional processing logic. The services define identical inputs and outputs and are based on head-end schema definitions. To use a callout, the integrator should create a web service implementing one or more of the WSDLs defined in the Common composite. Other composites should then be configured to target the new Callout-type service.

A typical example is for re-rendering the meter identifier being sent to the head-end system. If the standard meter ID should be modified, a process callout service is a candidate for making this change. Note that because the schema is based off the head-end system's definition, deviations from the schema are not allowed.

Configuring and Customizing the Adapter BPEL Processes

This section outlines how to configure and customize the Oracle Business Process Execution Language (BPEL) processes provided with the Adapter Development Kit to work with your implementation, including:

- **Editing Configuration Files**
- **Customizing the BPEL Processes**

Editing Configuration Files

This section outlines the changes that need to be made in various configuration and build files to enable the application servers used in your implementation to communicate with each other, and to support the specifics of your head-end system. These files are also used in packaging and deploying the processes.

Server Definitions

In order for your adapter to work properly, you must set up the application servers that run the components used by the adapter to communicate with each other as follows:

- The Oracle Utilities Smart Grid Gateway application server must be able to send and receive messages to and from the SOA Suite application server.
- The SOA Suite application server must be able to send and receive messages to and from the head-end system application server, and the Oracle Utilities Smart Grid Gateway application server.
- The head-end system application server must be able to receive and send messages from and to the SOA Suite application server.

The types of servers and the tokens that need to be replaced for each are listed below. Further below are the specific configuration files that must be modified, as well as the locations in each file that must be modified. Note that a port is listed, but may not be necessary depending on the type of installation. Also note that the SOA Server will have to have the partition name defined that is used.

- **SOA Server:** This is the application server running the SOA Suite components, including Oracle Service Bus (OSB) and Oracle Business Process Execution Language (BPEL). The SOA server is referenced using the following tokens:
 - {SOA_HOST}: the server where the SOA server has been installed
 - {SOA_PORT_NUMBER}: the port used by the SOA server
 - {SOA_PARTITION_DG}: the partition used by the SOA server. There are different partitions for the different adapters used by Oracle Utilities Smart Grid Gateway.
- **XAI (OUAF) Server:** This is the application server running the Oracle Utilities Smart Grid Gateway (including the Oracle Utilities Application Framework, or OUAF) software, including the XML Application Integration (XAI) components. The XAI (OUAF) server is referenced using the following tokens:
 - {WEB_WLHOST}: the server where Oracle Utilities Smart Grid Gateway and the XAI components have been installed
 - {WEB_WLPORT}: the port used by the Oracle Utilities Smart Grid Gateway application
 - {WEB_CONTEXT_ROOT}: the root directory at which the Oracle Utilities Smart Grid Gateway XAI inbound service
- **AMI Head-end Server:** This is the application server running the head-end system software. The AMI head-end server is referenced using the following tokens:

- `{Headend_MR_Server_DG}`: the URL for either the actual head-end system's MR_Server MultiSpeak implementation, or the URL to an emulator test harness being used (if applicable). For example, the Adapter Development Kit includes a soapUI configuration that can be used to emulate a head-end system for testing purposes.
- `{Headend_CD_Server_DG}`: the URL for either the actual head-end system's CD_Server MultiSpeak implementation, or the URL to an emulator test harness being used (if applicable). For example, the generic adapter includes a soapUI configuration that can be used to emulate a head-end system for testing purposes.
- `Headend_OD_Server_DG`: the URL for either the actual head-end system's MR_Server MultiSpeak implementation, or the URL to an emulator test harness being used (if applicable). For example, the generic adapter includes a soapUI configuration that can be used to emulate a head-end system for testing purposes.

In addition, credentials for the WebLogic server must be specified in the `build.properties` file in the `deploy` folder:

- `{WebLogic_UserID}`: the user ID used to connect to the WebLogic server
- `{WebLogic_Password}`: the password for the `{WebLogic_UserID}` used to connect to the WebLogic server

Customizing the BPEL Processes

This section provides an overview of how the sample BPEL processes provided with the Oracle Utilities Smart Grid Gateway Adapter Development Kit can be customized to meet specific business requirements, such as using custom XSL transformations and adding/editing the steps involved in the process.

Using Custom XSL Files

This process references XSL files to transform the messages from the Oracle Utilities Smart Grid Gateway standard format into the format used by the “generic” head-end system (based on the Multispeak protocol) when sending messages to the head-end system, and from the head-end system format into the Oracle Utilities Smart Grid Gateway standard format when sending messages back to Oracle Utilities Smart Grid Gateway. These XSL file references can be changed to custom XSL files that transform the messages into and from the format used by your head-end system. Refer to Oracle BPEL and Oracle JDeveloper documentation for more information about referencing XSL files within a BPEL process.

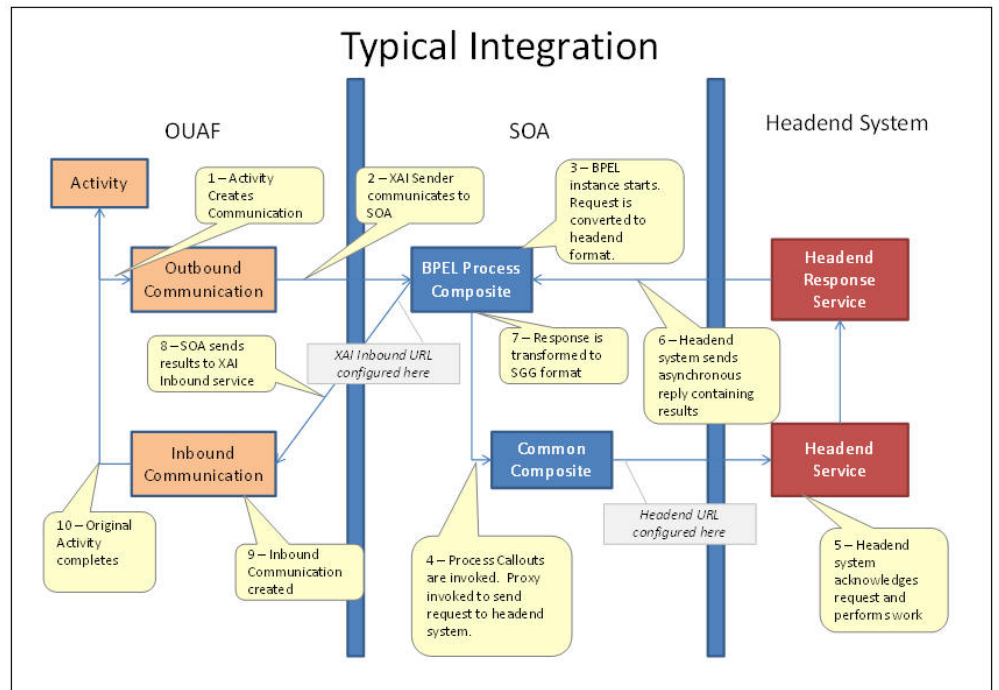
Making Changes to the Process

In addition to referencing custom XSL files for transformation, you can also add additional steps to the process to meet the specific business requirements of your implementation, or make other changes to the existing steps. Refer to Oracle BPEL and Oracle JDeveloper documentation for more information about adding and/or changing steps within BPEL process.

Note: Any changes made to this process for a particular AMI vendor and head-end system needs to be modeled based on the head-end system/AMI vendor's business processes, including configuring the required request/response transactions required for a specific command (such a remote connect).

Commands

This section describes the general structure and flow of each SOA composite in the adapter development kit for the two-way commands. Most follow this basic model:



This section describes the following commands:

- **Common**
- **Commission/Decommission**
- **Connect/Disconnect**
- **Device Status Check**
- **On Demand Read**

Common

The Common composite is a repository for files and logic that is used by other composites. For instance, the Proxies and default implementations of Process Callouts are defined in the Common composite.

Composite Properties

None.

Composite WSDLs

Callout Services

Process Callouts are customization points for integrators with identical inputs and outputs based on head-end formats. The Common composite contains default implementations in which incoming data is reflected back in an identical state. Each of the following is a Mediator component and uses an "echo"-type definition.

Process Callouts are usually associated with boolean composite properties which control whether they will be executed. They are activated by default so users can easily inject their customizations,

but setting the properties to false can be a performance optimization. When set to false, the associated Process Callout will not be made and execution will continue normally.

Service	Description
OA_CalloutService	Used when the asynchronous reply in DeviceStatusCheck arrives.
MR_CalloutService	Used during Commission/Decommission and OnDemandRead operations.
CB_CalloutService	Used by the asynchronous callback processes of OnDemandRead and Connect/Disconnect.
CD_CalloutService	Used by Connect/Disconnect.
OD_CalloutService	Used in DeviceStatusCheck.

Proxies

Placing all the proxy web services in the Common composite creates a place to conveniently set endpoints. Each is a Mediator component which passes the data to the head-end system without modification

Proxy Web Service	Description
OD_ServerProxy	Used to transmit a DeviceStatusCheck request to the head-end system.
CD_ServerProxy	Used to transmit a Connect/Disconnect request to the head-end system.
MR_ServerProxy	Used to transmit an OnDemandRead request to the head-end system.

Other

AuxiliaryRoutinesService - A container defining helpful, commonly used functions:

- **FindExpTime:** Many MultiSpeak functions contain an expTime element, which is used to deliver the length of time to wait for the command to complete before failing. (The included test harness does not support timeouts.) If the input to the command composite does not contain this timeout, the process will read from its properties the years, months, days, hours, minutes, and seconds to wait for a result. This web service utilizes a Java method to combine these inputs into the proper XML field used in the MultiSpeak API. The properties are also used to control the timeout wait period of asynchronous callbacks within a command's BPEL process.
- **FindTimeout:** When the expTime element and units are supplied as input, it is usually still necessary to compute the timeout used in asynchronous callbacks. This method accepts the MultiSpeak fields and replies in the "P0Y0M0DT0H0M0S" format understood by BPEL.

Commission/Decommission

The CommissionDecommission composite takes care of registering and de-registering a device with the head-end system.

Composite Properties

Property Name	Default Value	Description
isExecutingCommissionReceived-Callout	true	Controls whether the Request Arrived callout executes.
isExecutingCommissionCompleted-Callout	true	Controls whether the Request Completed callout executes.
isExecutingDecommissionReceived-Callout	true	Controls whether the Request Arrived callout executes.
isExecutingDecommissionCompleted-Callout	true	Controls whether the Request Completed callout executes.

Composite WSDLs

Composite	Description
CommissionDecommission Service	Entry point for the CommissionDecommission command. The operations are synchronous, so no additional WSDLs are needed.
MR_ServerCallout	Reference describing the process callout.
MR_ServerProxy	Reference describing the proxy used to invoke the head-end system.

Process Flow

1. SGG/OUAF invokes MeterAddNotification/MeterRemoveNotification operation using CommissionDecommissionService.
2. Test whether MeterAddNotification or MeterRemoveNotification is invoked. In both cases, the following steps are common for both the operations.
3. Composite properties are loaded and local variables are initialized.
4. Both the Header and Body inputs are transformed to MultiSpeak format.
5. If required, execute a process callout in MR_ServerCallout. Assign updated data to head-end request.
6. Invoke MeterAddNotification/MeterRemoveNotification via the MR_ServerProxy to the head-end system.
7. If required, execute a process callout in MR_ServerCallout. Assign updated data to head-end response.
8. Transform Header and Body head-end responses to SGG/OUAF format.
9. Reply to SGG/OUAF with synchronous results.

