# Oracle® Transportation Management

Integration Guide

Release 6.4.2

Part No. E81540-02

March 2018

# Copyright Notice

Oracle® Transportation Management Integration Guide, Release 6.4.2

Part No. E81540-02

Documentation Accessibility

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

Access to Oracle Support

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

# Contents

## Figures

# Send Us Your Comments

Oracle® Transportation Management Integration Guide, Release 6.4.2

Part No. E81540-02

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: otm-doc_us@oracle.com

If you have problems with the software, contact Support at https://support.oracle.com or find the Support phone number for your region at http://www.oracle.com/support/contact.html.

# Preface

This manual is for members of the Oracle Transportation Management and Global Trade Management implementation teams, who are responsible for connecting the system to other external systems through integration interfaces. This manual explains how to send and receive integration messages and the format of each message.

This manual does not cover the installation of any components required to import or export. See the Administration Guide and Installation Guide in your Oracle Transportation Management installer for installation and configuration instructions.

## Change History

| Date | Document Revision | Summary of Changes |
|---|---|---|
| 12/2016 | -01 | Expanded descriptions and reorganization to cover the Transmission schema design. |
| 03/2018 | -02 | Updated the sample URL for accessing the Customs Info status document. The URL may resemble the following: https://gtm.content.descartes.com/status/statusdocument.xml |
| | | |
| | | |
| | | |
| | | |
| | | |

# 1. Integration Overview

The Oracle Transportation Management (OTM) and Global Trade Management (GTM) suite of products expose a large number of interfaces to support integration with external applications. There are two distinct use cases for application integration covered in this guide: Business Process Integration and User Interface Integration. Data Tier Integration is covered in the Data Management Guide.

Business Process Integration is defined in this guide as the sending and receiving of functional messages between applications. These messages are implemented as **Transmission** or **Mobile Device Communication** XML documents.

The Transmission XML document is the primary application integration message and is covered in detail in the following chapters in this guide. Mobile Device Communication integration is built on the infrastructure described in the main chapters of this guide but uses a different XML message format and is optimized for high volume, small messages e.g. for Fleet power or trailer devices. The details for this integration are covered in the Chapter 17: Mobile Device Communications.

As an example functional message, the Transportation Orders (`TransOrder`) Transmission interface receives transportation order messages from an external application into OTM. This is referred to as an **inbound** interface. The **outbound** interfaces send messages from the OTM/GTM application to other external applications. For example, shipments planned from Orders in OTM can be sent in a `PlannedShipment` message to another system for additional processing.

The valid formats of all messages are described by Integration XML Schema Definition (XSD) documents.

This document will describe the following:-

- The purpose of each XSD and the available messages defined by it.
- The application level message protocol describing the process of message exchange.
- The transport level protocols supported for inbound and outbound message communication.

User Interface Integration is defined in this guide as the retrieval and modification of application data to support custom user interfaces. This is achieved by the use of the OTM/GTM Representational State Transfer (ReST) API, which is covered in Chapter 20: REST API Reference.

# 2. Understanding the Integration Schemas

This section will describe the overall format and design of the Integration XML Schema Definition (XSD) schemas for use when implementing interfaces.

## Definitions

The following definitions are used widely in the remainder of the document and deserve detailed explanation of their exact meaning.

| **Term** | **Definition** |
|---|---|
| Global element/type/attribute | This is an XSD definition which is an immediate child of the <xsd:schema> document element. |
| Local element/type/attribute | This is an XSD definition that is contained within a parent definition e.g. an element definition within a named complex type. |
| Local reference | This refers to using the 'ref=' attribute to reference a global element definition i.e. as opposed to a Local element definition. |
| Primary Document | These are the global element definitions for XML documents which can be sent as messages to and from OTM/GTM. The main example is the `Transmission`. |
| Transaction interface | These are the definitions which relate to the high level business object interface e.g. for Orders, Shipments, Invoices etc. The `Transmission` primary document will contain one or more Transaction interface elements. |

## The OTM/GTM Integration XML Schemas

The OTM/GTM Integration XML schemas define the data elements that the system sends or receives for each type of interface. The XML schema definitions are considered the true definition for the interfaces, and this Integration Guide covers concepts which apply to all schemas. Information appearing in the schemas will be specific to each interface and will take priority over the generic information in this guide.

