

Oracle Utilities Smart Grid Gateway Adapter for Landis + Gyr

Administrative User Guide

Release 2.2.0

E80270-01

December 2016

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

| | |
|---|-----------|
| Landis+Gyr Adapter Overview..... | 4 |
| Landis+Gyr Adapter Processing..... | 6 |
| Initial Measurement Data and Device Event Loading..... | 6 |
| Initial Measurements..... | 6 |
| Device Events..... | 8 |
| Base Package Business Objects..... | 11 |
| Device Communication..... | 11 |
| Communication Flows..... | 11 |
| Device Communication Base Package Business Objects..... | 12 |
| External System..... | 14 |
| Outbound Message Types..... | 14 |
| Inbound / Outbound Service Configuration..... | 14 |
| BPEL Processes..... | 16 |
| Configuring a Landis+Gyr Head-End System..... | 18 |
| Inbound Web Services..... | 18 |
| Message Senders..... | 19 |
| Outbound Message Types..... | 20 |
| External System..... | 20 |
| Service Provider..... | 21 |
| Processing Methods..... | 22 |
| Configuring Landis+Gyr Extendable Lookups..... | 24 |
| Extending the Landis+Gyr Adapter..... | 26 |
| The Landis+Gyr Test Harness..... | 27 |
| Test Harness Design..... | 27 |
| Locating the WSDL for the Test Harness..... | 28 |
| Web Services..... | 28 |
| General Services..... | 28 |
| Locate Meter Services..... | 29 |
| Meter Administration Services..... | 30 |
| Meter Attribute Administration Services..... | 33 |
| Landis+Gyr Interval Data Mapping..... | 37 |
| Non-Interval Usage with Additional Fields..... | 37 |
| XML 'Plain' XML Format..... | 37 |
| Non-Interval Usage to 'Plain' XML Mapping..... | 39 |
| 'Plain' XML to IMD Mapping..... | 39 |
| Non-Interval 'Plain' XML to IMD Mapping..... | 40 |
| Mapping Additional Fields..... | 41 |

Chapter 1

Landis+Gyr Adapter Overview

The Oracle Utilities Smart Grid Gateway Adapter for Landis+Gyr supports communication with the Landis+Gyr Command Center software version 6.0, including measurement data and device event loading, and command messaging in support of commissioning, connect, disconnect, decommissioning, and on-demand read. The following table describes the attributes of the adapter:

| Attribute | Details |
|------------------------------|---|
| Currently Supported Version | Gridstream Command Center 6.5 |
| Smart Meter Command Format | MultiSpeak v3.1 |
| Bulk Usage/Event Data Format | California Metering Exchange Protocol (CMEP) |
| Market(s) | North America, portions of Asia Pacific, Sweden, Australia, New Zealand, and Latin America. |
| Commodities Supported | Electricity, Gas, Water |
| Architecture | PLC and RF |

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 Landis+Gyr Command Center.

The following functionality is included:

Measurement Data and Device Event Loading - data parsing and transformation via Oracle Service Bus from Landis+Gyr 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 Landis+Gyr status codes and device event names to Oracle Utilities Service and Measurement Data Foundation standard values.

Smart Meter Command Processing - sending/receiving messages to/from the Landis+Gyr application to initiate smart meter commands from Oracle Utilities Smart Grid Gateway. The Landis+Gyr adapter supports the following types of commands and communications:

- **Device Status Check**- business objects and BPEL processes to support issuing device status check commands.
- **Meter Commissioning** - business objects and BPEL processes to support issuing meter commissioning commands (including registration and installation commands).
- **Meter Decommissioning**- business objects and BPEL processes to support issuing meter decommissioning commands.

- **Meter Retirement** - business objects and BPEL processes to support issuing meter retire (deregistration) commands.
- **On-Demand Read** - business objects and BPEL processes to support issuing on-demand read commands.
- **Remote Connect** - business objects and BPEL processes to support issuing remote connect commands.
- **Remote Disconnect** - business objects and BPEL processes to support issuing remote disconnect commands

Chapter 2

Landis+Gyr Adapter Processing

This section provides details concerning the OSB processing, BPEL Processes, 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.

Initial Measurement Data and Device Event Loading

The initial measurement data load and 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 Measurements

The usage data exported from the AMI head-end system as a file in Landis+Gyr format is loaded into Oracle Utilities as initial measurement data. The following OSB projects, delivered in the base product, help manage the usage processing:

1. **SGG-D3-USAGE-BASE** - contains components responsible for “actual” processing of incoming data. It should not be modified during configuration. This can be upgraded without affecting the customization and environment settings added to SGG-D3-USAGE-CM.
2. **SGG-D3-USAGE-CM** allows for customization and simplifies future upgrades.

When importing non-interval usage data, separate initial measurements can be created for difference measurement types. For instance, if the data includes Power Factor or Volt data, separate initial measurements are created for each of these. See [Non-Interval 'Plain' XML to IMD Mapping](#) for more information about specific units of measure that trigger the creation of separate initial measurements.

The runtime configuration settings for the SGG-D3-USAGE-CM project are stored in the EnvironmentSettings.xq XQuery file. You can use this file to adjust initial measurement data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRawIMD element.

The following table describes the elements included in the EnvironmentSettings.xq file:

| Element | Description | Valid Values |
|--------------------------------|--|---------------|
| populateRawIMD | Determines if the initial measurement data is populated as raw data. | true false |
| callPreProcessing | Determines if the preprocessing proxy service is called. | true false |
| callPostProcessing | Determines if the postprocessing proxy service is called. | true false |
| destinationRootElementInterval | Holds the name of inbound web service for the interval IMD seeder. | |
| destinationRootElementScalar | Holds the name of inbound web service for the scalar IMD seeder. In most cases it is the same as destinationRootElementInterval. | |
| modifyResultXMLInput | Specifies the name of an XQuery document (without the "xq" extension) used to map additional fields from the "plain" XML format to the result XML format sent as initial measurement data. See Mapping Additional Fields for more information. | |
| dateTimelnUTC | Indicates whether the Landis+Gyr system is sending date/time information in UTC (true) or local time of the device (false). If not provided the default behavior will be local time of the device. | true false |
| publishServices/service | Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker). | |
| filterUsage | Determines if usage should be filtered. | true false |

Publishing Initial Measurement Data

The Landis+Gyr adapter can be configured to publish initial measurement data for use in Oracle DataRaker or other external systems. Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

Publishing measurement data to Oracle DataRaker is supported by the following components provided with the SGG-D3-USAGE-CM OSB project:

- The **DataRakerBusinessService** business service is used to publish data to a specified JMS queue (defined as an Endpoint URI), from which the external system can receive the data. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Initial measurement data is published in the “native” initial measurement data format (the format of the initial measurement seeder business object).