Connect/Disconnect

The ConnectDisconnect composite is responsible for starting and stopping the recording of usage data for a meter.

Composite Properties

Property Name	Default Value	Description
CB_CDCallbackEndpoint	http:// txdev2k3vm5.us.oracle.com:8001 /soa-infra/services/Generic Adapter/ConnectDisconnect/ CB_ServerService	The default endpoint to which MultiSpeak should send the asynchronous CB_Server callback.
isExecutingCDReceivedCallout	true	Controls whether the Request Received callout executes.
isExecutingCDCompletedCallout	true	Controls whether the Request Received Completed callout executes.
isExecutingCDStatesChangedArrivedCallout	true	Controls whether the CD States Changed Notification arrival callout executes.
CallbackTimeoutYears	0	The number of years to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutMonths	0	The number of months to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutDays	0	The number of days to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutHours	0	The number of hours to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.

Property Name	Default Value	Description
CallbackTimeoutMinutes	0	The number of minutes to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutSeconds	45	The number of seconds to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.

Composite WSDLs

Composite	Description
ConnectDisconnectService	Entry point for the ConnectDisconnect operation. After a synchronous reply, further updates will be sent to the specified location asynchronously using DG-ConnectDisconnectStateChgNotification.
CB_ServerService	MultiSpeak WSDL hosted to receive asynchronous CDStatesChangedNotification callbacks from the head end system.
CB_ServerCallout	Reference describing the process callout used when the asynchronous callback from the head end system arrives.
CD_ServerCallout	Reference describing the process callout.
CD_ServerProxy	Reference describing the proxy used to invoke the head end system.
AuxiliaryRoutines	Reference to common helper routines in the Common composite.
DG-ConnectDisconnectStateChgNotification	A reference to an XAI Inbound service capable of processing asynchronous results from a Connect/Disconnect request.

Process Flow

1. SGG/OUAF invokes InitiateConnectDisconnect operation using ConnectDisconnectService.
2. Composite properties are loaded and local variables are initialized.
3. Test whether the input includes a expiration time
 - If Yes: Use AuxiliaryRoutines to compute timeout.
 - If No: Use composite properties and AuxiliaryRoutines to compute expiration time.
4. Both the Header and Body inputs are transformed to MultiSpeak format.

5. If required, execute a process callout in CD_ServerCallout. Assign updated data to head end request.
6. Add a callback URL to the head end request. This is the endpoint the head end will use to send the results.
7. Invoke InitiateConnectDisconnect via the CD_ServerProxy to the head end system.
8. If required, execute a process callout in CD_ServerCallout. Assign updated data to head end response.
9. Transform Header and Body head end responses to SGG/OUAF format.
10. Reply to SGG/OUAF with synchronous results.
11. Check for ErrorObject in the head end response.
12. If error object not found:

Await asynchronous response from head-end system for the period of expiration time calculated during start of this flow:

- Receive CDStatesChangeNotification from CB_ServiceService.
- Generate reply with no errors for CDStatesChangeNotification
- Synchronously reply to the head end.
- If required, execute a process callout in CB_ServerCallout. Assign updated data to the incoming request.
- Transform Header and Body of incoming request to OUAF XAI Inbound format described in DG-ConnectDisconnectStateChgNotification.
- If request contains responseURL
 - Invoke the XAI Inbound Service on given responseURL with the transformed incoming request.
- Else
 - Invoke the XAI Inbound Service on definedURL with the transformed incoming request.

Device Status Check

DeviceStatusCheck is used to determine the health of a meter and to test that the meter can be reached on the network.

Composite Properties

Property Name	Default Value	Description
OD_ServerCallbackEndpoint	http://127.0.0.1:8001/soa-infra/services/Sensus/DeviceStatusCheck/OA_ServerService	The default endpoint to which MultiSpeak should send the asynchronous OA_Server callback.
isExecutingInitODRequest-ReceivedCallout	true	Controls whether the Request Arrived callout executes.
isExecutingInitODEventRequest-Completed	true	Controls whether the Request Completed callout executes.
isExecutingODEventNotification-Arrived	true	Controls whether the ODEventNotification arrival callout executes.
timeout.callback.years	0	The number of years to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
timeout.callback.months	0	The number of months to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
timeout.callback.days	0	The number of days to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
timeout.callback.hours	0	The number of hours to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
timeout.callback.minutes	0	The number of minutes to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.

Property Name	Default Value	Description
timeout.callback.seconds	45	The number of seconds to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.

Composite WSDLs

Composite	Description
OD_ServerService	Entry point for DeviceStatusCheck. After a synchronous reply, further updates will be sent to the specified location asynchronously using DG-OutageDetectionEventNotification.
OA_ServerService	MultiSpeak WSDL hosted to received asynchronous ODEventNotification callbacks from the head end system.
OD_ServerProxy	Reference describing the proxy used to invoke the head end system to initiate a Device Status Check.
OA_ServerCallout	Reference describing the process callout used when the asynchronous callback from the head end system arrives.
OD_ServerCallout	Reference describing the process callout used when initiating a Device Status Check.
AuxiliaryRoutines	Reference to common helper routines in the Common composite.
DG-OutageDetectionEventNotificationXAI	A reference to an XAI Inbound service capable of processing asynchronous results from a Device Status Check request.

Process Flow

1. SGG/OUAF Initiates the Outage Detection service using OD_ServerService
2. Composite properties are loaded. Local variables are initialized. These mostly include boolean settings describing the state (such as whether or not the synchronous reply from the head-end system returned)
3. Test whether the input includes a expiration time (expTime element).
 - If Yes: Use AuxiliaryRoutines to compute timeout
 - If No: Use composite properties and AuxiliaryRoutines to compute expiration time
4. Both the Header and Body inputs are transformed to MultiSpeak format. This can be done either through direct mapping or through XSLT. In this instance, XSLT is used to transform the header and the body separately.
5. If the composite property indicates the process callout should occur, execute the “request arrived” process callout in OD_CalloutService. Assign updated data to head end request.
6. Add a callback URL to the head end request. This is the endpoint the head end will use to send the outage detection results. Some head end systems use other approaches, such as pre-configuring the URL for callbacks or allowing polling to find the results. In this case, the URL

is configured in a composite property and corresponds to an implementation of the OA_Server WSDL. When the head end system has results, it will deliver them to this URL using the ODEventNotification method.

7. Invoke InitiateOutageDetectionEventRequest via the OD_ServerProxy to the head end system.
8. Split processing to handle simultaneous activities. The Flow activity does this in BPEL and it makes sense here because one process will handle a synchronous response and return it to the caller. The other process will sleep, or “dehydrate,” until the asynchronous ODEventNotification arrives or until the timeout value is reached.
 - Handle synchronous response to SGG/OUAF
 - If the composite property indicates the process callout should occur, execute the “request completed” process callout in OD_CalloutService. Assign updated data to head end response.
 - Transform Header and Body head end responses to SGG/OUAF format.
 - Reply to SGG/OUAF with synchronous results.
 - Set boolean variable indicating that the synchronous reply has been returned.
 - Await asynchronous response from head-end system.
 - Receive ODEventNotification from OA_ServerService. If the timeout period elapses, raise an error and stop waiting.
 - Synchronously reply to the head end with no errors
 - If the composite property indicates the process callout should occur, execute the “data arrived” process callout in OA_CalloutService. Assign updated data to the incoming request.
 - Transform Header and Body to XAI Inbound format described in DG-OutageDetectionEventNotification.
 - Invoke the XAI Inbound Service with the new data.
9. Fault Handlers look for any remote Fault returned from the head-end system. The MultiSpeak API does not define named faults, but they are still possible to encounter. For example, a security error or network error could be raised.
 - If the synchronous reply has been sent back to the caller, Terminate the flow. This indicates an error has occurred and more research in Enterprise Manager is warranted.
 - If the synchronous reply has not been sent, abort further processing and compose a reply which contains the fault information.

On Demand Read

OnDemandRead interrogates a meter for the usage at the current point in time.

Composite Properties

Property Name	Default Value	Description
CB_ServerCallbackEndpoint	http:// txdev2k3vm5.us.oracle.com:8001 /soa-infra/services/Generic Adapter/OnDemandRead/ CB_Server	The default endpoint to which MultiSpeak should send the asynchronous CB_Server callback
IsExecutingOnDemandRead RequestReceived	true	Controls whether the Request Received callout executes
IsExecutingOnDemandRead RequestReceivedResponse	true	Controls whether the Request Received Completed callout executes
IsExecutingReadingChanged Notification	true	Controls whether the Reading Changed Notification arrival callout executes
CallbackTimeoutYears	0	The number of years to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutMonths	0	The number of months to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutDays	0	The number of days to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutHours	0	The number of hours to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutMinutes	0	The number of minutes to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.

Property Name	Default Value	Description
CallbackTimeoutSeconds	45	The number of seconds to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.

Composite WSDLs

Composite	Description
OnDemandReadService	Entry point for the OnDemandRead operation. After a synchronous reply, further updates will be sent to the specified location asynchronously using DG-ReadingChangedNotification.
CB_ServerService	MultiSpeak WSDL hosted to receive asynchronous ReadingChangedNotification callbacks from the head end system.
MR_ServerProxy	Reference describing the proxy used to invoke the head end system to initiate a On Demand Read.
MR_ServerCallout	Reference describing the process callout used when initiating a On Demand Read.
CB_ServerCallout	Reference describing the process callout used when the asynchronous callback from the head end system arrives.
AuxiliaryRoutines	Reference to common helper routines in the Common composite.
DG-ReadingChangedNotification XAI	A reference to an XAI Inbound service capable of processing asynchronous results from a On Demand Read request.

Process Flow

1. SGG/OUAF invokes InitiateMeterReadingsByMeterID operation using OnDemandReadService.
2. Composite properties are loaded. Local variables are initialized.
3. Test whether the input includes a expiration time
 - If Yes: Use AuxiliaryRoutines to compute timeout
 - If No: Use composite properties and AuxiliaryRoutines to compute expiration time
4. Both the Header and Body inputs are transformed to MultiSpeak format.
5. If required, execute a process callout in MR_ServerCallout. Assign updated data to head end request.
6. Add a callback URL to the head end request. This is the endpoint the head end will use to send the results.
7. Invoke InitiateMeterReadingsByMeterID via the MR_ServerProxy to the head end system.

8. If required, execute a process callout in MR_ServerCallout. Assign updated data to head end response.
9. Transform Header and Body head end responses to SGG/OUAF format
10. Reply to SGG/OUAF with synchronous results
11. Check for ErrorObject in the head end response.
12. If error object not found
 - Await asynchronous response from head end for the period of expiration time calculated during start of this flow
 - Receive ReadingChangedNotification from CB_ServiceService.
 - Generate reply with no errors for ReadingChangedNotification.
 - Synchronously reply to the head end.
 - If required, execute a process callout in CB_ServerCallout. Assign updated data to the incoming request.
 - Transform Header and Body of incoming request to OUAF XAI Inbound format described in DG-ReadingChangedNotificationXAI.
 - If request contains responseURL
 - Invoke the XAI Inbound Service on given responseURL with the transformed incoming request.
 - Else
 - Invoke the XAI Inbound Service on definedURL with the transformed incoming request.