There are three sets of XML schema documents:

- Transmission related schemas
- Mobile Device related schemas
- Workflow Web Service related schemas

  **Note**: In prior versions, the Transmission related schema definitions were contained in the GLogXML.xsd and GLogXML-GTM.xsd schema definitions. Starting in version 6.4.2, these definitions have now been reorganized into a new set of schemas split by functional area.

This is a significant change and will be covered in detail in the following section - Transmission Schema.

The Mobile Device Communication Message Schema defines the format in which you send or receive mobile device messages and are covered in Chapter 17: Mobile Device Communications.

The Workflow Web Service related schemas are used by a set of Web Services which expose Agent workflow triggers. See Workflow Web Service for details.

### *W3C XML Schema Version*

All schema files conform to the W3C XML Schema standard (see http://www.w3.org/XML/Schema) defined by the following namespace name:

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

### *XML Schema Changes*

All changes to XSD documents for a particular release are described in the XML Interface Changes Guide document.

### *Viewing the XML Schemas*

The schema files can be obtained from the following User Interface menu location:-

Business Process Automation > Integration > Integration Manager > Retrieve Schemas

When integrating to Oracle Transportation Management using XML, you must create documents that follow the structure and rules of the Oracle Transportation Management XML schemas. We recommend that you use an XML management tool to view the schema files. This will help in understanding the Oracle Transportation Management and Global Trade Management data elements and relationships.

The W3C XML Schema site (http://www.w3.org/XML/Schema) provides links to several such tools. The examples in this document use the freely available Oracle XSD Visual Editor which is a built-in part of the JDeveloper IDE (see http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html for details.)

For a full description of the Visual Editor tool please see Developing Applications Using XML in the Oracle Fusion Middleware User's Guide for JDeveloper online documentation (http://docs.oracle.com).

## Transmission Schema

### *Schema Files*

The following schema files contain all definitions for the Transmission schema:

| Schema Name | Description |
|---|---|
| Transmission.xsd | Definitions for all Primary Documents. |
| TransmissionCommon.xsd | Definitions for types shared across interfaces. |
| Transaction.xsd | This is a convenient schema which imports all other Transaction definition schemas. |

| **Schema Name** | **Description** |
|---|---|
| GTM.xsd | Transaction interfaces for GTM application functionality. |
| LocationContact.xsd | Transaction interfaces for Location and Contact business objects and other related common types used in other interfaces e.g. Involved Party. |
| ShipUnit.xsd | Transaction interfaces for Ship Units common to Orders, Shipments etc. |
| Item.xsd | Transaction interfaces for Item and Package related business objects. |
| Document.xsd | Transaction interfaces for Document Content for Content Management Systems. |
| Configuration.xsd | Low level transaction interfaces e.g. User management. |
| Shipment.xsd | Transaction interfaces for Shipment related business objects including Planned, Actual and Tendered shipments. |
| Order.xsd | Transaction interfaces for Purchase Orders and Order Release business objects. |
| Planning.xsd | Transaction interfaces related to planning of Orders onto Shipments. |
| Finance.xsd | Transaction interfaces related Invoices, Payments etc. |
| Rate.xsd | Transaction interfaces related to Rating structures used for cost calculations. |
| Job.xsd | Transaction interfaces for Brokerage and Forwarding. |
| GenericTransaction.xsd | Miscellaneous transaction interfaces that can apply to many business object types e.g. for object status updates. |

### *XML Namespaces*

Starting in version 6.4.2 a new namespace URL will be used to identify XML messages which are valid according to the new set of schema files listed in the preceding table.

The following table lists the namespace URLs and associated physical file name for each new schema defined in version 6.4.2:

| **XSD Schema File** | **Namespace** |
|---|---|
| Transmission.xsd and all other non-GTM schemas (listed in the Transmission Schema table). | `http://xmlns.oracle.com/apps/otm/transmission/v6.4` |
| GTM.xsd | `http://xmlns.oracle.com/apps/gtm/transmission/v6.4` |

## *Primary XML Documents*

There are three primary XML documents defined in the Transmission schema that are used inbound and outbound for both Oracle Transportation Management interfaces and Global Trade Management interfaces:

- Transmission
    - The Transmission is the primary document used for messages inbound to and outbound from the system. Each Transmission can contain multiple transactions to be processed. A unique Transmission Number is assigned to each inbound and outbound message.
- TransmissionAck
    - The TransmissionAck is the response message to the receipt of the Transmission. It contains the confirmation for the receipt of the Transmission with the unique assigned Transmission Number, or an error if the Transmission could not be persisted.
- TransmissionReport
    - The TransmissionReport summarizes the errors that were detected during the processing of the Transmission. The report is optionally sent after all the transactions in the Transmission have been completed (successfully processed or with generated errors). The requirement to receive a TransmissionReport is indicated in the inbound Transmission Header.