I

Filtering Initial Measurement Data

The Landis+Gyr adapter can be configured to filter initial measurement data passed into Oracle Utilities Smart Grid Gateway and Meter Data Management. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterUsage>true</filterUsage>
```

When filtering is enabled, only measurements whose <externalUOM> matches one of the values defined in the **Landis+Gyr UOM Code to Standard UOM Mapping** extendable lookup (D3-HeadendUOMLookup) are passed into the system for processing.

NOTE: Filtering of scalar initial measurement data is not supported in the Landis+Gyr adapter.

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 for the SGG-D3-USAGE-CM project. 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.

Prioritized Initial Measurement Processing

The Landis+Gyr adapter prioritizes processing of initial measurements created from smart meter commands and/or completion events by setting the Execution Method flag in these types of initial measurements to “Real Time” (D1RT). In addition, initial measurements received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing.

See **Initial Measurement Data Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Device Events

The device event data exported from the head-end system as a file in Landis+Gyr format is loaded into Oracle Utilities as a Device Event. One of your configuration tasks is to customize the device events processing.

The required functionality is delivered in the base product as two OSB projects:

1. **SGG-D3-EVENT-BASE** containing components responsible for "actual" processing of incoming data. It can be upgraded in future without affecting the customization and environment settings that done in SGG-D3-EVENT-CM project.
2. **SGG-D3-EVENT-CM** allows the customization and simplifies the future upgrades.

The runtime configuration settings for the SGG-D3-EVENT-CM project are stored in the EnvironmentSettings.xq XQuery file. You can use this file to adjust device event data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRaw element.

The following table describes the elements included in the EnvironmentSettings.xq file:

| Element | Description | Valid Values |
|-------------------|---|---------------|
| populateRaw | Determines if the device event data is populated as raw data. | true false |
| callPreProcessing | Determines if the preprocessing proxy service is called. | true false |

| Element | Description | Valid Values |
|-------------------------|--|---------------|
| callPostProcessing | Determines if the postprocessing proxy service is called. | true false |
| destinationRootElement | Holds the name of inbound web service for the device event seeder. | |
| modifyResultXMLInput | Specifies the name of an XQuery document (without the ".xq" extension) used to map additional fields from the "plain" XML format to the result XML format sent as device event data. See Mapping Additional Fields for more information. | |
| dateTimelnUTC | Indicates whether the Landis+Gyr system is sending date/time information in UTC (true) or local time of the device (false). If not provided the default behavior will be local time of the device. | true false |
| publishServices/service | Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker). | |
| filterEvents | Determines if events should be filtered. | true false |

Publishing Events

The Landis+Gyr adapter can be configured to publish device events for use in Oracle DataRaker or other external systems. Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

Publishing device events to Oracle DataRaker is supported by the following components provided with the SGG-D3-EVENT-CM OSB project:

- The **DataRakerBusinessService** business service is used to publish data to a specified JMS queue (defined as an Endpoint URI), from which the external system can receive the data. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Device event data is published in the “native” device event data format (the format of the device event seeder business object).

Filtering Events

The Landis+Gyr adapter can be configured to filter device events passed into Oracle Utilities Smart Grid Gateway and Meter Data Management. Filtering data is enabled by setting the <filterEvents> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterEvents>true</filterEvents>
```

When filtering is enabled, only device events whose <externalEventName> matches one of the values defined in the **Landis+Gyr Device Event Mapping** extendable lookup (D3-DeviceEventMappingLookup) are passed into the system for processing.

Subscribing to Real-Time Device Events

The Landis+Gyr Command Center provides the ability to "subscribe" to device events from within their online interface. This is done by creating a subscriber in Command Center with an associated callback URL as well as a list of events types that subscriber is interested to receive. When an event that is subscribed to occurs it is sent to the callback URL in real time.

Within L+G Events are configured with one of three "alarm" settings. These settings determine how often the events will be sent to subscribers:

1. Alarm: immediately delivered from the meter
2. Advisory: sent based upon a delivery schedule
3. Log Only: sent only upon request (not applicable for our implementation real time event processing)

Command Center will communicate the events using a CIM format that describes the message as a noun/verb combination. The details of the event itself will be contained within a "payload" element of the standard structure. The payload will be formatted using the EndDeviceEvent message structure. This message identifies device events using a CIM 4-part category number. These numbers are four period separated numbers that will describe the type of device and the event. For example: 3.33.1.257 is for "Tamper attempt suspected".

- Segment 1: End Device event domain code (e.g. 3. meter/10. collector/11. router/12. HAN device)
- Segment 2: End Device Event Domain Part Codes (e.g. 1. Access/2. Battery)
- Segment 3: End Device Event Type Codes (e.g. 1. Alarm/2. Alarm Mgt)
- Segment 4: End Device Event Index (e.g. 1. Abort/2. Access Attempt)

Refer to the Landis+Gyr documentation for details about the CIM Category Numbers. CIM Category Numbers must be mapped to standard device event names using the Landis+Gyr Device Event Mapping extendable lookup.

SGG receives these messages through a BPEL composite that saves the incoming request as a file to be picked up by OSB.

The **AMIEventSubscriber** composite is responsible for receiving the event messages based on subscriptions defined in the L+G Command Center. The callback URL configured for the subscription in the Command Center should point to this BPEL composite.

The following OSB projects parse individual device events from the message and perform the validation and mapping of the information to the Device Event Seeder Format.

1. **SGG-D3-CIM-EVENT-BASE** contains components responsible for "actual" processing of incoming data. It can be upgraded in future without affecting the customization and environment settings that done in SGG-D3-CIM-EVENT-CM project.
2. **SGG-D3-CIM-EVENT-CM** allows the customization and simplifies the future upgrades.

The runtime configuration settings for the SGG-D3-CIM-EVENT-CM project are stored in the EnvironmentSettings.xq XQuery file. You can use this file to adjust device event data processing. For example, if you want to load raw data you would specify "true" for the content of the populateRaw element.

The following table describes the elements included in the EnvironmentSettings.xq file:

| Element | Description | Valid Values |
|-------------------------|---|---------------|
| populateRaw | Determines if the device event data is populated as raw data. | true false |
| callPreProcessing | Determines if the preprocessing proxy service is called. | true false |
| callPostProcessing | Determines if the postprocessing proxy service is called. | true false |
| destinationRootElement | Holds the name of inbound web service for the device event seeder. | |
| publishServices/service | Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker). | |

| Element | Description | Valid Values |
|--------------|--|--------------|
| filterEvents | Determines if events should be filtered. | true |
| | | false |

Processing statistics are gathered for any real time events that are received (even if there is just one event in the message) in the same manner as device events received via the flat-file interface.

Prioritized Device Event Processing

The Landis+Gyr adapter prioritizes processing of device events created from smart meter commands and/or completion events by setting the Execution Method flag in these types of device events to “Real Time” (D1RT). In addition, device events received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing.