Working with Enterprise Manager

Oracle Enterprise Manager (OEM) is useful for troubleshooting and diagnosing issues communicating with the head end system. In particular, security and other communications issues can be discerned.

To locate the instance of a service that is showing a problem, open the Dashboard view of the service. Each run instance is time-indexed. The State column in the first table contains the most important information for each instance. Completed messages show up with a green check icon. Terminated instances are indicated with a grey stop icon. Instances that are still running are marked as such. Runtime exceptions are in the bottom window.

The screenshot shows the Oracle Enterprise Manager Dashboard for the service 'DeviceStatusCheck [1.0]'. The top navigation bar includes 'Dashboard', 'Instances', 'Faults and Rejected Messages', 'Unit Tests', and 'Policies'. The 'Recent Instances' section displays a table with the following data:

Instance ID	Name	Conversation ID	State	Start Time
1792068			Terminated	Oct 12, 2012 3:40:42 PM
1792032		urn:SDE6058014931	Completed	Oct 12, 2012 1:36:34 PM
1792026			Completed	Oct 12, 2012 1:36:16 PM
1671020		urn:A52A7FF0DB2E	Completed	Jul 31, 2012 12:41:58 PM
1671014			Completed	Jul 31, 2012 12:41:46 PM

The 'Recent Faults and Rejected Messages' section shows a table with the following data:

Error Message	Recovery	Fault Time	Fault Location	Composite Instance ID	Logs
Unable to find operation: null		Aug 1, 2012 4:40:07 PM	OD_ServerService Unavailable		
Unable to find operation: null		Aug 1, 2012 2:40:29 PM	OD_ServerService Unavailable		
Unable to find operation: null		Aug 1, 2012 2:16:09 PM	OD_ServerService Unavailable		
Unable to find operation: null		Aug 1, 2012 2:15:59 PM	OD_ServerService Unavailable		
Unable to find operation: null		Aug 1, 2012 2:14:53 PM	OD_ServerService Unavailable		
<bpelFault><faultType>0</faultType>		Jul 31, 2012 10:38:05 AM	DeviceStatusCheck	1670903	
javax.xml.ws.soap.SOAPFaultException:		Jul 31, 2012 10:38:04 AM	OD_ServerProxy	1670903	

When you click on the instance ID, a new window opens showing the entire process flow. This view is particularly useful in debugging. For instance, clicking on the 1792068 instance in the above image, shows this flow sequence:

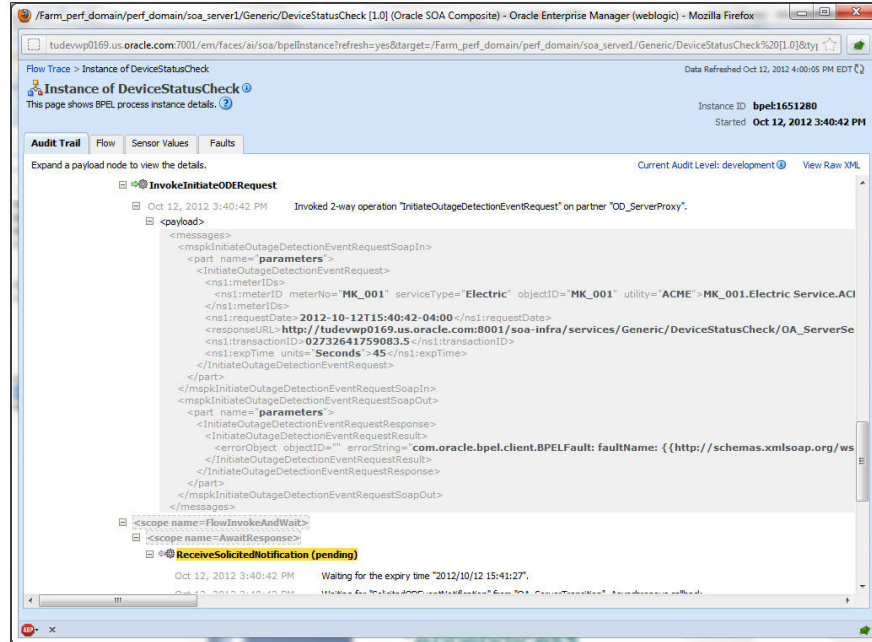
The screenshot shows the 'Flow Trace' view for instance 1792068. The page displays a 'Faults (1)' section with the following error message:

Error Message	Recc	Fault Time	Composite Instance	Fault Location
<bpelFault><faultType>0</faultType><selectionFailure xmlns='http://schemas.xmlsoap.org/ws/2003/03/busi		Oct 12, 2012 3:40:42 PM	DGTestHarness	OD of bpel:1651282

The 'Trace' section shows a detailed audit trail of the message flow through various component instances:

Type	Instance	Usage	State	Composite Instance	Time
Web Service	OD_ServerService	Service	Completed	DeviceStatusCheck of 1792068	Oct 12, 2012 3:40:42 PM
BPDL Component	DeviceStatusCheck		Terminated	DeviceStatusCheck of 1792068	Oct 12, 2012 3:40:42 PM
Web Service	AuxiliaryRoutines	Reference	Completed	DeviceStatusCheck of 1792068	Oct 12, 2012 3:40:42 PM
Web Service	AuxiliaryRoutinesService	Service	Completed	Common of 1792069	Oct 12, 2012 3:40:42 PM
BPDL Component	AuxiliaryRoutines		Completed	Common of 1792069	Oct 12, 2012 3:40:42 PM
Web Service	OD_ServerCallout	Reference	Completed	DeviceStatusCheck of 1792068	Oct 12, 2012 3:40:42 PM
Web Service	OD_ServerCalloutService	Service	Completed	Common of 1792070	Oct 12, 2012 3:40:42 PM
Mediator Component	OD_ServerCalloutMediator		Completed	Common of 1792070	Oct 12, 2012 3:40:42 PM
Web Service	OD_ServerProxy	Reference	Completed	DeviceStatusCheck of 1792068	Oct 12, 2012 3:40:42 PM
Web Service	OD_ServerProxy	Service	Completed	Common of 1792071	Oct 12, 2012 3:40:42 PM
Mediator Component	OD_ServerMediator		Completed	Common of 1792071	Oct 12, 2012 3:40:42 PM
Web Service	OD_Server	Reference	Completed	Common of 1792071	Oct 12, 2012 3:40:42 PM
Web Service	OD_Server	Service	Completed	Common of 1792072	Oct 12, 2012 3:40:42 PM
BPDL Component	OD		Completed	DGTestHarness of 1792072	Oct 12, 2012 3:40:42 PM
BPDL Component	UtilityOps		Completed	DGTestHarness of 1792072	Oct 12, 2012 3:40:42 PM

Note the State column showing the status of each instance within the flow. SOA processes are made up of several calls to SOA components and web services. The Instance column shows a rough ordering of the operations. A typical approach to troubleshooting would find the “lowest” instance of an error or termination. This would be the error that is deepest within the process and is usually the source of the problem. In the above case, the deepest error is in the very first composite for the process. More information can be found by clicking into the link within the Instance column.



Each step of the composite is shown in this view. The level of detail is dependent on the auditing level set on the server. In this case, the error is in the response from the head end system.

OEM also contains a centralized location to control security. On a Dashboard screen for a composite, the Policies tab shows the OWSM policies attached to the composite. Typically, a log policy is placed on any inbound or outbound communication. Also, all services or references are delivered with attached basic http security policies.

Policy Name	Attached To	Policy Reference Status	Category	Total Violations
oracle/log_policy	MR_Server	Disable	Management	0
oracle/log_policy	OD_Server	Disable	Management	0
oracle/log_policy	UtilService	Disable	Management	0
oracle/log_policy	CD_Server	Disable	Management	0
oracle/log_policy	Util	Disable	Management	0
oracle/wss_http_token_client_policy	OA_Server	Disable	Security	0
oracle/log_policy	OA_Server	Disable	Management	0
oracle/wss_http_token_client_policy	CB_Server	Disable	Security	0
oracle/log_policy	CB_Server	Disable	Management	0

MultiSpeak Implementation

This section lists the subset of MultiSpeak schemas and WSDLs utilized by the ADK.

MultiSpeak Head End System Service Definitions

- CB_Messages.wsdl
- CB_Server.wsdl
- CD_Messages.wsdl
- CD_PortTypes.wsdl
- CD_Server.wsdl
- MR_Messages.wsdl
- MR_PortTypes.wsdl
- MR_Server.wsdl
- OA_Messages.wsdl
- OA_Server.wsdl
- OA_ServerCollisions.wsdl
- OD_Messages.wsdl
- OD_PortTypes.wsdl
- OD_Server.wsdl

MultiSpeak Head End XML Schemas

- cpsm.xsd
- gml.xsd
- xlink.xsd
- mspcommon.xsd
- CB_Server.xsd
- CD_Server.xsd
- MR_Server.xsd
- OA_Server.xsd
- OD_Server.xsd

The Test Harness

Oracle Utilities Smart Grid Gateway Adapter Development Kit includes a test harness that can be configured to simulate a general head-end system for testing the two-way commands. The test harness includes a BPEL composite, web services for standard meter functions, and an XML file that can be used to contain information for one or more meters. This chapter describes the test harness and its components, including:

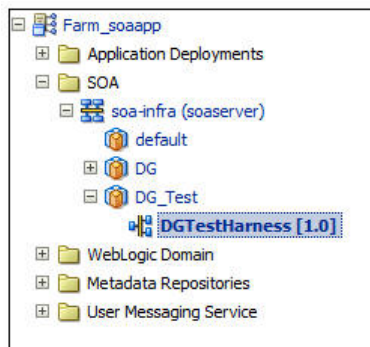
- **Locating the WSDL for the Test Harness**
- **Web Services**
- **The Sample Meters File**

Locating the WSDL for the Test Harness

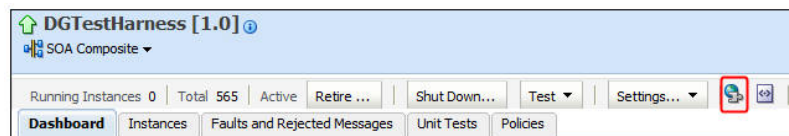
Follow these procedures to locate the test harness WSDL:

How to Use Enterprise Manager to Locate the WSDL

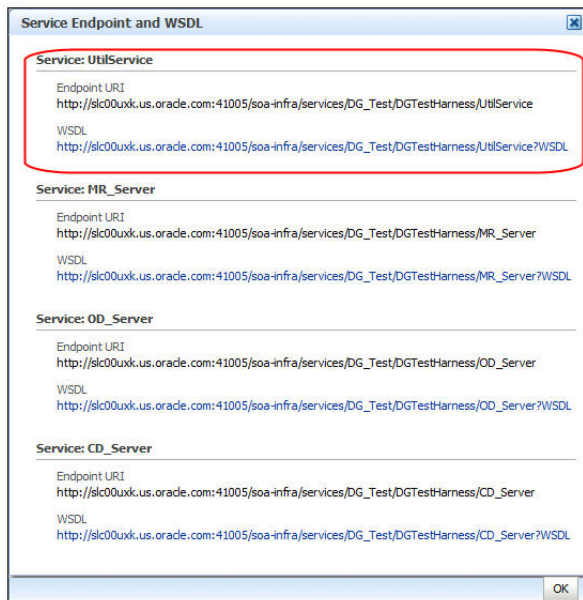
1. Open Enterprise Manager and use the navigation pane to open the dashboard of the test harness composite:



2. The top bar of the dashboard contains several buttons and icons. One of these is a “world” icon with a puzzle piece over it. Click this icon to display a list of the WSDLs and endpoint URIs for the composite:



- Click the UtilService WSDL URL link to see the WSDL in the browser, or right click and save it to your machine



Depending on your requirements, it may be necessary to download the associated schema found in the `wSDL:types` section. The URL can be pasted into a browser tab and downloaded in the same manner as the WSDL. The main schema has imported schemas that may also be required.