## *Transmission Processing Styles*

There are two different styles for processing the contents of a Transmission:

- Transactional style: where the Transactions within the Transmission are processed as messages used to create, modify or delete transactional content or to trigger processing for transactional content.
- Query style: where the Transactions within the Transmission are request/reply transactions to retrieve content or return the results of processing against transactional content.

The style used is based on a combination of the value of the **Transmission Type** Transmission header property and the interfaces present in the Transmission body content. If no Transmission Type value is specified the style defaults to Transactional.

It is not supported to mix Transactional style content with Query style content in the same Transmission. It is also not valid to request Query style processing for interfaces which do not support that style.

### Transactional Style Interfaces

Transactional Interfaces are 'submitted' to OTM for subsequent processing similar to asynchronous processing[1]. This allows the OTM Application server to manage workload and ensure changes to persistent object data are handled within a transactional context.

[1] Strictly speaking the process is a synchronous transaction that accepts the message for later processing followed by the asynchronous processing of message. The synchronous part returns the reference which can be used to track the results of processing.

The following diagram shows how the documents are used when processing an inbound transactional Transmission XML into Oracle Transportation Management.

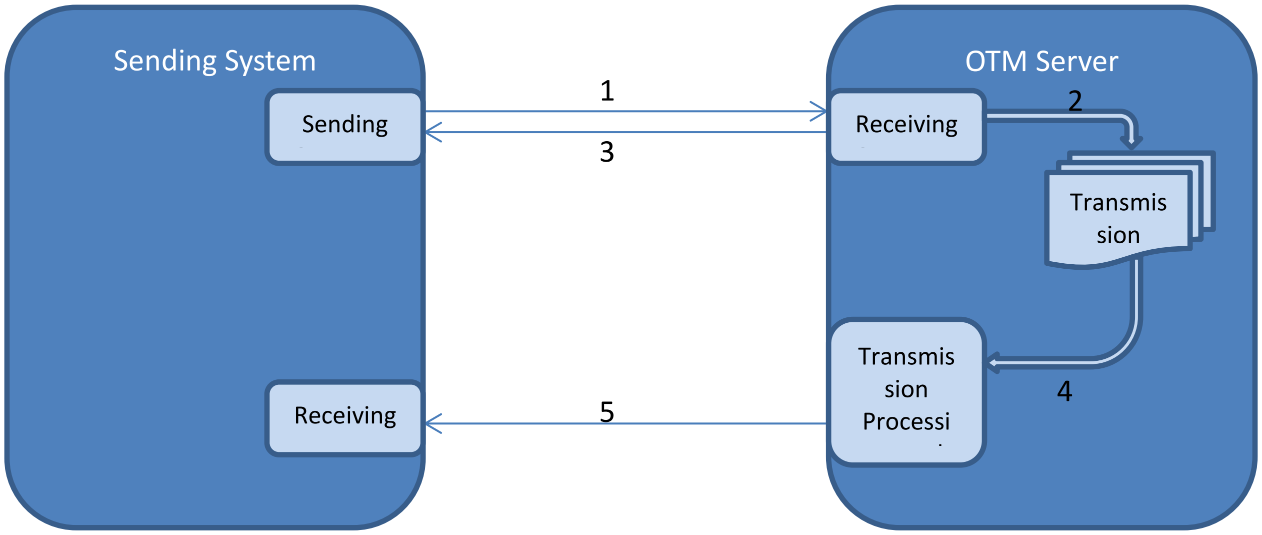


**Figure 2-1**

1. Transmission XML document is sent to OTM. The nature of the sending component depends on the transport protocol used e.g. it will be an HTTP client if sending the message via HTTP POST.
2. Transmission message is persisted for later processing generating a unique Transmission number in the process.
3. A TransmissionAck document is returned which will contain the Transmission number. The nature of the response depends on the transport protocol used e.g. if using HTTP it will be a synchronous response to the initial HTTP POST. If using message oriented middleware the response will be queued to a configured 'Acknowledgment' queue.
4. The background Transmission processing daemon retrieves the Transmission at some time later and processes it e.g. creates or updates the business object using the original XML message content.
5. A TransmissionReport is optionally sent to confirm the status (PROCESSED or ERROR) of the Transmission message. Again the report message contains the Transmission number for correlation. The sending of the report is configurable i.e. 'always', 'never' or 'on error only'. The nature of the receiving component depends on transport method e.g. it can be a Servlet URL, Message Queue, Web Service endpoint and the transport method can be different to original inbound transport method.