See **Device Event Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Base Package Business Objects

The Landis+Gyr adapter base package includes the following initial measurement business objects:

| Business Object Name | Description |
|---------------------------|--|
| D3-InitialLoadIMDInterval | Landis+Gyr Initial Load IMD - Interval Used when loading Landis+Gyr interval measurements into the system for the first time. |
| D3-InitialLoadIMDScalar | Landis+Gyr Initial Load IMD - Scalar |

Device Communication

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

Communication Flows

The table below lists the communications created for each Landis+Gyr command:

| Command | Outbound Communication | Inbound Communication | Completion Event |
|---|---|---|--|
| Remote Connect (This command has sub-commands) | Initiate MR by Mtr Num Initiate Connect Disconnect | Reading Changed Notification Connect Disconnect State Changed Notification | Connect Device Create IMD Completion Event |
| Remote Disconnect (This command has sub-commands) | Initiate Connect Disconnect Initiate MR by Mtr Num | Connect Disconnect State Changed Notification Reading Changed Notification | Disconnect Device Create IMD Completion Event |
| Device Commissioning (Registration) | L+G Add Meter to Inventory | | Device Commissioning |
| Device Commissioning (Installation) | L+G Meter Exchange Notification | | Device Commissioning |

| Command | Outbound Communication | Inbound Communication | Completion Event |
|------------------------------------|--|--|-----------------------------|
| Device Decommissioning | Meter Remove Notification | | Device Decommissioning |
| Device Deregistration | L+G Meter Retire Notification | | Device Deregistration |
| On-Demand Read (Scalar) | Initiate MR by Mtr Num | Reading Changed Notification | Create IMD Completion Event |
| On-Demand Read (Scalar) - CIM | CIM Meter On Demand Read (Scalar) | CIM Meter On Demand Read Response | Create IMD Completion Event |
| On-Demand Read (Interval) | Initiate MR by Mtr Num | Reading Changed Notification | Create IMD Completion Event |
| On-Demand Read (Interval) - CIM | CIM Meter On Demand Read (Interval) | CIM Meter On Demand Read Response | Create IMD Completion Event |
| Device Status Check | CIM Ping | CIM Ping Response | |
| Demand Reset | Schedule Demand Reset (Multispeak) | Schedule Demand Reset Response (Multispeak) | Create IMD Completion Event |

Device Registration Commission Commands

Landis+Gyr device commission commands can be used to “register” the device and notify the L+G head-end system that meters have been added to inventory. Device commission commands of this type have the “Registration-Only Mode” flag set to “Yes”. An Enter algorithm on the “Commission Ready” state evaluates the Registration-Only Mode of the command and if set to “Yes”, the command skips the default “Waiting for Measurement” state and is transitioned to the “Execute Completion Event” state, and an activity log entry is created.

Only the device registration request is sent to the head-end system for device commission commands of this type.

Device registration commands are typically created when new devices are added to inventory in an asset management system such as Oracle Utilities Operational Device Management.

Device Installation Commission Commands

Landis+Gyr device commission commands can be used to notify the L+G head-end system that meters have been installed or exchanged. An Enter algorithm on the “Commission Ready” state evaluates the “Is Installation Check Unnecessary” flag of the command and if set to “False”, the algorithm creates an “L+G Meter Exchange Notification” outbound communication and sends an installation notification to the head-end system.

Device Deregistration Commands

The Landis+Gyr adapter supports the Device Deregistration command (based on the D1-DeviceDeregistration business object). This command sends a communication that deregisters the device in the head-end system, and is most often used when retiring a device. The specific message sent is defined for the Device Deregistration processing role for the L+G head-end system service provider.

Device deegistration commands are typically created when devices are retired in an asset management system such as Oracle Utilities Operational Device Management.

Device Communication Base Package Business Objects

The Landis+Gyr Adapter base package includes the following communication business objects:

| Business Object Name | Description |
|-------------------------------|---|
| D3-AddMeterToInventoryMultiSp | L+G Add Meter to Inventory (MultiSpeak) |
| D3-CIMGetLPData | CIM Meter On Demand Read (Interval) |
| D3-CIMMeterOnDemandRead | CIM Meter On Demand Read (Scalar) |
| D3-CIMMeterReadingResponse | CIM Meter On Demand Read Response |

| Business Object Name | Description |
|--------------------------------|---|
| D3-CIMPing | CIM Ping |
| D3-CIMPingResponse | CIM Ping Response |
| D3-ConnectDisconStateChgNtf | Connect Disconnect State Changed Notification |
| D3-InitiateConnectDisconnect | Initiate Connect Disconnect |
| D3-InitiateMRByMtrNbr | Initiate Meter Read By Meter (MultiSpeak) |
| D3-MeterAddNotificationMultiSp | Meter Add Notification (MultiSpeak) |
| D3-MeterExNotificationMultiSp | L+G Meter Exchange Notification |
| D3-MeterRetireNotification | L+G Meter Retire Notification |
| D3-MtrRmvNotifMultiSpeak | Meter Remove Notification (MultiSpeak) |
| D3-ReadingChgNotification | Reading Changed Notification |
| D3-ScheduleDemandReset | Schedule Demand Reset (Multispeak) |
| D3-ScheduleDemandResetResponse | Schedule Demand Reset Response (Multispeak) |
| D3-ReadingChgNotification | Reading Changed Notification |

Landis+Gyr Event Data Mapping

The Landis+Gyr event file format maps as follows into the business object, D1-DeviceEventMappingLookup:

| Landis+Gyr 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 | <p>This element has no real bearing on the device type within MDM/SGG. Its valid values include (although the element itself is free-form):</p> <p>Meter</p> <p>Collector</p> <p>Router</p> |
| 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 | <p>Valid values include (although the element itself is free-form):</p> <p>Alert</p> <p>Information</p> |
| 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 or Real-time)
- The corresponding message senders
- Batch Control (if Processing Method is set to Batch)
- Message XSL, W3C Schema, and Response XSL (as applicable)

Outbound Message Types

Acknowledgment and response messages are sent and received validating that commands have been transmitted. These notifications are based on the following outbound message types.

| Outbound Message Type | Description |
|-----------------------|--|
| D3-ADDMTRINV | Add Meter to Inventory |
| D3-COMMS | Commission Device |
| D3-CONNECT | Connect Device |
| D3-DECOMMS | Decommission |
| D3-DEMRESET | Demand Reset |
| D3-DERDEV | Deregister Device |
| D3-DISCONN | Disconnect Device |
| D3-DVCSTCHK | Device Status Check |
| D3-INITMRN | Initiate Meter Read by Meter Number |
| D3-INITMTR | Initiate Meter Read by Meter Number |
| D3-MTRADDNOT | Meter Add Notification Outbound Message Type |
| D3-MTRESX | Meter Exchange Notification OB MSG |
| D3-MTRRMV | Meter Remove Notification |

Inbound / Outbound Service Configuration

The inbound/outbound message utility allows you to configure your system to receive information from and to send information to external applications using XML. The Landis+Gyr adapter for Smart Grid Gateway uses one inbound web service to map device events. This is the same inbound web service used by the D1 application.