How to Use a Direct URL to locate the WSDL

The WSDL can be accessed without Enterprise Manager by understanding the paths used on the SOA server. In general, they have the following form:

```
http://{server name}:{port number}/soa-infra/services/{partition}/
{Composite}/{Web Service}?WSDL
```

So by default, the test harness WSDL can be found at

```
http://{server name}:{port number}/soa-infra/services/DG_Test/
DGTestHarness/UtilService?WSDL
```

Web Services

This section describes the web services included in the test harness BPEL composite, including:

- **Reserved Services**
- **General Services**
- **Locate Meter Services**
- **Meter Administration Services**
- **Meter Attribute Administration Services**

Reserved Services

This section describes the reserved services of the test harness composite.

ValidateHeader

This service is reserved for internal use only.

General Services

This section describes the general services of the test harness composite, including:

- **LoadMeterIndex**
- **ViewAuditTrail**

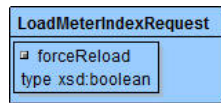
LoadMeterIndex

This web service loads the data store from the internal file. By default if the store is already in memory, it will *not* reload. This behavior can be overridden with the forceReload parameter.

Input - LoadMeterIndexInput

Part: payload

Element: LoadMeterIndexRequest

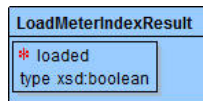


Parameter	Description
forceReload	A switch telling the system whether to reload the meter index from the configuration file. Default value is false.

Output - LoadMeterIndexOutput

Part: payload

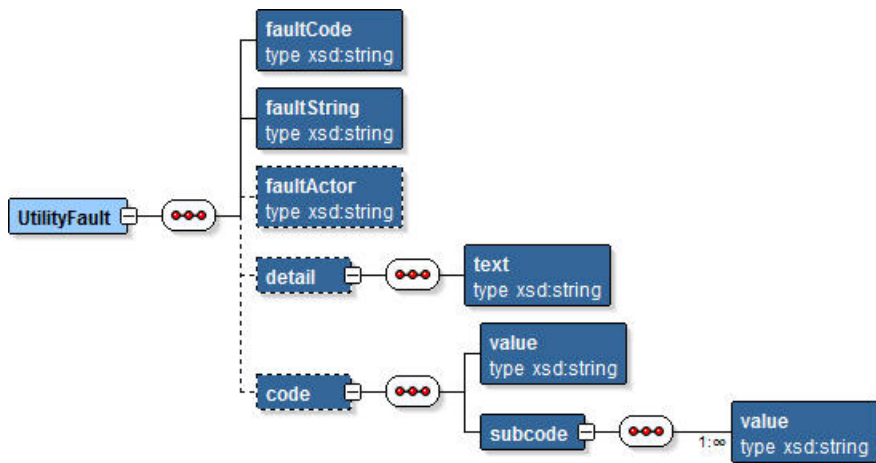
Element: LoadMeterIndexResult



Parameter	Description
loaded	A boolean value for whether or not the index was reloaded from the configuration file

Fault - UtilityFault

Fault with similar mapping to SGG/OUAF faults.



Typically, the faultCode, faultString, faultActor, and detail/text elements will be populated.

ViewAuditTrail

This web service returns the audit log for the entire session.

Input - ViewAuditTrailInput

Part: payload

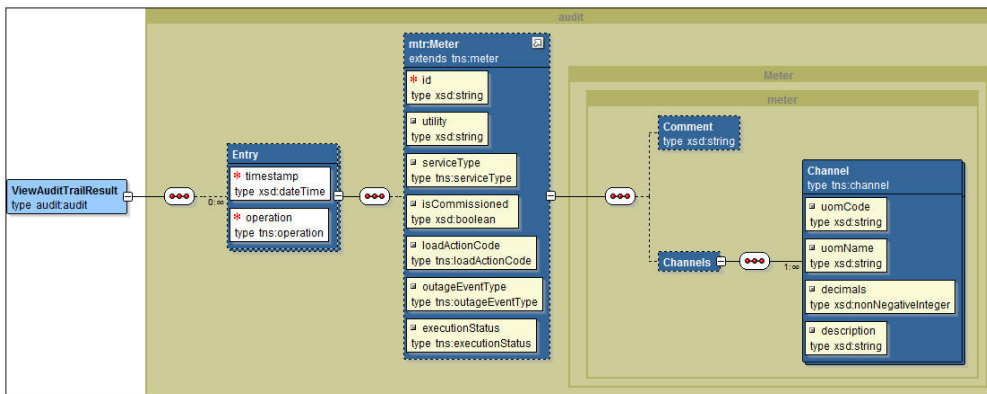
Element: ViewAuditTrailRequest

Parameters: This is an empty request. There are no parameters.

Output - ViewAuditTrailOutput

Part: payload

Element: ViewAuditTrailResult



This element is an entry consisting of a timestamp and an Operation. Each entry may have an associated meter object showing what changed.

Fault - See **UtilityFault** on page 3-27.

Locate Meter Services

This section describes the locate meter web services of the test harness composite, including:

- **FindMeters**
- **IsMeterDefined**
- **GetMeter**

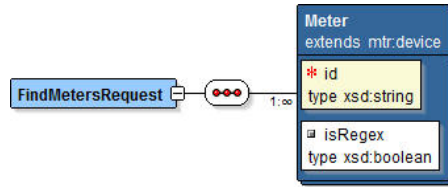
FindMeters

This web service queries the data store for one or more meters. The difference between GetMeter and FindMeters is GetMeter can return at most one meter and it must match the provided ID exactly. GetMeter will throw an error if the ID is not found. FindMeters can return more than one meter (when using the regex) and will not throw an error when the ID does not match any of the meters in the index.

Input - FindMetersInput

Part: payload

Element: FindMetersRequest

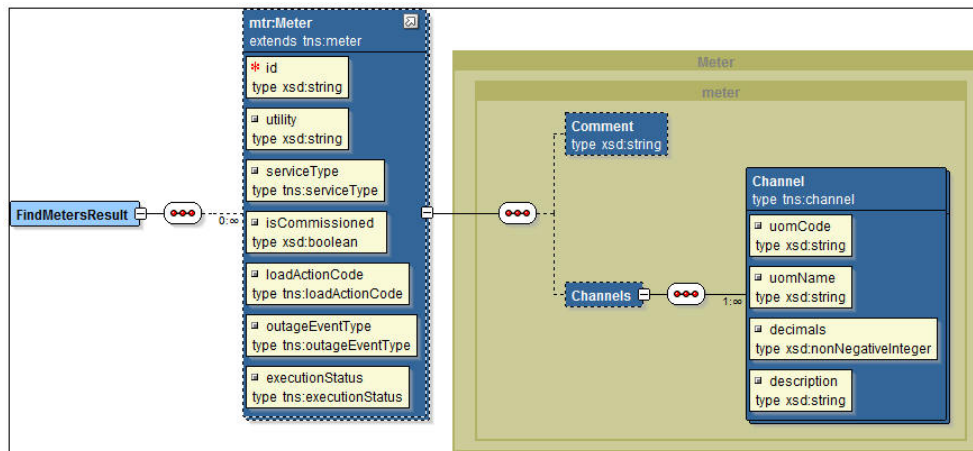


Parameter	Description
id	The meter ID for which to search
isRegex	The provided id can be a regex value when this parameter is true. Hint: to search for all meters in the system, use ".*" for the ID.

Output - FindMetersOutput

Part: payload

Element: FindMetersResult



Zero or more meter objects can be returned from the search

Fault - See **UtilityFault** on page 3-27. Unlike other methods, FindMeters does not throw an exception if the meter is not found.

IsMeterDefined

This web service queries whether a particular meter is defined in the data store.

Input - IsMeterDefinedInput

Part: payload

Element: IsMeterDefinedRequest



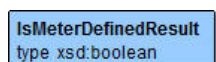
Parameter	Description
-----------	-------------

id	The meter ID for which to search
----	----------------------------------

Output - IsMeterDefinedOutput

Part: payload

Element: IsMeterDefinedResult



Whether or not the provided ID is part of the index.

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

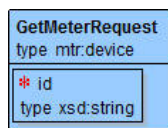
GetMeter

This web service returns all the attributes of a single meter from the in-memory data store. The difference between **GetMeter** and **FindMeters** is **GetMeter** can return at most one meter and it must match the provided ID exactly. **GetMeter** will throw an error if the ID is not found. **FindMeters** can return more than one meter (when using the regex) and will not throw an error when the ID does not match any of the meters in the index.

Input - GetMeterInput

Part: payload

Element: GetMeterRequest



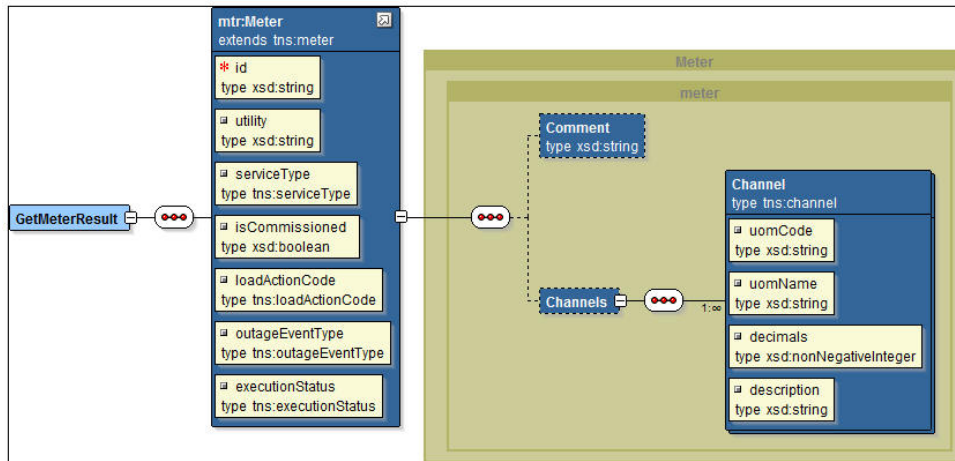
Parameter	Description
-----------	-------------

id	The meter ID for which to search
----	----------------------------------

Output - GetMeterOutput

Part: payload

Element: GetMeterResult



The meter object requested by the ID.