### Query Style Interfaces

Query style interfaces are used when synchronous processing is required for example, for a rating request.

The following diagram shows the documents used when processing an inbound query Transmission.

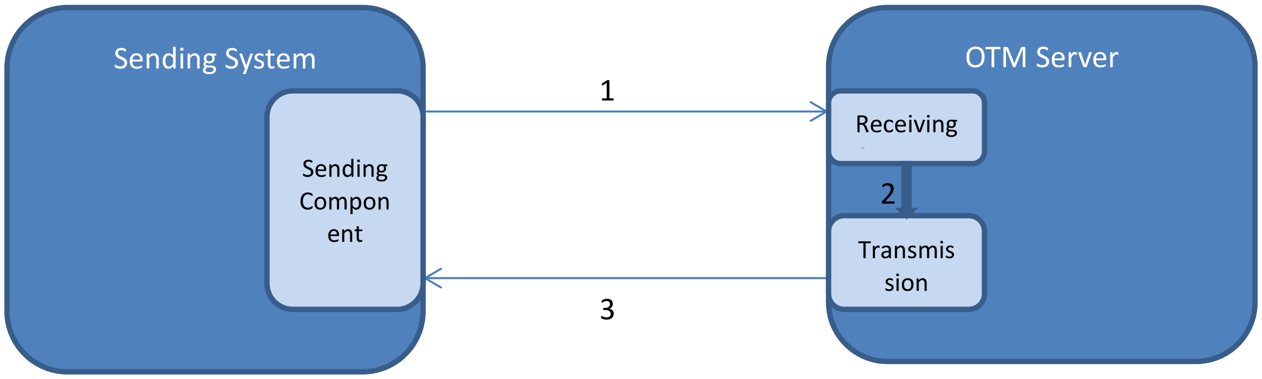


**Figure 2-2**

1. Transmission XML is sent to OTM. The nature of the sending component depends on the transport protocol used e.g. it will be an HTTP client if sending message via HTTP POST.
2. The internal processing is called. The Transmission Type property must be present and be either 'QUERY' or 'SERVICE' depending on the interface to be processed.
3. The processing results are returned as either a TransmissionAck or a new Transmission document. The document used can depend on:-
    a. The protocol used. For example, Web Service calls must always return the document declared in the WSDL output parameter.
    b. Custom property settings (if not using a Web Service). For example, when sending multiple RIQ requests via HTTP, the following property will result in a Transmission being returned:-

    ```
    glog.integration.remoteQuery.WrapReplyInTransmission=1
    ```

### *Transmission Structure*

The Transmission XML document has a root element with local name `Transmission`. The current namespace URL is `http://xmlns.oracle.com/apps/otm/transmission/v6.4`.

The `Transmission` contains a `TransmissionHeader` and a `TransmissionBody`. The `TransmissionBody` contains one or more interface transactions, each wrapped in a `GLogXMLElement` element. Each `GLogXMLElement` can contain an optional `TransactionHeader` element but must contain an **interface element** which must be a type of `GLogXMLTransaction` element. See figure below for schema diagram representation.

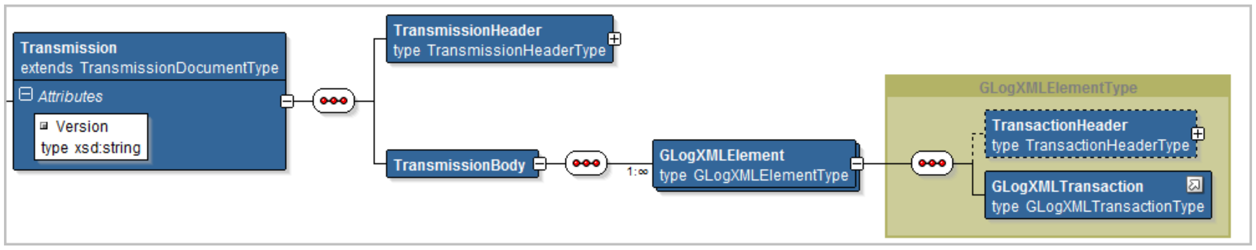


**Figure 2-3**

The Transmission schema uses **Substitution Groups** to define the possible interface elements that can be contained by a `GLogXMLElement`. This removes the need for the Transmission schema to list all possible interface elements. This approach allows future schemas to define new interfaces without having to change the Transmission schema which is a more extensible approach.