Inbound Web Services

Inbound web services 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 inbound web 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 for Landis+Gyr includes the following inbound web services:

| Inbound Web Service | Description |
|----------------------|---------------------|
| D1-BulkRequestHeader | Bulk Request Header |

| Inbound Web Service | Description |
|--------------------------------|--|
| D1-BulkRequestUpdate | Bulk Request Update |
| D1-BulkResponse | Bulk Response |
| D1-DeviceEventSeeder | Used for upload of device events. 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. 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. |
| D3-CIMMeterReadingsResponse | CIM Meter On Demand Read Response Retrieve response from CIM On Demand Read command |
| D3-CIMPingResponse | CIM Ping Response Retrieve response from CIM Device Status Check command |
| D3-ConDisconStChgNotification | Initiate Connect Disconnect response. Retrieve response from the Initiate Connect Disconnect command. |
| D3-ReadingChangedNotification | Reading Changed Notification Notification that a Landis+Gyr device reading has changed. |
| D3-ScheduleDemandResetResponse | Scheduled Demand Reset Response Retrieve response from Demand Reset command |

Message Senders

Message 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 message senders you need to create is based on the types of messages the system is designed to accept.

The Oracle Utilities Smart Grid Gateway adapter for Landis+Gyr includes the following message senders:

| Message Sender | Description |
|----------------|--|
| D3-Comms | Commission Device |
| D3-Connect | Connect Device |
| D3-Decomm | Decommission Device |
| D3-Decomms | Decommissioning Sender |
| D3-DemReset | Demand Reset |
| D3-DerDevice | Deregister Device |
| D3-Disconnec | Disconnect Device |
| D3-InitMTR | Initiate Meter Read by Meter Number Outbound Message |
| D3-MTREXMS | Meter Exchange Notification Message Sender |

| Message Sender | Description |
|----------------|-------------------------------|
| D3-RTSender | Real Time Sender |
| D3-RTSnd | Real-time Sender (Landis+Gyr) |
| D3-SDemReset | SG Demand Reset |

BPEL Processes

These processes are responsible for performing the conversion from Oracle Utilities format to MultiSpeak 3.0 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, one of the following second business callout services is invoked to determine if the related “received” or “completed” callout should be executed:

- isExecutingCommissionReceivedCallout
- isExecutingCommissionCompletedCallout
- isExecutingDecommissionReceivedCallout
- isExecutingDecommissionCompletedCallout
- isExecutingAddMeterToInventoryReceivedCallout
- isExecutingAddMeterToInventoryCompletedCallout
- isExecutingMeterExchangeNotificationReceivedCallout
- isExecutingMeterExchangeNotificationCompletedCallout

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

CIMDeviceStatusCheck Composite Process: This process is similar to CIM OndemandRead BPEL process. SGG uses the result of On Demand Read command to identify the status of the meter. If reads are successfully returned, then meter is running healthy otherwise it is considered as meter inactive/dead.

DemandReset Composite Process: Invokes the remote endpoint to trigger the demand reset event. An asynchronous reply responds to the OUAF layer when the reading arrives.

LGProcessCallout 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.

Web Services

These web services are all defined in the Landis+Gyr 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.

| Web Service | Related BPEL Process | Description |
|-------------|------------------------|--|
| MR_CB | OnDemandRead | This web service is defined by the Landis+Gyr head end system's implementation of MR_Server. |
| | CommissionDecommission | |

| Web Service | Related BPEL Process | Description |
|------------------|---|---|
| | DemandReset | <p>The WSDL defines the interface for requesting a meter reading from the head end system.</p> <p>The actual definition can be obtained from L&G or downloaded from multispeak.org. Build 3.0aa is appropriate if obtained from MultiSpeak.</p> <p>Default endpoint must be changed in configuration: http://demo.turtletech.com/Multispeak/webapi/MR_CB.asmx</p> |
| CD_CB | ConnectDisconnect | <p>This web service is defined by the Landis+Gyr implementation of CB_CD.</p> <p>The WSDL defines the interface for requesting a meter's connection or disconnection on the head end system.</p> <p>This web service defines the interface for reporting a connection or disconnection by the head end system.</p> <p>This web service is only invoked by the head end system; not OUAF.</p> <p>Only the CDStateChangedNotification web method is implemented in the composite.</p> <p>Default endpoint must be changed in configuration: http://demo.turtletech.com/Multispeak/webapi/CD_CB.asmx</p> |
| CIMService | CIMOnDemandRead CIMDeviceStatusCheck | <p>This web service is defined by the L+G head end's implementation of AMIRequest Server.</p> <p>The WSDL defines the interface for requesting a meter reading from the head end system.</p> <p>The actual definition should be obtained from L&G or downloaded from L&G SDK for CIM 2.0.</p> |
| LGProcessCallout | OnDemandRead ConnectDisconnect CommissionDecommission | <p>Imported from LGProcessCallout Composite</p> <p>Default endpoint must be changed in configuration: http://127.0.0.1:8000/soa-infra/services/default/LGProcessCallout/LGProcessCallout</p> |

Landis+Gyr Command Center Web Services

The following table describes the Land+Gyr Command Center web services and operations used for the Oracle Utilities Smart Grid Gateway command messaging:

| Smart Grid Gateway Command | AMI Adapter Business Objects | Landis+Gyr Web Services | Landis+Gyr Operations |
|-----------------------------------|--|-------------------------|--------------------------------|
| Device Commissioning | D3-MeterAddNotificationMultiSp | MR | MeterAddNotification |
| Device Decommissioning | D3-MtrRmvNotifMultiSpeak | MR | MeterRemoveNotification |
| Remote Connect/ Remote Disconnect | D3-InitiateConnectDisconnect | CD | InitiateConnectDisconnect |
| On-Demand Read | D3-InitiateMRByMtrNbr | MR | InitiateMeterReadByMeterNumber |
| On-Demand Read (CIM) | D3-CIMGetLPData D3-CIMMeterOnDemandRead | CIMService | ScheduleDemandRead |
| Demand Reset | D3-ScheduleDemandReset | MR | CIM |

Chapter 3

Configuring a Landis+Gyr Head-End System

This section outlines the configuration required for the Oracle Utilities Smart Grid Gateway Adapter for Landis+Gyr to communicate with the Landis+Gyr Command Center software.

Inbound Web Services

Inbound web 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 L+G Command Center in response to a command request.

The following inbound web 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 inbound web services.

| Inbound Web Service Name | Description |
|--------------------------------|--------------------------------------|
| D1-BulkRequestHeader | Bulk Request Header |
| D1-BulkRequestUpdate | Bulk Request Update |
| D1-BulkResponse | Bulk Response |
| D1-DeviceEventSeeder | Device Event Seeder |
| D1-InitialLoadIMD | IMD Seeder |
| D1-PayloadErrorNotif | Payload Error Notification |
| D1-PayloadStatistics | Payload Statistics |
| D1-PayloadSummary | Payload Summary |
| D3-ConDisconStChgNotification | Initiate Connect Disconnect Response |
| D3-CIMMeterReadingsResponse* | CIM Meter On Demand Read Response |
| D3-CIMPingResponse | CIM Ping Response |
| D3-ReadingChangedNotification* | Reading Changed Notification |
| D3-ScheduleDemandResetResponse | Scheduled Demand Reset Response |

*The Landis+Gyr adapter supports both MultiSpeak and CIM On Demand Read commands. You only need to configure the inbound service for the protocol you wish to use.