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

Meter Administration Services

This section describes the meter administration services of the test harness composite, including:

- **AddMeters**
- **RemoveMeter**
- **AddMeterChannel**
- **RemoveMeterChannel**
- **ReadScalarMeter**

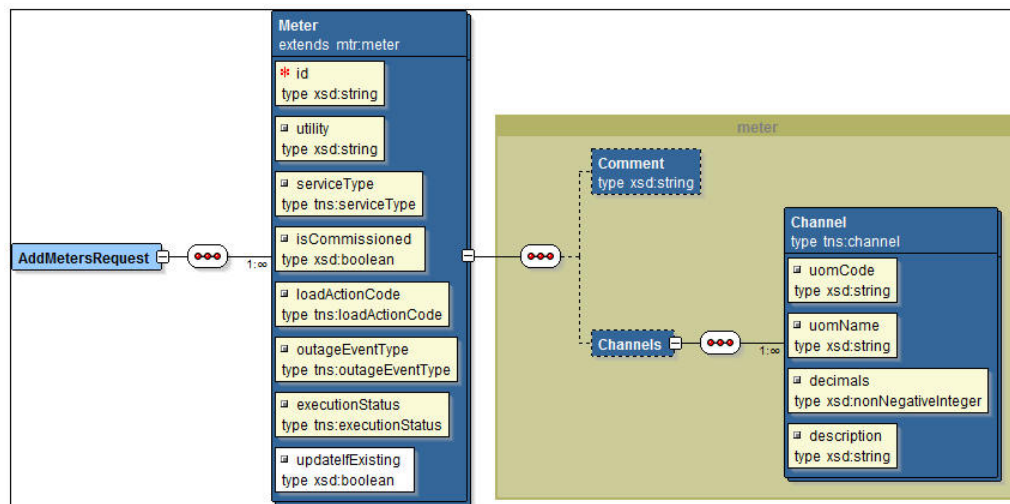
AddMeters

This web service adds a set of meters to the in-memory data store. This will not permanently add it to the control file.

Input - AddMetersInput

Part: payload

Element: AddMetersRequest

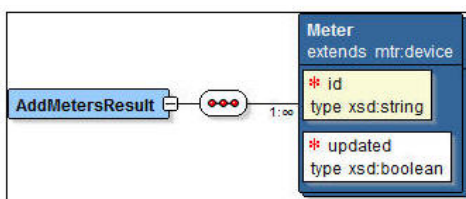


Parameter	Description
id	The identification code for the meter.
utility	An informational string.
serviceType	One of the valid ServiceType values (see schema). "Electric" is the only option at this time.
isCommissioned	Whether or not the meter is in a commissioned state.
loadActionCode	One of the possible LoadActionCode values used in Connect and Disconnect (see schema).
outageEventType	One of the possible OutageEventType values used in Device Status Check (see schema).
executionStatus	One of the possible ExecutionStates (see schema). These values control how the meter will respond to commands.
updateIfExistsing	Whether or not to update the meter with the provided values if it already exists in the index.
Comment	An informational string describing the purpose of the meter.
Channels	A listing of unit of measures supported by this meter.
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.
decimals	The number of digits to the right of the decimal that should be generated when reading the meter.
description	A longer description of the unit of measure.

Output - AddMetersOutput

Part: payload

Element: AddMetersResult



Whether or not each meter was added to the index.

Fault - See **UtilityFault** on page 3-27.

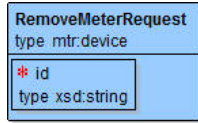
RemoveMeter

This web service removes a meter from the in-memory data store. This will not permanently remove it from the control file.

Input - RemoveMeterInput

Part: payload

Element: RemoveMeterRequest

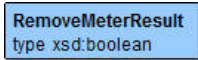


Parameter	Description
id	The ID for the meter to be removed.

Output - RemoveMeterOutput

Part: payload

Element: RemoveMeterResult



Whether or not the meter was removed from the index.

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

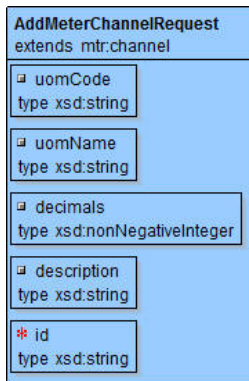
AddMeterChannel

This web service adds a new channel to a single meter.

Input - AddMeterChannelInput

Part: payload

Element: AddMeterChannelRequest



Parameter	Description
id	The identification code for the meter.
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.
decimals	The number of digits to the right of the decimal that should be generated when reading the meter.
description	A longer description of the unit of measure.

Output - AddMeterChannelOutput

Part: payload

Element: AddMeterChannelResult

AddMeterChannelResult
type xsd:boolean

Whether or not the channel was added to the index.

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

RemoveMeterChannel

This web service removes a Channel from a meter.

Input - RemoveMeterChannelInput

Part: payload

Element: RemoveMeterChannelRequest

RemoveMeterChannelRequest
extends mtr:channelID

- uomCode
type xsd:string
- uomName
type xsd:string
- * id
type xsd:string

Parameter	Description
id	The ID for the meter to be removed.
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.

These three parameters are combined to locate a unique channel

Output - RemoveMeterChannelOutput

Part: payload

Element: RemoveMeterChannelResult

RemoveMeterResult
type xsd:boolean

Whether or not the channel was removed from the meter.

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

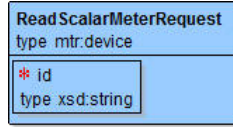
ReadScalarMeter

This web service generates a scalar reading for each channel of a given meter.

Input - ReadScalarMeterInput

Part: payload

Element: ReadScalarMeterRequest

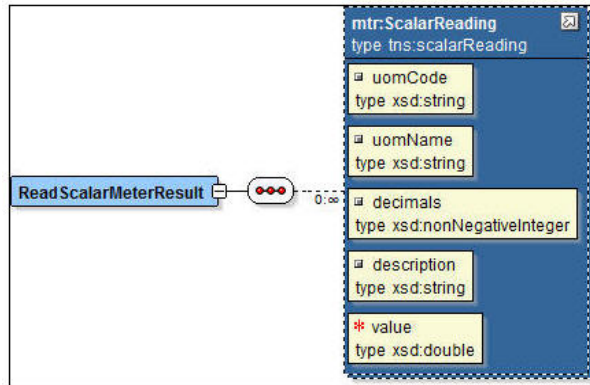


Parameter	Description
id	The ID for the meter to be read.

Output - ReadScalarMeterOutput

Part: payload

Element: ReadScalarMeterResult



Zero or more scalar readings for the given meter.

Parameter	Description
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.
decimals	The number of digits to the right of the decimal that should be generated when reading the meter.
description	A longer description of the unit of measure.
value	A random number representing the scalar reading.

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

Meter Attribute Administration Services

This section describes the meter attribute administration services of the test harness composite, including:

- **GetOutageEventType**
- **SetOutageEventType**
- **GetLoadActionCode**
- **SetLoadActionCode**
- **IsCommissioned**
- **SetCommissioned**
- **GetExecutionStatus**
- **SetExecutionStatus**

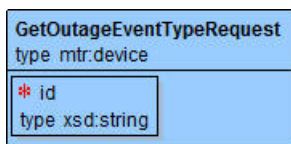
GetOutageEventType

This web service queries the outage event type for a given meter. The OutageEventType is used by DeviceStatusCheck.

Input - GetOutageEventTypeInput

Part: payload

Element: GetOutageEventTypeRequest

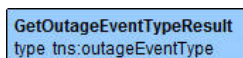


Parameter	Description
id	The ID for the meter for which the OutageEventType should be retrieved.

Output - GetOutageEventTypeOutput

Part: payload

Element: GetOutageEventTypeResult



The value of the OutageEventType attribute for the requested meter.

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

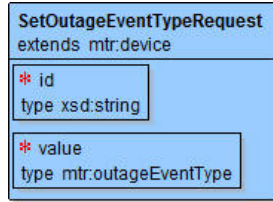
SetOutageEventType

This web service updates the outage event type for a given meter.

Input - SetOutageEventTypeInput

Part: payload

Element: SetOutageEventTypeRequest



Parameter	Description
id	The ID for the meter for which the OutageEventType should be set.
value	The new value of OutageEventType to set on the meter.

Output - SetOutageEventTypeOutput

Part: payload

Element: SetOutageEventTypeResult



The boolean response indicates the success or failure of the update (not the current field status).

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

GetLoadActionCode

This web service queries the load action code for a given meter. This is the Connect/Disconnect behavior.

Input - GetLoadActionCodeInput

Part: payload

Element: GetLoadActionCodeRequest

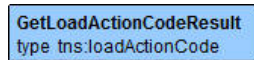


Parameter	Description
id	The ID for the meter for which the LoadActionCode should be retrieved.

Output - GetLoadActionCodeOutput

Part: payload

Element: GetLoadActionCodeResult



The value of the LoadActionCode attribute for the requested meter.

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

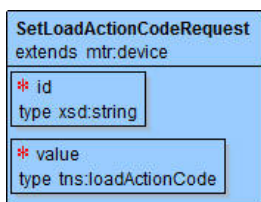
SetLoadActionCode

This web service updates the load action code for a given meter.

Input - SetLoadActionCodeInput

Part: payload

Element: SetLoadActionCodeRequest

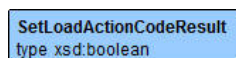


Parameter	Description
id	The ID for the meter for which the LoadActionCode should be set.
value	The new value of LoadActionCode to set on the meter.

Output - SetLoadActionCodeOutput

Part: payload

Element: SetLoadActionCodeResult



The boolean response indicates the success or failure of the update (not the current field status).

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

IsCommissioned

This web service queries the commissioning status for a given meter. This is the Commission/Decommission behavior.

Input - IsCommissionedInput

Part: payload

Element: IsCommissionedRequest

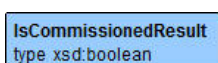


Parameter	Description
id	The ID for the meter for which the Commissioned status should be retrieved.

Output - IsCommissionedOutput

Part: payload

Element: IsCommissionedResult



The value of the Commissioned status attribute for the requested meter.

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

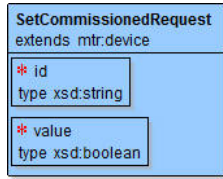
SetCommissioned

This web service updates the commissioning status for a given meter.

Input - SetCommissionedInput

Part: payload

Element: SetCommissionedRequest

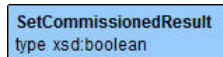


Parameter	Description
id	The ID for the meter for which the Commissioned status should be set.
value	The new value of Commissioned status to set on the meter.

Output - SetCommissionedOutput

Part: payload

Element: SetCommissionedResult



The boolean response indicates the success or failure of the update (not the current field status).

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

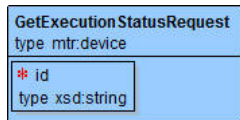
GetExecutionStatus

This web service queries the status of the property controlling the overall execution of the command.

Input - GetExecutionStatusInput

Part: payload

Element: GetExecutionStatusRequest

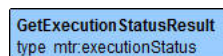


Parameter	Description
id	The ID for the meter for which the ExecutionStatus should be retrieved.

Output - GetExecutionStatusOutput

Part: payload

Element: GetExecutionStatusResult



The value of the ExecutionStatus attribute for the requested meter.

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

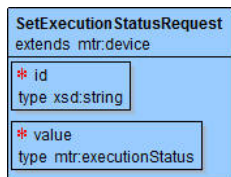
SetExecutionStatus

This web service updates the property controlling the overall completion of the command.