The `GLogXMLTransaction` is essentially a marker element which specifies the base type – `GLogXMLTransactionType` - of the elements that can be used in its place i.e. substituted for it. Only elements whose type extends the `GLogXMLTransactionType` can be substituted for the `GLogXMLTransaction` element. At present there are three types which extend `GLogXMLTransactionType` to further qualify the interface and provide additional control for how the interface can be used. See figure below for a schema diagram of the three types and the subsequent explanation.

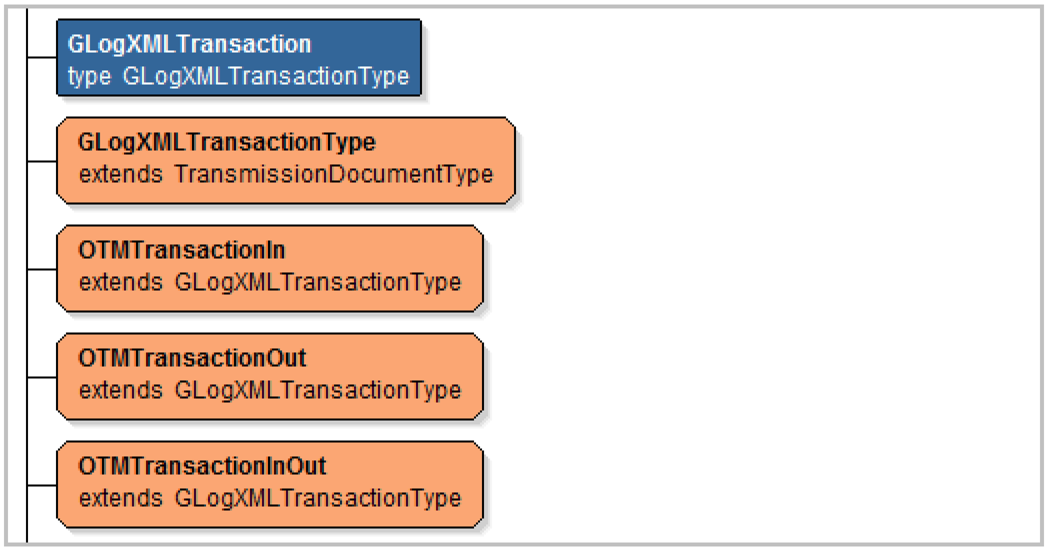
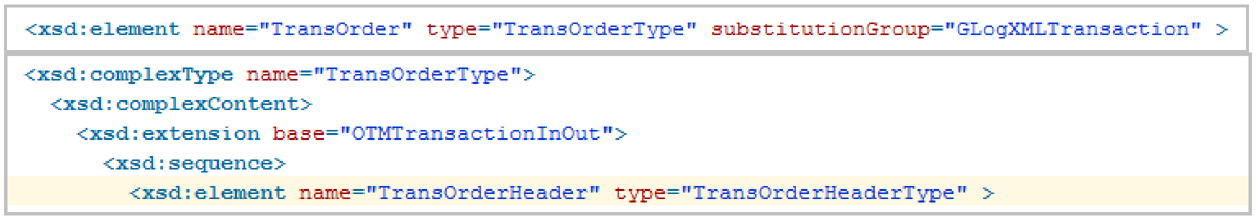

GLogXMLTransaction
type GLogXMLTransactionType
GLogXMLTransactionType
extends TransmissionDocumentType
OTMTransactionIn
extends GLogXMLTransactionType
OTMTransactionOut
extends GLogXMLTransactionType
OTMTransactionInOut
extends GLogXMLTransactionType

**Figure 2-4**

- `OTMTransactionIn` – Interfaces which are only valid for inbound processing must extend this type
- `OTMTransactionOut` - Interfaces which are only valid for outbound processing must extend this type
- `OTMTransactionInOut` - Interfaces which are valid for both inbound and outbound processing must extend this type

For example, the `TransOrder` interface is valid inbound and outbound and so has the following schema definition (some information has been removed to aid clarity for this discussion):