Note: The following apply to all of the above inbound web services:

Message Options

- **Trace:** No
- **Debug:** No
- **Active:** Yes

Operations

- **Operation Name:** Same as web service name
- **Schema Type:** Business Object
- **Schema Name:** Applicable business object code
- **Transaction Type:** Add

Message Senders

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

The following message 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 message senders.

| Message Sender | Description |
|----------------|--|
| D3-Comms | Commission Device |
| D3-Connect | Connect Device |
| D3-Decomm | Decommission Device |
| D3-Decomms | Decommissioning Sender |
| D3-DemReset | Demand Reset |
| D3-DerDevice | Deregister Device |
| D3-Disconnect | Disconnect Device |
| D3-InitMTR | Initiate Meter Read by Meter Number Outbound Message |
| D3-MTREXMS | Meter Exchange Notification Message Sender |
| D3-RTSender | Real Time Sender |
| D3-RTSnd | Real-time Sender (Landis+Gyr) |
| D3-SDemReset | SG Demand Reset |

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

Main Tab:

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

Context Tab:

- **HTTP Header:** SOAPAction: http://xmlns.oracle.com/ouaf/multispeak_3.0/<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/D3/<SERVICE>/<SERVICE>

where:

- **<OPERATION>**: the operation performed by the message 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 message 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 |
|-----------------------|--|
| D3-ADDMTRINV | Add Meter to Inventory |
| D3-COMMS | Commission Device |
| D3-CONNECT | Connect Device |
| D3-DECOMMS | Decommission |
| D3-DEMRESET | Demand Reset |
| D3-DERDEV | Deregister Device |
| D3-DISCONNEC | Disconnect Device |
| D3-DVCSTSCHK | Device Status Check |
| D3-INITMRN | Initiate Meter Read by Meter Number |
| D3-INITMTR | Initiate Meter Read by Meter Number |
| D3-MTRADDNOT | Meter Add Notification Outbound Message Type |
| D3-MTREX | Meter Exchange Notification OB MSG |
| D3-MTRRMV | Meter Remove Notification |

Note: The following apply to all of the above 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 L+G Command Center must be present in your system. If this is 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 - Landis+Gyr:

- **External System:** LG

- **Description:** Landis+Gyr
- **Outbound Message Types::**

| Outbound Message Type | Description | Message Sender |
|-----------------------|--|--|
| D3-ADDMTRINV | Add Meter to Inventory | Message sender associated with the Add Meter to Inventory Outbound Message Type |
| D3-COMMS | Commission Device | Message sender associated with the Commission Device Outbound Message Type |
| D3-CONNECT | Connect Device | Message sender associated with the Connect Device Outbound Message Type |
| D3-DECOMMS | Decommission | Message sender associated with the Decommission Device Outbound Message |
| D3-DEMRESET | Demand Reset | Message sender associated with the Demand Reset Outbound Message Type |
| D3-DERDEV | Deregister Device | Message sender associated with the Deregister Device Outbound Message Type |
| D3-DISCONNEC | Disconnect Device | Message sender associated with the Disconnect Device Outbound Message Type |
| D3-DVCSTSCHK | Device Status Check | Message sender associated with the Device Status Check Outbound Message Type |
| D3-INITMRN | Initiate Meter Read by Meter Number | Message sender associated with the Initiate Meter Read By Meter Number Outbound Message Type |
| D3-INITMTR | Initiate Meter Read by Meter Number | Message sender associated with the Initiate Meter Read By Meter Number Outbound Message Type |
| D3-MTRADDNOT | Meter Add Notification Outbound Message Type | Message sender associated with the Meter Add Notification Outbound Message Type |
| D3-MTREX | Meter Exchange Notification OB MSG | Message sender associated with the Meter Exchange Notification Outbound Message Type |
| D3-MTRRMV | Meter Remove Notification | Message sender associated with the Meter Remove Notification Outbound Message Type |

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

- **Processing Method:** Real-time
- **Message XSL:** D3-Request.xsl
- **Response XSL:** D3-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 L+G Command Center must be present in your system. If this is 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 - Landis+Gyr:

- **Service Provider:** LG
- **Description:** Landis+Gyr
- **External Reference ID:** L+G
- **External System:** Landis+Gyr
- **Out Name/ID in Their System:**
- **AMI Device ID Type:** Internal Meter Number
- **AMI Measuring Component ID Type:** Channel ID

Processing Methods

Processing methods define the format or means by which a service provider receives and/or sends data from and/or to the application, including as 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 L+G Command Center.

The following types of processing methods must be configured for the L+G 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 inbound web service uses this processing method to determine which type of initial measurement business object to instantiate when receiving usage from the L+G Command Center.

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 inbound web service uses this processing method to determine which type of device event business object to instantiate when receiving device events from the L+G Command Center.

UOM Translation

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 L+G UOM codes to standard UOM codes when receiving usage from the L+G Command Center.

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 L+G service provider, based on the requirements of each implementation using the “How to Create OB COMM/Send OB Message” processing method business object (D1–HowToCreateActivityOBComm).

| Command | Processing Role | Default Business Object | Default Outbound Message Type |
|---------------------|---------------------|-------------------------------|-------------------------------|
| Device Commission | Device Registration | D3-AddMeterToInventoryMultiSp | Add Meter to Inventory |
| Device Commission | Device Installation | D3-MeterExNotificationMultiSp | Meter Exchange Notification |
| Device Decommission | Device Removal | D3-MtrRmvNotifMultiSpeak | Decommission |

| Command | Processing Role | Default Business Object | Default Outbound Message Type |
|--|---------------------------|---|---|
| | | | Meter Remove Notification |
| Device Deregistration | Device Deregistration | D3-MeterRetireNotification | Deregister Device |
| On-Demand Read (Scalar), Multispeak and CIM | On-Demand Read (Scalar) | MS: D3-InitiateMRByMtrNbr CIM: D3- CIMMeterOnDemandRead | MS: Initiate Meter Read by Meter Number CIM: CIM On Demand Read |
| On-Demand Read (Interval) | On-Demand Read (Interval) | D3-CIMGetLPData | CIM On Demand Read |
| Demand Reset | Demand Reset | D3-ScheduleDemandReset | Demand Reset |
| Device Status Check | Device Status Check | D3-CIMPing | Device Status Check |
| Remote Connect | Remote Connect | D3-InitiateConnectDisconnect | Connect Device |
| Remote Disconnect | Remote Disconnect | D3-InitiateConnectDisconnect | Disconnect Device |

Chapter 4

Configuring Landis+Gyr Extendable Lookups

This section outlines some of the extendable lookups that must be configured for use with the Landis+Gyr adapter. Refer to the Oracle Utilities Application Framework documentation for more information about working with extendable lookups.

CIM Response Status Extendable Lookup

The CIM Response Status extendable lookup is used to map descriptions to response status codes received from the L+G Command Center.