Input - SetExecutionStatusInput

Part: payload

Element: SetExecutionStatusRequest

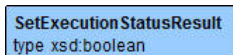


Parameter	Description
id	The ID for the meter for which the ExecutionStatus should be set.
value	The new value of ExecutionStatus to set on the meter.

Output - SetExecutionStatusOutput

Part: payload

Element: SetExecutionStatusResult



The boolean response indicates the success or failure of the update (not the current field status).

Fault - See **UtilityFault** on page 3-27. Thrown when meter id is not found.

The Sample Meters File

The adapter development kit includes an XML file and schema that can be used for configuring one or more meters for use with the test harness. The file, `metersdb.xml`, is located in the `Test/DGTestHarness` directory and can be edited with an appropriate XML editor such as XML Spy. This section describes the attributes in the `metersdb.xml` file.

```
meterdb.xml
<?xml version="1.0" encoding="UTF-8"?>
<Meters xmlns="http://www.multispeak.org/Version_4.1_Release/Meters">
  <Meter id="MK_001" utility="EProvider" serviceType="Electric"
    isCommissioned="true" loadActionCode="Connect"
    outageEventType="PowerOn" executionStatus="Success">
    <Comment>Good ping</Comment>
  </Meter>
  <Meter id="SN_SCALAR" utility="EProvider" serviceType="Electric"
    isCommissioned="true" loadActionCode="Connect"
    executionStatus="Success">
    <Comment>ODR: Scalar Read</Comment>
    <Channels>
      <Channel decimals="2" uomCode="kWh"
        uomName="Kilowatt Per Hour" description="Billing"/>
    </Channels>
  </Meter>
  ...

```

The `meterdb.xml` file contains one or more **Meter** elements that have the following attributes:

Meter Attribute	Definition
id	The meter identifier. This value should match the <code>amiDeviceID</code> setting in the Oracle Utilities Application Framework.
utility	This is informational only.
serviceType	This is informational only.
ServiceType	An enumeration that is described in the schema.
isCommissioned	A boolean value that describes whether or not the meter is commissioned or decommissioned. The associated MultiSpeak commands are <code>MeterAddNotification</code> and <code>MeterRemoveNotification</code> .
loadActionCode	This value is for the <code>Connect</code> and <code>Disconnect</code> commands and has enumeration values described in the schema. The value is returned in the <code>CDStatesChangedNotification</code> MultiSpeak command.
outageEventType	This value is used by <code>DeviceStatusCheck</code> and gives the status returned by the <code>ODEventNotification</code> call. Its enumeration values are described in the schema.

Meter Attribute	Definition
executionStatus	<p>This value does not reflect a MultiSpeak command, but is instead intended to give a state of the operation. There are four valid values:</p> <ul style="list-style-type: none"> • Success: When a meter has this status the operation will complete without error. • ResponseTimeout: When a meter has this status an asynchronous reply will never arrive (not relevant for Commission/Decommission commands). • SyncOperationFailure: When a meter has this status the initial communication to the simulated head-end system will produce an error. • AsyncOperationFailure: When a meter has this status, the asynchronous callback from the head-end system will arrive, but will indicate an error (not relevant for Commission/Decommission commands).

A meter can also contain the following elements:

- **Comment:** A field which is for informational use only and is meant to indicate the purpose of the meter.
- **Channels:** Used in reading the meter for On Demand Read commands. A channel contains the following attributes:

Channel Attribute	Definition
uomCode	The units of measure code that should be returned when the meter is read.
uomName	The units of measure name that should be returned when the meter is read.
description	A longer form. When the meter is read, a random number is generated for this value.
decimals	A value to indicate how many places to the right of the decimal that random number should have.

Chapter 4

Configuring Oracle Utilities Smart Grid Gateway

The Oracle Utilities Smart Grid Gateway Adapter Development Kit is based on the Oracle Utilities Application Framework (OUAF). For information about using and configuring basic Framework functions, see the Oracle Utilities Application Framework documentation.

This section covers configuration of functions specific to an adapter developed with the Oracle Utilities Smart Grid Gateway Adapter Development Kit. It presents conceptual information to help you understand how the system works as well as how the various configuration options affect system functionality. Once you have an understanding of the system's capabilities, you can plan your data setup and design any customizations you want to implement.

This chapter provides details on the components and configurations required for an adapter developed with the Smart Grid Gateway Adapter Development Kit, including the following:

- **Understanding the Adapter**
- **Understanding the Adapter Processing**

Understanding the Adapter

An Oracle Utilities Smart Grid Gateway adapter can support communication with a head-end system, including measurement data and device event loading, and command messaging in support of commissioning, connect, disconnect, decommissioning, status check, and on-demand read.

The adapter uses Oracle Service Bus (OSB) and Oracle Business Process Execution Language (BPEL) to facilitate communication between Oracle Utilities Smart Grid Gateway and the head-end system.

The following functionality can be configured:

Measurement Data and Device Event Loading - data parsing and transformation via Oracle Service Bus from the smart meter format into the Oracle Utilities Service and Measurement Data Foundation unified format for measurement data and device events.

Measurement Data and Device Event Processing - configurable mapping for meter status codes and device event names to Oracle Utilities Service and Measurement Data Foundation standard values.

Commissioning Communication - business objects and BPEL processes to support the Meter Add Notification message.

Connect Communication - business objects and BPEL processes to support Initiate Connect/Disconnect and Connect/Disconnect State Change Notification messages.

Disconnect Communication - business objects and BPEL processes to support Initiate Connect/Disconnect and Connect/Disconnect State Change Notification messages.

Decommissioning Communication - business objects and BPEL processes to support Meter Removal Notification message.

On-Demand Read - business objects and BPEL processes to support Initiate Meter Read by Meter Number and Reading Changed Notification messages.

Device Status Check - business objects and Oracle BPEL processes to support the outbound Initiate Outage Detection communication, the inbound Outage Detection Event Notification communication, and the processing of Event Notifications.

Understanding the Adapter Processing

This section provides details concerning the OSB processing, BPEL Processes, and OUAF objects supplied as part of the base package. This information illustrates how the base package objects were designed, and can serve as the basis for any customizations you create as part of your implementation.

This section includes:

- **Initial Measurement Data Load and Device Events**
- **Device Communication**

Initial Measurement Data Load and Device Events

The initial measurement data load and subsequent device event processing use OSB to poll for, parse, and transform the head-payloads into the Oracle Utilities Smart Grid Gateway service format. Payloads contain measurements and meter events in some head-end specific format OSB then places each service call into a JMS queue within the Oracle Utilities applications. The JMS client consumes the entries and invokes the respective services in parallel then a service creates initial measurements with data in a common format with head-end-specific processing as needed. A second service creates device events with data in a common format

Initial Measurement and Device Events

The usage data exported from the AMI head-end system as a file in the meter format is loaded into Oracle Utilities as Initial measurement data. You can customize the usage processing by configuring the following base product OSB projects:

1. SGG-DG-CSV-BASE - contains components that are not to be changed on customer site. They implement functions specific to the CSV format processing such as validation and transformation.
2. SGG-DG-CSV-CM - allows for customization and simplifies future upgrades.
3. SGG-DG-XML-BASE - contains components that are not to be changed on customer site. They implement functions specific to the IMD and Event Upload format processing such as validation and transformation.
4. SGG-DG-XML-CM - allows for customization and simplifies future upgrades.

For additional information about the sample OSB implementation included in the Oracle Utilities Smart Grid Gateway Adapter Development Kit, see **Chapter 2: Implementing OSB Processing for One-Way Commands**.

Base Package Business Objects

The adapter development kit base package includes the following initial measurement and device event business objects:

Business Object Name	Description
DG-InitialLoadIMDInterval	Generic Initial Load IMD - Interval Used when loading customized adapter interval measurements into the system for the first time.
DG-InitialLoadIMDScalar	Generic Initial Load IMD - Scalar
DG-DeviceEventMappingLookup	Generic Device Event Mapping
DG-HeadendUOMLookup	Generic UOM Code to Standard UOM Mapping

Business Object Name	Description
DG-IntStsCodeToCondMapLookup	Generic Interval Status Mapping This is a child BO of Interval Status Code to Condition Mapping

Device Event Mapping BO

A head-end system may be capable of sending a large selection of device events. The adapter can map these events into standard Oracle Utilities Smart Grid Gateway event names and categories. To meet this end, an extendable lookup business object, DG-DeviceEventMappingLookup, can be used to hold device event names specific to the head-end system.

Device Communication

The basic communication for all business processing is essentially the same. A communication request is sent from the Oracle Utilities application to the head-end system. This request would be for a connect/disconnect, commission/decommission, measurement data, device status check, or an on-demand read. The designated BPEL process transforms the request from Oracle Utilities format to MultiSpeak format and invokes the related head-end system web service. The head-end system then returns a reply, and the BPEL process transforms the reply message back to the appropriate format so that Oracle Utilities can receive the response.

The following sections describe the key components in this processing, including:

- **Base Package Business Objects**
- **External System**
- **Outbound Message Types**
- **XAI Configuration**
- **BPEL Processes**

Base Package Business Objects

The adapter development kit base package includes the following communication business objects:

Business Object Name	Description
DG-ConnectDisconStateChgNtf	Generic - Connect/Disconnect State Change
DG-InitiateConnectDisconnect	Generic - Initiate Connect Disconnect
DG-InitiateOutageDetection	Generic - Initiate Outage Detection
DG-InitiateMeterByMeterId	Generic - Initiate Meter Read By Meter ID
DG-MeterAddNotification	Generic - Meter Add Notification
DG-MeterRemoveNotification	Generic - Meter Remove Notification
DG-OutageDetectEvtNotification	Generic - Outage Detection Event Notification
DG-ReadingChgNotification	Generic - Reading Changed Notification

Event Data Mapping

The head-end system event file format can map as follows into the business object, D1-DeviceEventMappingLookup:

Head-End System Flat File Field	Device Event Seeder BO Element	Comments
Transaction ID (from Header record)	External Source Identifier	This is the file name.
Device Identifier	External Device Identifier	
Event Name	External Event Name	
Event Creation Date/Time	Event Date/Time	
Device Type	External Device Type	This element has no real bearing on the device type within MDM/SGG. Its valid values include (although the element itself is free-form): Meter Collector Router
Service Location ID	External Service Location ID	
Communication Module Serial Number	External Communication Module Identifier	
Event Category ID	External Event Category	
Event Severity	External Event Severity	Valid values include (although the element itself is free-form): Alert Information
Status Value	External Status Value	This represents additional information that relates to the event itself.
Status Date/Time	External Status Date/Time	The date & time at which the additional information referenced above had occurred.

External System

You must create an External System for each external system to which Oracle Utilities Smart Grid Gateway will send messages. Each external system defines a set of outbound message types that will be sent to that system. Each external system outbound message type also specifies the following:

- The processing method used to send the message (Batch, XAI, or Real-time)
- The corresponding XAI senders
- Batch Control (if Processing Method is set to Batch)
- Message XSL, W3C Schema, and Response XSL (as applicable)

Outbound Message Types

Acknowledgement and response messages are sent and received validating that commands have been transmitted.