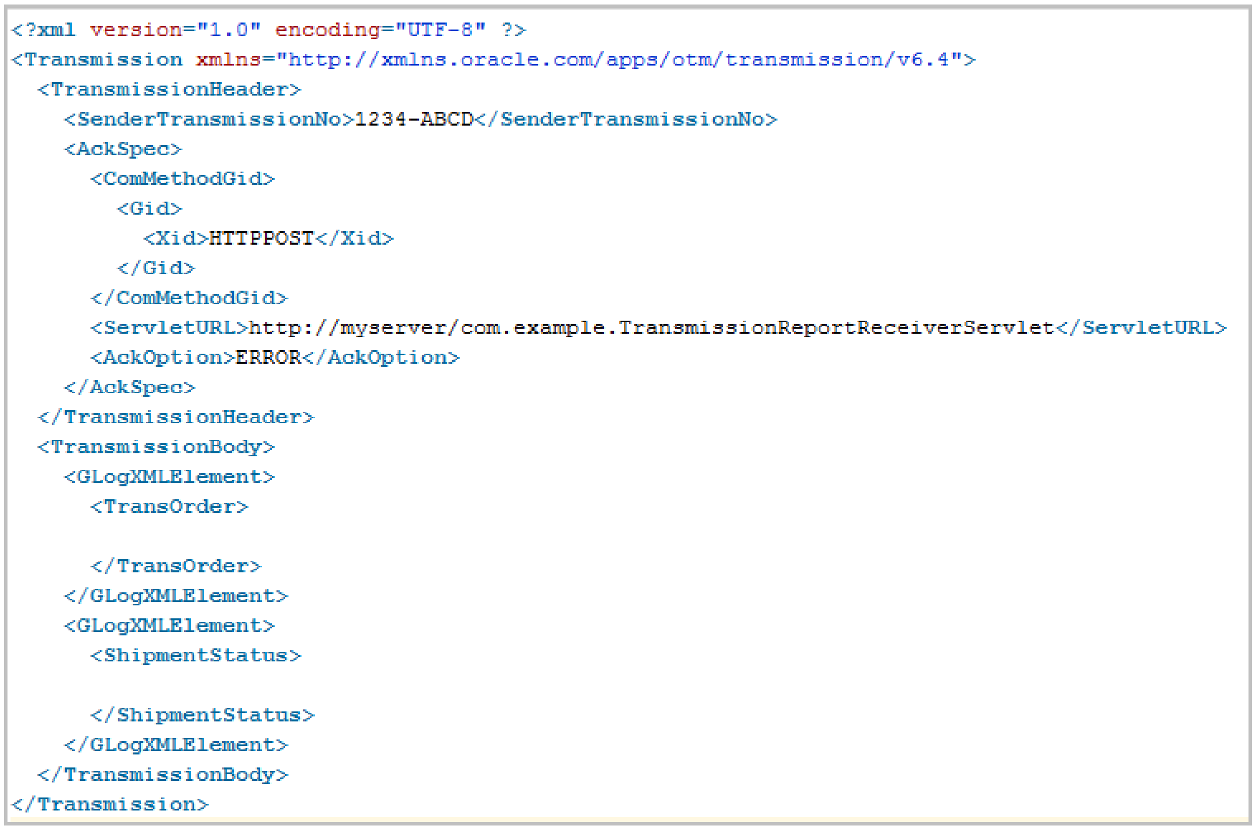
```
<xsd:element name="TransOrder" type="TransOrderType" substitutionGroup="GLogXMLTransaction" >
```

```
<xsd:complexType name="TransOrderType">
  <xsd:complexContent>
    <xsd:extension base="OTMTransactionInOut">
      <xsd:sequence>
        <xsd:element name="TransOrderHeader" type="TransOrderHeaderType" >
```

**Figure 2-5**

The `TransOrder` element definition declares that the element is a `TransOrderType` and that the element can be substituted for the `GLogXMLTransaction` element. The `TransOrderType` extends the `OTMTransactionInOut` type which ensures that it is a valid substitution (because it in turn extends the `GLogXMLTransactionType`). The `OTMTransactionInOut` type declares that is valid for both inbound and outbound Transmissions.