Each value defined for the CIM Response Status extendable lookup should include the following:

- **Response Status:** The CIM status code for the response status
- **Description:** A description of the response status
- **Usage Flag:** The status of the lookup value (can be Active or Inactive)

CIM Data Source Extendable Lookup

The CIM Data Source extendable lookup is used to map descriptions to data sources defined in the L+G Command Center.

Each value defined for the CIM Data Source extendable lookup should include the following:

- **Data Source:** The CIM code for the data source
- **Description:** A description of the data source
- **Usage Flag:** The status of the lookup value (can be Active or Inactive)

Landis+Gyr Device Event Mapping

The Landis+Gyr Device Event Mapping extendable lookup is used to determine which type of device event business object to instantiate when receiving device events from the L+G Command Center.

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

- **Head-End System Event Name:** The event name used by the Landis+Gyr Command Center
- **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.

Landis+Gyr UOM Code to Standard UOM Mapping

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

Each value defined for the Landis+Gyr UOM Code to Standard UOM Mapping extendable lookup should include the following:

- **Head-end UOM:** The unit of measure code used by the Landis+Gyr Command Center
- **Unit of Measure:** The unit of measure defined in the system.
- **Description:** A description of the unit of measure code.

Landis+Gyr Interval Status Code to Condition Mapping

Interval usage received from the Landis+Gyr Command Center can include Landis+Gyr 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 Landis+Gyr Interval Status Code to Condition Mapping extendable lookup is used to determine how to map Landis+Gyr interval status codes to standard status codes when receiving usage from the Landis+Gyr Command Center.

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

- **Interval Status:** The Landis+Gyr interval status code
- **Condition:** The condition code to which the interval status code is to be mapped, from the Measurement Condition extendable lookup.
- **Description:** A description of the interval status code.

Chapter 5

Extending the Landis+Gyr Adapter

The Oracle Utilities Smart Grid Gateway Adapter for Landis+Gyr supports a number of commands, including:

- Demand Reset
- Device Commission
- Device Decommission
- Device Deregistration
- Device Status Check (CIM)
- On-Demand Read (Multispeak)
- On-Demand Read (CIM)
- Remote Connect
- Remote Disconnect

The Adapter for Landis+Gyr can be extended to support additional commands provided by the Landis+Gyr Command Center.

Chapter 6

The Landis+Gyr Test Harness

Oracle Utilities Smart Grid Gateway Adapter for Landis+Gyr (LG) includes a test harness that can be configured to simulate the Landis+Gyr Gridstream Command Center head-end system for testing the two-way commands. The test harness is Multispeak 3.0 standard compliant and 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.

Test Harness Design

The L+G Harness is divided into two main layers. A "front end" set of services implements the LG-specified interfaces. They receive requests corresponding to:

- MR_CB (Meter Reading_Customer Billing)
 - MeterAddNotification
 - MeterRemoveNotification
 - InitiateMeterReadByMeterNumber
- CD_CB (Connect/Disconnect_Customer Billing)
 - InitiateConnectDisconnect

The LG Harness will send below responses to corresponding BPEL composites:

- CB_MR (Customer Billing_Meter Reading)
 - ReadingChangedNotification
- CB_CD (Customer Billing_Connect/Disconnect)
 - CDStateChangedNotification

Each of these services calls into the "back end" layer which defines meters and sets their attributes. These meters are stored in a file within the test harness called meterdb.xml. This file can be modified pre-deployment. Post-deployment changes to the file are not supported. However, the Test Harness retains an in-memory "database" of the meters in the file. The in-

memory representation can be modified using the Utility web services. Note that any changes to the in- memory structure will be lost when the server is restarted or the Test Harness composite is redeployed.

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:
3. 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/LG_Test/LGTestHarness/UtilService?WSDL
```

Web Services

This section describes the web services included in the Landis+Gyr test harness BPEL composite.

General Services

This section describes the general services of the Landis+Gyr test harness composite.

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 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](#) (see [UtilityFault](#) for more details).

ViewAuditTrail

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

Input — ViewAuditTrailInput

Part: payload

Element: ViewAuditTrailRequest

Output — ViewAuditTrailOutput

Part: payload

Element: ViewAuditTrailResult

An Entry consisting of a timestamp and an Operation. Each entry may have an associated meter object showing the latest update.

Fault — See [UtilityFault](#), above.

UtilityFault

Fault with similar mapping to SGG/OUAF faults:

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

Locate Meter Services

This section describes the locate meter web services of the Landis+Gyr test harness composite.

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](#). Unlike other methods, FindMeters does not throw an exception if the meter is not found. As such, it can be used to test for the existence of a Meter prior to querying for it.

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](#). 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](#). Thrown when meter id is not found.

Meter Administration Services

This section describes the meter administration services of the Landis+Gyr test harness composite.

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. |
| macID | A MAC address that must be unique within the system. |
| 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. |
| groupName | The name linking multiple meters together into a set. |
| jobExecutionStatus | One of the possible Job Execution Status values (see schema). This attribute determines how requested jobs perform. |
| updateIfExisting | 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](#)

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](#). 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

Whether or not the channel was added to the index.

Fault — See [UtilityFault](#). Thrown when meter id is not found.

RemoveMeterChannel

This web service removes a Channel from a meter.

Input — RemoveMeterChannelInput

Part: payload

Element: RemoveMeterChannelRequest

| 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

Whether or not the channel was removed from the meter.

Fault — See [UtilityFault](#). 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. |

Meter Attribute Administration Services

This section describes the meter administration services of the Landis+Gyr test harness composite.

GetLoadActionCode

This web service queries whether the given meter is connected or disconnected. This method is used by the Connect/Disconnect service. The values for load action code are:

- Connect
- Disconnect

Input — GetLoadActionCodeInput

Part: payload

Element: GetLoadActionCodeRequest

| Parameter | Description |
|-----------|--|
| id | The ID for the meter for which the load action code status should be retrieved |

Output — GetLoadActionCodeOutput

Part: payload

Element: GetLoadActionCodeResult

The connection status of the requested meter.

Fault — See [UtilityFault](#). Thrown when meter id is not found.

SetLoadActionCode

This web service updates the load action code for a given meter. This method is used by the Connect/Disconnect service. The values for load action code are:

- Connect
- Disconnect

Input — SetLoadActionCodeInput

Part: payload

Element: SetLoadActionCodeRequest

| Parameter | Description |
|-----------|---|
| id | The ID for the meter for which the load action code status 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](#). Thrown when meter id is not found.

IsCommissioned

This web service queries the commissioning status for a given meter. This service is used by the Commission/Decommission process. The commissioning attribute can be true or false.

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](#). Thrown when meter id is not found.

SetCommission

This web service updates the commissioning status for a given meter. This service is used by the Commission/Decommission process. The commissioning attribute can be true or false.

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](#). Thrown when meter id is not found.

GetExecutionStatus

This web service queries the status of the property controlling the overall execution of the command. The possible values of execution status are:

- Success - The command should complete successfully
- ResponseTimeout - The asynchronous response will never arrive
- SyncOperationFail - A simulated fault will occur in the during the initial request
- AsyncOperationFailure - A simulated fault will occur in the asynchronous response

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](#). Thrown when meter id is not found.

SetExecutionStatus

This web service updates the property controlling the overall completion of the command. The possible values of execution status are:

- Success - The command should complete successfully
- ResponseTimeout - The asynchronous response will never arrive
- SyncOperationFail - A simulated fault will occur in the during the initial request
- AsyncOperationFailure - A simulated fault will occur in the asynchronous response

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](#). Thrown when meter id is not found.

Chapter 7

Landis+Gyr Interval Data Mapping

This section describes how data in Landis+Gyr import files is mapped to the XML document format created by OSB and sent to Oracle Utilities Smart Grid Gateway.

When Landis+Gyr data is processed, it is initially received in a tilda-separated file format, which is converted into a "plan" XML format before being converted into the "result" XML format which is sent to the IMD Seeder and/or Device Event Seeder inbound services.

Non-Interval Usage with Additional Fields

The following is a sample file of non-interval usage that contains additional fields.

ID~PremiseID~ESIID~Provisioned~Meter~kWh~DateTime~Peak~PeakDateTime~Dmd~TouA~TouB~TouC~TouD~TouE~Volts~PF~P

EMED01~SL002~~~96968280~34315.000~08042010120000AM~2.66~03122009063000AM~~34315.000~0.000~0.000~0.000~0.000~

EMED01~SL001~~~96968285~33693.000~08042010120000AM~2.62~03122009061500AM~~33693.000~0.000~0.000~0.000~0.000~

This data is mapped to the "plain" XML format.

XML 'Plain' XML Format

The "Plain" XML contain elements to hold the extra fields (highlighted in bold).

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
  targetNamespace=" http://xmlns.oracle.com/LandisGyrUsage"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MeterReads">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="MeterRead">
          <xs:complexType>
            <xs:sequence>
```

```

<xs:element name="Origin"/>
<xs:element name="ServProvExtRefId"/>
<xs:element name="RecordType">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="MEPMD01" />
      <xs:enumeration value="EMED01" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="RecordVersion" minOccurs="0">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="20080519" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="TimeStamp" />
<xs:element name="Premise" minOccurs="0" />
<xs:element name="ESIID" minOccurs="0" />
<xs:element name="Provisioned" minOccurs="0" />
<xs:element name="MeterID" />
<xs:element name="Purpose" minOccurs="0" />
<xs:element name="Comodity" minOccurs="0" />
<xs:element name="Units" minOccurs="0" />
<xs:element name="CalcConst" minOccurs="0"/>
<xs:element name="Interval" minOccurs="0"/>
<xs:element name="Count" minOccurs="0"/>
<xs:element name="FirstIntervalDateTime" />
<xs:element name="Data">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Row" maxOccurs="unbounded" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="v" />
          <xs:attribute name="s" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="kWh" minOccurs="0" />
<xs:element name="Peak" minOccurs="0" />
<xs:element name="PeakDateTime" minOccurs="0" />
<xs:element name="Dmd" minOccurs="0" />
<xs:element name="TouA" minOccurs="0" />
<xs:element name="TouB" minOccurs="0" />
<xs:element name="TouD" minOccurs="0" />
<xs:element name="TouC" minOccurs="0" />
<xs:element name="TouE" minOccurs="0" />
<xs:element name="Volts" minOccurs="0" />
<xs:element name="PF" minOccurs="0" />
<xs:element name="ExtraFields" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ExtraField" maxOccurs="255" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="FieldName" minOccurs="0"/>
            <xs:element name="FieldValue" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="RawData" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