XAI Configuration

The XML Application Integration (XAI) utility allows you to configure your system to receive information from and to send information to external applications using XML. The adapter for Smart Grid Gateway uses one XAI inbound service to map device events. This is the same XAI inbound service used by the D1 application.

XAI Inbound Services

XAI inbound systems define the details of how messages are received from an external system, including the inbound communication business object (or business service or service script) to be invoked when the response message is received. As in the case of inbound communication business objects, the set of XAI inbound services you need to create is based on the types of messages the system is designed to send.

The Oracle Utilities Smart Grid Gateway Adapter Development Kit includes the following XAI inbound services:

XAI Inbound Service	Description
D1-BulkRequestHeader	Bulk Request Header Schema Name: D1-BulkRequestHeader
D1-BulkRequestUpdate	Bulk Request Update Schema Name: D1-BULKUPD
D1-BulkResponse	Bulk Response Schema Name: D1-BulkResponse
D1-DeviceEventSeeder	Used for upload of device events. Schema Name: D1-DeviceEventSeeder. The Device Event Seeder business object serves as a means of adding device events both from outside the application and from online. Its pre-processing algorithms determine the device event type - which in turn defines the device event BO that should be used to create the device event. If a device event type can't be determined, the device event is created using this BO. Such a device event can then be re-processed - and if successful, a new device event is created.
D1-InitialLoadIMD	Used for initial measurement upload. Schema Name: D1-IMDSeeder. The IMDSeeder business object is used to determine the type of initial measurement business object to instantiate when receiving usage readings from a head-end system.

XAI Inbound Service	Description
D1-DeviceStatusCheck	<p>Device Status Check</p> <p>Schema Name: D1-DeviceStatusCheck</p> <p>This service is invoked by the integration layer to instantiate a Device Status Check command.</p>
D1-InitialLoadIMD	<p>Used by OSB to instantiate an IMD</p> <p>Schema Name: D1-IMDSeeder</p> <p>This XAI Inbound service is used by OSB to instantiate an Initial Measurement Data for incoming interval usage in the Generic format.</p>
D1-RemoteConnect	<p>Remote Connect</p> <p>Schema Name: D1-RemoteConnect</p> <p>This service is invoked by the integration layer to instantiate a Remote Connect command.</p>
D1-RemoteDisconnect	<p>Remote Disconnect</p> <p>Schema Name: D1-RemoteDisconnect</p> <p>This service is invoked by the integration layer to instantiate a Remote Disconnect command.</p>
DG-ConDisconStChgNotification	<p>Initiate Connect Disconnect response.</p> <p>Schema Name: DG-ConnectDisconStateChgNtf</p> <p>Retrieves asynchronous response from Initiate Connect Disconnect command.</p>
DG-OutageDetectionEventNotification	<p>Initiate Outage Detection Response</p> <p>Schema Name: DG-OutageDetectEvtNotification</p> <p>Retrieve response from the Initiate Outage Detection Event Notification command.</p>
DG-ReadingChangedNotification	<p>Reading Changed Notification</p> <p>Schema Name: DG-ReadingChgNotification</p> <p>Notification that a device reading has changed.</p>

XAI Senders

XAI senders define the details of how messages are sent to an external system. As in the case of outbound communication business objects and outbound message types, the set of XAI senders you need to create is based on the types of messages the system is designed to accept.

BPEL Processes

These processes are responsible for performing the conversion from Oracle Utilities format to MultiSpeak 4.1 format, invoking process callouts and invoking the remote endpoint to trigger the device events.

OnDemandRead Composite Process - Invokes the remote endpoint to trigger the on-demand read event. An asynchronous reply responds to the OUAF layer when the reading arrives.

ConnectDisconnect Composite Process - Invokes the remote endpoint to trigger the connect/disconnect event. An asynchronous reply responds to the OUAF layer when confirmation of the requested event arrives.

CommissionDecommission Composite Process - Invokes the remote endpoint to trigger the commission or decommission event. After the synchronous call completes, a one of the following second business callout services is invoked to determined if the related “received” or “completed” callout should be executed:

- isExecutingCommissionReceived-Callout
- isExecutingCommissionCompleted-Callout
- isExecutingDecommissionReceived-Callout
- isExecutingDecommissionCompleted-Callout

DeviceStatusCheck Composite - Invokes the remote endpoint to trigger the initiate outage detection event. An asynchronous reply responds to the OUAF layer when confirmation of the requested event arrives.

ProcessCallout Composite - This business callout provides a point at which customers and implementers can incorporate custom business logic and transformations. This composite includes the WSDLs and processing logic for all of the MultiSpeak processes. The default implementation of each method is a direct return of the input.

For additional information about the BPEL processes included in the Oracle Utilities Smart Grid Gateway Adapter Development Kit, see **Chapter 3: Configuring BPEL Processing for Two-Way Commands**.

Web Services

These web services are all defined in the head-end system. The WSDLs were added to a Meta Data Storage (MDS) layer in OUAF and all references to the WSDL point to this MDS location. These web services have HTTP security by default. You may need to modify the security as a part of your implementation.

Web Service	Related BPEL Process	Description
CB_ServerService	ConnectDisconnect	<p>This web service defines the return interface, the means by which the status is returned to the calling system.</p> <p>This web service is only be invoked by the head end system, not OUAF. Only the CDStateChangeNotification web method is implemented in the composite.</p> <p>The endpointURI format is: http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/Generic Adapter/ConnectDisconnect/CB_ServerService</p>
CB_Server	OnDemandRead	<p>This web service defines the return interface, the means by which the reading is returned to the calling system.</p> <p>This web service is only be invoked by the head end system, not OUAF. Only the ReadingChangedNotification web method is implemented in the composite.</p> <p>The endpointURI format is: http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/Generic Adapter/OnDemandRead/CB_Server</p>
OA_ServerService	DeviceStatusCheck	<p>This web service defines the asynchronous return for InitiateOutageDetectionEventRequest for solicited responses.</p> <p>This web service is only be invoked by the head end system, not OUAF. Only the ODEventNotification, PingURL, and GetMethods web methods are implemented in the composite.</p> <p>The endpointURI format is: http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/Generic Adapter/DeviceStatusCheck/OA_ServerService</p>

For additional information about the web services included in the Oracle Utilities Smart Grid Gateway Adapter Development Kit, see **Chapter 3: Configuring BPEL Processing for Two-Way Commands**.

Configuring a Head-End System

This section outlines the configuration required for the Oracle Utilities Smart Grid Gateway to communicate with the smart meters. This includes:

- **XAI Inbound Services**
- **XAI Senders**
- **Outbound Message Types**
- **External System**
- **Service Provider**
- **Processing Methods for Service Provider**

XAI Inbound Services

XAI inbound services define the details of how messages are received from an external system. This includes incoming usage and device events, as well as messages sent from the head-end system in response to a command request.

The following XAI Inbound Services must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating XAI inbound services.

XAI Inbound Service Name	Description	Schema Type	Schema Name
D1-BulkRequestHeader	Bulk Request Header	Business Object	D1-BulkRequestHeader
D1-BulkRequestUpdate	Bulk Request Update	Business Object	D1-BULKUPD
D1-BulkResponse	Bulk Response	Business Object	D1-BulkResponse
D1-DeviceEventSeeder	Device Event Seeder	Business Object	D1-DeviceEventSeeder
D1-DeviceStatusCheck	Device Status Check	Business Object	D1-DeviceStatusCheck
D1-InitialLoadIMD	IMD Seeder	Business Object	D1-IMDSeeder
D1-PayloadErrorNotif	Payload Error Notification	Business Object	D1-PayloadErrorNotif
D1-PayloadStatistics	Payload Statistics	Business Object	D1-PayloadStatistics
D1-PayloadSummary	Payload Summary	Business Object	D1-PayloadSummary
DG-ConDisconStChgNotification	Initiate Connect Disconnect Response	Business Object	DG- ConnectDisconStateChgNtf
DG- OutageDetectionEventNotification	Initiate Outage Detection Response	Business Object	DG- OutageDetectEvtNotification
DG-ReadingChangedNotification	Reading Changed Notification	Business Object	DG-ReadingChgNotification

XAI Senders

XAI senders define the details of how messages are sent to an external system, such as messages containing device command requests.

The following XAI Senders must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating XAI senders.

XAI Sender	Description	Operation	Service
DG-INTOUTDET	Initiate Outage Detection Request	InitiateOutageDetectionEventRequest	OD_ServerService
DGXAIsender	Used for Meter Decommission	MeterDecommission	CommissionDecommission Service

Note: The following apply to all of the above XAI senders:

Main Tab:

- **Invocation Type:** Real-time
- **XAI Class:** RTHTTPSNDR (Sender routes message via HTTP real-time)
- **MSG Encoding:** UTF-8 message encoding

Context Tab:

- **HTTP Header:** SOAPAction:<OPERATION>
- **HTTP Login User:** <USER_ID>
- **HTTP Login Password:** <PASSWORD>
- **HTTP Method:** POST
- **HTTP URL 1:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/Generic/<SERVICE>

where:

- **<OPERATION>**: the operation performed by the XAI Sender (see Operation column in the table above)
- **<USER_ID>**: the user ID used to log into WebLogic Enterprise Manager
- **<PASSWORD>**: the password used to log into WebLogic Enterprise Manager
- **<EM_SERVER_IP>**: the machine name or IP address of server where the WebLogic Enterprise Manager is installed
- **<EM_SERVER_PORT>**: the port where the WebLogic Enterprise Manager is installed
- **<SERVICE>**: the service invoked by the XAI Sender (see Service column in the table above)

Outbound Message Types

Outbound message types define specific types of messages sent to an external system, such as messages containing device command requests.

The following outbound message types must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating outbound message types.

Outbound Message Type	Description
DGDECOMMOMT	Meter Decommission

Note: The following apply to all of the outbound message types:

- **Business Object:** D1-OutboundMessage (Outbound Message)
- **Priority:** Priority 50

External System

External systems represent external applications with which the Smart Grid Gateway will exchange messages or data. In the case of the Smart Grid Gateway adapters, external systems represent the head-end systems with which the adapters communicate.

An external system that represents the head-end system must be present in your system. If this are not present in your configuration, add it, along with the following Outbound Message Types. Refer to the Oracle Utilities Application Framework documentation for more information about creating external systems.

External System:

- **External System:** Generic
- **Description:** Generic
- **Outbound Message Types:**

The following apply to all of the above outbound message types:

- **Processing Method:** Real-time
- **Message XSL:** DG-Request.xsl
- **Response XSL:** DG-Response.xsl

Service Provider

Service providers represent external entities that serve various roles relative to the application, including head-end systems, billing systems to which the application sends bill determinant data, market participants in a deregulated environment, outage management systems that receive meter event data from the application, or other parties that require or provide information to the system. The head-end systems that collect and send measurement data and meter events to the application are defined as service providers.

A service provider that represents the head-end system must be present in your system. If this are not present in your configuration, add it. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about creating service providers.

Service Provider - Generic:

- **Service Provider:** Generic
- **Description:** Generic

- **External Reference ID:** Generic
- **External System:** Generic
- **Our Name/ID in Their System:**
- **AMI Device ID Type:** Internal Meter Number
- **AMI Measuring Component ID Type:** Channel ID

Processing Methods for Service Provider

Processing methods define the format or means by which a service provider receives and/or sends data from and/or to the application, including bill determinants, usage data, or device events. Processing methods are also used to define how to create information internal to the application such as initial measurement data and device events. Processing methods can also be used to define how command requests are sent to the head-end system.