The following is an example Transmission XML document containing two transaction interfaces (`TransOrder` followed by `ShipmentStatus`).



```
<?xml version="1.0" encoding="UTF-8" ?>
<Transmission xmlns="http://xmlns.oracle.com/apps/otm/transmission/v6.4">
  <TransmissionHeader>
    <SenderTransmissionNo>1234-ABCD</SenderTransmissionNo>
    <AckSpec>
      <ComMethodGid>
        <Gid>
          <Xid>HTTPPOST</Xid>
        </Gid>
      </ComMethodGid>
      <ServletURL>http://myserver/com.example.TransmissionReportReceiverServlet</ServletURL>
      <AckOption>ERROR</AckOption>
    </AckSpec>
  </TransmissionHeader>
  <TransmissionBody>
    <GLogXMLElement>
      <TransOrder>

      </TransOrder>
    </GLogXMLElement>
    <GLogXMLElement>
      <ShipmentStatus>

      </ShipmentStatus>
    </GLogXMLElement>
  </TransmissionBody>
</Transmission>
```

**Figure 2-6**

### AckSpec

The AckSpec element controls whether or not a TransmissionReport message is sent when the transactions within a Transmission have been completely processed i.e. when they have been persisted and any dependent workflow has been completed.

> **NOTE:** AckSpec is only valid for Transactional Style interfaces.

The AckOption child element specifies when to receive a TransmissionReport. If unspecified, the behavior is property controlled where the default property setting (`glog.integration.TransmissionReport=”on error”`) is equivalent to the `ERROR` option.

- ERROR = Send Transmission Report only when there are errors.
- YES = Send Transmission Report in all cases.
- NO = Do not send Transmission Report, even if there are errors.

## *Common Schema Elements and Types*

The following sections describe, at a high level, a few of the most common data elements that are likely to be present in many interfaces. More detailed information is available in Chapter 6: Inbound Transmission Processing and Chapter 7: Outbound Transmission Processing.

## Global IDs (GIDs)

GIDs are global identifiers that Oracle Transportation Management uses to define a primary key for various types of information (e.g., orders, shipments, locations, payment vouchers, etc.). A GID consists of the following two parts:

- **Domain name**: (optional). The domain name typically identifies a company or business unit and is used to separate data and secure it from other data in a shared, web-based environment. For example, if you are using Oracle Transportation Management in an environment where many companies may be using the same Oracle Transportation Management installation, the domain allows you to isolate data in Oracle Transportation Management for each company. Therefore, many users from different companies can work in the same Oracle Transportation Management installation (or web site) and use data that is private and specific to their company. If you do not include a domain name in a GID, it can be viewed across all domains in your system.
- **External ID (XID)**: (mandatory). The ID that defines the item on the external system. An external system is any system other than Oracle Transportation Management.

It is recommended to only use uppercase characters in GIDs and XIDs in your transmissions. The Oracle Transportation Management user interface normally only search for uppercase text strings and therefore it may be difficult to find records with mixed case XIDs.

**Note:** You should also be careful not to create GIDs with trailing spaces, as these records will not be retrievable via the User Interface.

## Transaction Code

Transaction Codes tell Oracle Transportation Management what to do with the transactions it receives from other systems. Transaction Code elements will have the type `TransactionCodeType` which is an enumerated type and should contain one of the following values:

- **I**: Insert. Use this transaction code to send new information to Oracle Transportation Management. Oracle Transportation Management creates a new record. If the record already exists, then the transaction will generate a "record already exists" error.
- **II**: Insert and Ignore. When used, if the record already exists, then it is not updated and the "record already exists" error message is not logged. If it does not exist, then it is inserted.
- **U**: Update. Oracle Transportation Management updates an existing record.
- **UU**: Oracle Transportation Management updates the existing record while suppressing "no data found" constraint violations.
- **IU**: Insert and Update. Oracle Transportation Management creates a new record unless it already exists, in which case Oracle Transportation Management updates the existing record with the new information.
- **UI**: Update and Insert. This works the same way as IU.
- **D**: Delete. Use this transaction code to delete an existing record.
- **DD**: Oracle Transportation Management deletes the existing record while suppressing "no data found" constraint violations.
- **NP**: No Persist. Use this transaction code to keep Oracle Transportation Management from persisting data to the database. For example, enter NP if you do not want to persist public locations. This is the default TransactionCode when the data is sent outbound from Oracle Transportation Management.
- **RC**: Replace Children. Use this transaction code to delete all child data corresponding to the top level parent, update the top level parent, and insert the new child data. You use the ReplaceChildren element to specify what child elements Oracle Transportation Management should replace. The remaining elements are processed using the IU transaction code.

- **RP**: Replace Primary/Parent. Use this transaction code to replace the primary/parent object without replacing the child objects. This will remove all fields in the primary/parent object that are not contained in the incoming xml, and will perform an insert/update on all the child data.
- **R**: Replace. Use this transaction code to replace the primary/parent and child objects. This is a combination of the RC and RP transaction codes.