```

```
</xs:element>
</xs:schema>
```

Non-Interval Usage to 'Plain' XML Mapping

The following table shows the mapping between fields in incoming non-interval data and child elements of MeterReads/ MeterRead element in the "Plain" XML format:

| L+G Interval Structure Field | "Plain" XML element |
|------------------------------|---|
| ID | RecordType |
| Premise ID | Premise |
| ESIID | ESIID |
| Provisioned | Provisioned |
| Meter | MeterID |
| kWh | kWh |
| Date/Time of initial Read | FirstIntervalDateTime |
| Peak | Peak |
| Peak Date/Time | PeakDateTime |
| Dmd | Dmd |
| TouA | TouA |
| TouB | TouB |
| TouC | TouC |
| TouD | TouD |
| TouE | TouE |
| Volts | Volts |
| PF | PF |
| <extraFieldName1> | ExtraFields/ExtraField/FieldName with value of <extraFieldName1> ExtraFields/ExtraField/FieldValue with value in <extraFieldName1> |
| <extraFieldName2> | ExtraFields/ExtraField/FieldName with value of <extraFieldName2> ExtraFields/ExtraField/FieldValue with value in <extraFieldName2> |
| <extraFieldName3> | ExtraFields/ExtraField/FieldName with value of <extraFieldName3> ExtraFields/ExtraField/FieldValue with value in <extraFieldName3> |
| <extraFieldNameN> | ExtraFields/ExtraField/FieldName with value of <extraFieldNameN> ExtraFields/ExtraField/FieldValue with value in <extraFieldNameN> |
| RawData | A record content from incoming file. |

'Plain' XML to IMD Mapping

The following table outlines how data from the "plain" XML format is mapped to the InitialLoadIMD format when the Landis+Gyr record type is set to "MEPMD01" (interval usage).

| "Plain" XML element | InitialLoadIMD element | Note |
|---------------------|------------------------|------|
| Record Type | N/A | |
| RecordVersion | N/A | |
| Premise ID | N/A | |
| ESIID | N/A | |
| Provisioned | N/A | |

| "Plain" XML element | InitialLoadIMD element | Note |
|-------------------------------|---------------------------|--|
| Meter ID | dvclIdN | as is |
| Purpose | N/A | |
| Commodity | N/A | |
| Units | externalUOM | as is |
| CalcConst | mcm | as is |
| Interval | spi | Transform from DDHHMM to SPI |
| Count | N/A | |
| FirstIntervalDateTime | stDt | Convert to OUAF date/time format |
| Data | msrs | Value -> msrs/mL/q |
| Status -> msrs/mL/sts/stsL/st | | |
| N/A | imdType | 'D1IL |
| Origin | externalId | the origin attribute in "Plain" XML (incoming file name) |
| N/A | serviceProviderExternalId | L+G |
| RawData | rawData | Vendor-specific "raw" data |

Non-Interval 'Plain' XML to IMD Mapping

The following table outlines how non-interval data in the "plain" XML format is mapped to the InitialLoadIMD format when the Landis+Gyr record type is set to "EMED01" (non-interval usage).