The following types of processing methods must be configured for the head-end system service provider. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about configuring processing methods.

Initial Measurement Creation

Initial measurement creation processing methods define the business objects used to create initial measurements. The IMD Seeder XAI Inbound Service uses this processing method to determine which type of initial measurement business object to instantiate when receiving usage from the head-end system.

Processing Method - Initial Measurement Creation

- **Service Provider:** Generic
- **Processing Role:** Initial Measurement Creation
- **Description:** Generic Initial Measurement Creation
- **Status:** Active
- **Default Processing Method:**
 - **Business Object:** DG-InitialLoadIMDInterval (Generic Initial Load IMD - Interval)
- **Override Processing Method:**
 - **Measuring Component Type:** Electric Residential kWh Scalar
 - **Business Object:** DG-InitialLoadIMDScalar (Generic Initial Load IMD - Scalar)

Device Event Mapping

Device event mapping processing methods define how head-end-specific device events are mapped to standard device event names. The Device Event Seeder XAI Inbound Service uses this processing method to determine which type of device event business object to instantiate when receiving device events from the head-end system.

Processing Method - Device Event Mapping

- **Service Provider:** Generic
- **Processing Role:** Device Event Mapping
- **Description:** Generic Device Event Mapping
- **Status:** Active
- **Default Processing Method:**
 - **Business Object:** DG-DeviceEventMappingLookup (Generic Device Event Mapping)

- **Override Processing Method:** based on implementation-specific requirements

UOM Mapping

UOM mapping processing methods define how head-end-specific unit of measure (UOM) codes are mapped to standard UOM codes. This processing method is used to determine how to map head-end system UOM codes to standard UOM codes when receiving usage from the head-end system.

Processing Method - UOM Mapping

- **Service Provider:** Generic
- **Processing Role:** UOM Mapping
- **Description:** Generic UOM Mapping
- **Status:** Active
- **Default Processing Method:**
 - **Business Object:** DG-HeadendUOMLookup (Generic UOM Code to Standard UOM Mapping)
- **Override Processing Method:** based on implementation-specific requirements

Commands

Command processing methods define how command requests are sent to a head-end system. More specifically, they define the type of outbound communication business object to create for each type of command, and the outbound message type to send to the head-end system.

The following types of command processing methods can be configured for the head-end system service provider, based on the requirements of each implementation.

Device Commission

- **Service Provider:** Generic
- **Processing Role:** Device Commission
- **Description:** Generic Device Commission
- **Status:** Active
- **Processing Method:**
 - **Default Business Object:** DG-MeterAddNotification (Generic Meter Add Notification)
 - **Default Outbound Message Type:** Generic Commissioning Outbound Message Type

Device Decommission

- **Service Provider:** Generic
- **Processing Role:** Device Decommission
- **Description:** Generic Device Decommission
- **Status:** Active
- **Processing Method:**
 - **Default Business Object:** DG-MeterRemoveNotification (Generic - Meter Remove Notification)
 - **Default Outbound Message Type:** Meter Decommissioning

Device Status Check

- **Service Provider:** Generic
- **Processing Role:** Device Status Check
- **Description:** Initiate Outage Detection
- **Status:** Active
- **Processing Method:**
 - **Default Business Object:** DG-InitiateOutageDetection (Generic - Initiate Outage Detection)
 - **Default Outbound Message Type:** Initiate Outage Detection Request

On-Demand Read (Scalar)

- **Service Provider:** Generic
- **Processing Role:** On-Demand Read (Scalar)
- **Description:** On-Demand Read Generic (Scalar)
- **Status:** Active
- **Processing Method:**
 - **Default Business Object:** DG-InitiateMeterByMeterId (Generic - Initiate Meter Read by Meter ID)
 - **Default Outbound Message Type:** Initiate Meter Read by Meter ID

Remote Connect

- **Service Provider:** Generic
- **Processing Role:** Remote Connect
- **Description:** Generic Initiate Connect
- **Status:** Active
- **Processing Method:**
 - **Default Business Object:** DG-InitiateConnectDisconnect (Generic - Initiate Connect Disconnect)
 - **Default Outbound Message Type:** Connect Device

Remote Disconnect

- **Service Provider:** Generic
- **Processing Role:** Remote Disconnect
- **Description:** Generic Initiate Disconnect
- **Status:** Active
- **Processing Method:**
 - **Default Business Object:** DG-InitiateConnectDisconnect (Generic - Initiate Connect Disconnect)
 - **Default Outbound Message Type:** Initiate Connect Disconnect

Configuring Extendable Lookups

This section outlines some of the extendable lookups that must be configured for use with the customized adapter. These include:

- **Device Event Mapping**
- **Head-End UOM Code to Standard UOM Mapping**
- **Interval Status Code to Condition Mapping**

Refer to the Oracle Utilities Application Framework documentation for more information about working with extendable lookups.

Device Event Mapping

The Device Event Mapping extendable lookup is used to determine which type of device event business object to instantiate when receiving device events from the head-end system.

Each value defined for the Device Event Mapping extendable lookup should include the following:

- **Head-End System Event Name:** The event name used by the head-end system.
- **Description:** A description of the device event
- **Status:** The status of the lookup value (can be Active or Inactive)
- **Standard Event Name:** The standard event name for device events of this type, from the “Standard Event Name” extendable lookup.

Example: The “Tampering” device event name could be mapped to the “Device Tampering” standard device event name as follows:

- **Head-End System Event Name:** Tampering
- **Description:** Tampering Detected
- **Status:** Active
- **Standard Event Name:** Device Tampering

Head-End UOM Code to Standard UOM Mapping

Usage received from the head-end system may use utility-specific unit of measures (UOMs). These custom UOMs must be mapped to standard UOM codes. The head-end system UOM Code to Standard UOM Mapping extendable lookup is used to determine how to map head-end system UOM codes to standard UOM codes when receiving usage from the head-end system.

Each value defined for the head-end system UOM Code to Standard UOM Mapping extendable lookup should include the following:

- **Head-End Unit of Measure:** The unit of measure code used by the head-end system
- **Unit of Measure:** The unit of measure defined in the system. See *Defining Units of Measure* in the *Oracle Utilities Service and Measurement Data Foundation User's Guide* for more information about creating UOM codes for use with Oracle Utilities Smart Grid Gateway.
- **Description:** A description of the unit of measure code.
- **Status:** The status of the lookup value (can be Active or Inactive)

Example: The head-end system “KWH” unit of measure code could be mapped to the “Kilowatt Hours” standard UOM code as follows:

- **Unit of Measure:** Kilowatt Hours
- **Head-end UOM:** kWh

- **Description:** Head-End System Kilowatt Hours

Interval Status Code to Condition Mapping

Interval usage received from the head-end system can include interval status codes that indicate the status or condition of the interval value. These interval status codes must be mapped to standard condition codes in the system. The Generic Interval Status Code to Condition Mapping extendable lookup is used to determine how to map head-end system interval status codes to standard status codes when receiving usage from the head-end system.

Each value defined for the Generic Interval Status Code to Condition Mapping extendable lookup should include the following:

- **Interval Status:** The head-end system interval status code
- **Description:** A description of the interval status code.
- **Status:** The status of the lookup value (can be Active or Inactive)
- **Condition:** The condition code to which the interval status code is to be mapped, from the Measurement Condition extendable lookup.

Example: The head-end system “Missing” interval status code could be mapped to the “Missing” condition code as follows:

- **Interval Status:** Missing
- **Condition:** Missing
- **Description:** Head-End System Missing

Extending the Adapter Development Kit

The Oracle Utilities Smart Grid Gateway Adapter Development Kit supports a number of commands, including:

- Commission Device
- Decommission Device
- Device Status Check
- On-Demand Read
- Remote Connect
- Remote Disconnect

The Adapter Development Kit can be extended to support additional commands provided by smart meters. See **Creating Custom Commands** on page 9-26 of the *Oracle Utilities Smart Grid Gateway Configuration Guide* for more information about adding commands to the adapter.

Appendix

Glossary

This glossary provides definitions of commonly used terms.

Command Effective Date/Time

The date and time when a device command becomes effective.

Command Expiration Date/Time

The date and time when a device command expires.

Commissioning

A command issued to establish communication between a device and the head-end system. The goal is to ensure connectivity has been established with the device, that any information needed to communicate with the meter has been defined in both Oracle Utilities Smart Grid Gateway and the head end system, and the meter will begin capturing usage and events.

Decommissioning

A command issued to inform the head-end system when a meter needs to be removed from a service point, so that no further reads or events will arrive from the meter. Decommissioning is invoked when a meter must be removed or deactivated. The goal is to stop any communication between the device and the head-end system.

Head-End System

A system that collects measurement data and meter events for eventual submission to the application. Many devices can communicate to the application through a single head-end system. A utility may have numerous head-end systems through which they communicate with devices.

On-Demand Read

A request for the most up-to-date reading from a particular meter. It is not guaranteed to return immediately; it could require a person to manually read the meter. The purposes are to check the meter's operational status and/or obtain a more recent reading than is currently available.

Payload

An upload component which contains measurements and meter events in a format specific to the

head-end. Payloads are part of the initial upload of measurement data.

Remote Connect

A command issued when a meter needs to be connected at a service point.

Index

A

Adapter 4-2
 processing 4-3

B

BPEL Processes 4-8
 CommissionDecommission 4-8
 ConnectDisconnect 4-8
 OnDemandRead 4-8
 ProcessCallout 4-8
Business Objects 4-3, 4-4
 DG-ConnectDisconStateChgNtf 4-4
 DG-DeviceEventMappingLookup 4-3
 DG-HeadendUOMLookup 4-3
 DG-InitialLoadIMDInterval 4-3
 DG-InitialLoadIMDScalar 4-3
 DG-InitiateConnectDisconnect 4-4
 DG-InitiateMeterByMeterId 4-4
 DG-InitiateOutageDetection 4-4
 DG-IntStsCodeToCondMapLookup 4-4
 DG-MeterAddNotificationMultiSp 4-4
 DG-MtrRmvNotifMultiSpeak 4-4
 DG-OutageDetectEvtNotification 4-4
 DG-ReadingChgNotification 4-4

C

Communication Processes 4-4

D

Device Event Mapping 4-4

E

External System 4-5

G

General Configuration 4-1

I

Inbound Services 4-6
 D1-BulkRequestHeader 4-6
 D1-BulkRequestUpdate 4-6
 D1-BulkResponse 4-6

D1-DeviceEvent 4-6
D1-DeviceStatusCheck 4-7
D1-InitialLoadIMD 4-6, 4-7
D1-RemoteConnect 4-7
D1-RemoteDisconnect 4-7
DG-ConDisconStChgNotification 4-7
DG-OutageDetectionEventNotification 4-7
DG-ReadingChangedNotification 4-7

Initial Measurement 4-3

O

Oracle Service Bus 4-3
OUAF 4-4
Overview 1-1

W

Web Services 4-9
 CB_Server 4-9
 CB_ServerService 4-9

X

XAI
 configuration 4-6
 inbound services 4-6
 senders 4-8