| "Plain" XML Element | InitialLoadIMD element | Notes |
|---------------------|------------------------|--|
| RecordType | N/A | |
| Premise | N/A | |
| ESIID | N/A | |
| Provisioned | N/A | |
| MeterID | dvclIdN | as is |
| kWh | enQty | as is |
| | uom | KWH |
| TimeStamp | enDt | Convert to OUAF date/time format This should be mapped to enDt. Relevant only to non-interval data. |
| Peak | enQty | as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KW |
| PeakDateTime | enDt | Convert to OUAF date/time format |
| Dmd | enQty | as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KW and 'Dmd' in " <mcIdN> |
| TouA | enQty | as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KWH and 'kWh Rate A' in " <mcIdN> |
| TouB | enQty | as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KWH and 'kWh Rate B' in " <mcIdN> |
| TouC | enQty | as is |

| "Plain" XML Element | InitialLoadIMD element | Notes |
|---------------------|---------------------------|---|
| | | A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KWH and 'kWh Rate C' in "<mcIdN> |
| TouD | enQty | as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KWH and 'kWh Rate D' in "<mcIdN> |
| TouE | enQty | as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KWH and 'kWh Rate E' in "<mcIdN> |
| Volts | enQty | as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with Volts |
| PF | enQty | as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with PF |
| N/A | imdType | D1IL |
| Origin | externalId | the origin element in "Plain" XML (incoming file name) |
| N/A | serviceProviderExternalId | L+G This is a constant. There is no such field in incoming structure. |
| RawData | rawData | Vendor-specific "raw" data When the "populateRawIMD" parameters in the EnvironmentSettings.xq file is set to true. |

Mapping Additional Fields

Measurement and device event files received from the Landis+Gyr head-end system can include additional fields containing data to be imported into Oracle Utilities Smart Grid Gateway. These additional fields must be mapped to elements within the XML document processed by OSB and sent to Smart Grid Gateway.

This mapping can be performed through use of a custom XQuery document, specified in the EnvironmentSettings.xq file via the "modifyResultXMLInput" parameter.

The following sample XQuery documents illustrate how additional fields can be mapped into the XML format sent to Smart Grid Gateway.

Sample XQuery — Initial Measurements

The following XQuery is an example that shows a transformation that passes in a root element with 3 children (the “result” XML, the “plain” XML, the environment settings) that returns a modified “result” XML. For testing purposes, it changes the original value in the <enQty> “result” element and replaces it with a value from the “plain” XML depending on an environment setting variable. The <serviceProviderExternalId> value was also replaced by a hard-coded value.

```

declare namespace lan = "http://xmlns.oracle.com/LandisGyrUsage";
declare namespace xf = "http://tempuri.org/D3/lgimd";
declare namespace soap = "http://schemas.xmlsoap.org/soap/envelope/";
declare function xf:modifyResultXML($modifyResultXMLInput as element(*)) as element(*){
<InitialLoadIMDList>
{
  for $InitLoadIMD in $ modifyResultXMLInput/InitialLoadIMDList/InitialLoadIMD
  return
    <InitialLoadIMD>
      <preVEE>
        <dvcIdN>{ data($InitLoadIMD/preVEE/dvcIdN) }</dvcIdN>
        <externalId>{ data($InitLoadIMD/preVEE/externalId) }</externalId>
        <uom>{ data($InitLoadIMD/preVEE/uom) }</uom>
        <mcIdN>{ data($InitLoadIMD/preVEE/mcIdN) }</mcIdN>

```

```

        <enDt>{ data($InitLoadIMD/preVEE/enDt) }</enDt>
        {
            if ($modifyResultXMLInput/EnvironmentSettings/test1="true")
            then <enQty>{ data($modifyResultXMLInput/lan:MeterReads/lan:MeterRead/lan:ExtraFields/
lan:ExtraField[lan:FieldName
= 'EF4']/lan:FieldValue) }</enQty>
            else <enQty>{ data($modifyResultXMLInput/lan:MeterReads/lan:MeterRead/lan:ExtraFields/
lan:ExtraField[lan:FieldName= 'EF2']/lan:FieldValue) }</enQty>
            }
        <imdType>{ data($InitLoadIMD/preVEE/imdType) }</imdType>
    </preVEE>
    <serviceProviderExternalId>NewSPID</serviceProviderExternalId>
    </InitialLoadIMD>}</InitialLoadIMDList>
};
declare variable $modifyResultXMLInput as element(*)external;
xf:modifyResultXML($modifyResultXMLInput)

```

Sample XQuery — Device Events

The following XQuery is an example that shows a transformation that passes in a root element with 3 children (the "result" XML, the "plain" XML, and the environment settings) and returns a modified "result" XML. For testing purposes, it changes the original value in the `<externalCommunicationModuleIdentifier>` "result" element and replaces it with a value from the "plain" XML depending on an environment setting variable. The `<externalServiceLocationId>` value is also replaced by a hard-coded value.

```

declare namespace lan = "http://xmlns.oracle.com/LandisGyrEvent";
declare namespace xf = "http://tempuri.org/D3/event";
declare namespace soap = "http://schemas.xmlsoap.org/soap/envelope/";
declare function xf:modifyResultXML($modifyResultXMLInput as element(*)) as element(*){
    <DeviceEventSeeder>
        <externalSenderId>{data($modifyResultXMLInput/DeviceEventSeeder/externalSenderId) }</
externalSenderId>
        <deviceIdentifierNumber>{ data($modifyResultXMLInput/DeviceEventSeeder/
deviceIdentifierNumber) }</deviceIdentifierNumber>
        <externalEventName>{ data($modifyResultXMLInput/DeviceEventSeeder/externalEventName) }</
externalEventName>
        <eventDateTime>{ data($modifyResultXMLInput/DeviceEventSeeder/eventDateTime) }</
eventDateTime>
        <externalSourceIdentifier>{ data($modifyResultXMLInput/DeviceEventSeeder/
externalSourceIdentifier)
        }</externalSourceIdentifier>
        <eventInformation>
            <externalEventCategory>{data($modifyResultXMLInput/DeviceEventSeeder/eventInformation/
externalEventCategory)
            }</externalEventCategory>
            <externalEventSeverity>{data($modifyResultXMLInput/DeviceEventSeeder/eventInformation/
externalEventSeverity)
            }</externalEventSeverity>
            <externalDeviceType>{data($modifyResultXMLInput/DeviceEventSeeder/eventInformation/
externalDeviceType)
            }</externalDeviceType>
            <externalServiceLocationId>{1234 }</externalServiceLocationId>
            {
                if ($modifyResultXMLInput/EnvironmentSettings/testA="true")
                then <externalCommunicationModuleIdentifier>{ data($modifyResultXMLInput/lan:DeviceEvents/
lan:DeviceEvent/lan:DeviceType) }</externalCommunicationModuleIdentifier>
                else <externalCommunicationModuleIdentifier>{data($modifyResultXMLInput/lan:DeviceEvents/
lan:DeviceEvent/lan:CategoryId) }</externalCommunicationModuleIdentifier>
            }
            <externalStatusValue>{data($modifyResultXMLInput/DeviceEventSeeder/eventInformation/
externalStatusValue)
            }</externalStatusValue>
            <externalStatusDateTime>{data($modifyResultXMLInput/DeviceEventSeeder/eventInformation/
externalStatusDateTime) }</externalStatusDateTime>
        </eventInformation>
    </DeviceEventSeeder>
};
declare variable $modifyResultXMLInput as element(*) external;
xf:modifyResultXML($modifyResultXMLInput)

```