### Replace Children

Replace Children content is for inbound use only and is defined in elements whose type is `ReplaceChildrenType`. It is used in conjunction with a `TransactionCode` value of "RC". By specifying the `ManagedChild` content elements, it is possible to limit which child objects are replaced by new values within the present Transmission. See the Extended Inbound Validation section for more details.

### Send Reason

Send Reason information is outbound only and is defined in elements whose type is `SendReasonType`. The information is used to show the internal application event which resulted in the generation of the outbound Transmission, for example, a SHIPMENT CREATED event.

### Integration Saved Query

Integration Saved Queries are used in inbound transmissions and are defined by elements whose type is `IntSavedQueryType`. It allows an interface to support a method of identifying an object to be modified when the objects primary key is not known. An Integration Saved Query essentially represents an SQL query that must already have been created in the user interface. The element content will then contain the GID of this query and any parameters required by the query as arguments.

See Using Integration Saved Queries for more details.

### Flex Fields

Flex fields are so called because they provide a flexible set of table columns for OTM/GTM implementers to customize their data model to suit their functional needs. They can be entered via the User Interface and/or via Integration.

Flex fields cover a number of data types and have been added to most of the major business objects and their corresponding interfaces and are available for both inbound and outbound messages.

The valid schema types are described in the following table

| **Schema Type** | **DB Data type** | **Number** | **Description** |
|---|---|---|---|
| FlexFieldStringType | VARCHAR2(150) | 20 | Holds string data |
| FlexFieldNumberType | NUMBER() | 10 | Holds numbers with any precision |
| FlexFieldCurrencyType | OTM Currency UOM | 3 | Holds financial amount |
| FlexFieldDateType | DATE | 10 | Holds date/time values |

Not all flex field types have been added to all interfaces. Please consult the relevant schema documentation to determine the support available for each business object.

In addition to the above, the RATE_GEO and RATE_OFFERING outbound interfaces have equivalent Flex Fields available which are defined slightly differently due to the database centric nature of these (outbound only) interfaces. These interfaces use the following elements:

| **Schema Element** | **DB Data type** | **Number** | **Description** |
|---|---|---|---|
| ATTRIBUTE1…n | VARCHAR2(150) | 20 | Holds string data |
| ATTRIBUTE_NUMBER1…n | NUMBER() | 10 | Holds numbers with any precision |
| ATTRIBUTE_DATE1…n | DATE | 10 | Holds date/time values |

### Date & Time

The GLogDateTimeType schema type is a platform neutral representation of the date and time to the accuracy of seconds. It is a string in the format YYYYMMDDHHMMSS where:

| | | |
|---|---|---|
| YYYY | - | The year e.g. 2012 |
| MM | - | The month e.g. 01 is January, 12 is December |
| DD | - | The day number e.g. 01 to 31 |
| HHMMSS | - | The time in 24 hour clock e.g. 231530 |

An example value is "20121031083030" which represents October 31st, 2012 at 30 seconds after 8.30a.m.

Additionally, the GLogDateTimeType can hold a Time Zone ID element (TZId) and a Time Zone Offset. When date elements are inbound the Time Zone ID is used to calculate times relative to the system time. On outbound messages, the Time Zone Offset will be populated to show the offset in hours from the system Time Zone compared to the target system locale. See Chapter 6: Inbound Transmission Processing, Time Zones section for details.

## *Backward Compatibility*

To support backward compatibility, the version 6.4.1 GLogXML and GLogXML-GTM schemas will continue to be distributed with version 6.4.2 but will only describe XML messages for versions up to but **not** including version 6.4.2.

However, all message formats currently accepted by versions prior to version 6.4.2 will continue to be accepted in version 6.4.2, but access to features new in OTM/GTM version 6.4.2 schemas is only supported by sending version 6.4.2 messages.

The following table lists the namespace URLs and associated physical file name for each schema for versions prior to version 6.4.2:

| **XSD Schema File** | **Target Namespace** |
|---|---|
| GLogXML.xsd | http://xmlns.oracle.com/apps/otm |
| GLogXML-GTM.xsd | http://xmlns.oracle.com/apps/gtm |

As was the case for all previous versions of OTM and GTM, XML messages are accepted without a namespace declaration. However, as there are now two possible 'assumed' namespaces that could apply to such an XML message, there is a new 'version' attribute available on the Transmission defined in 6.4.2. This should be used to contain the string value '6.4.2' when an XML Transmission without namespaces, but is otherwise valid according to the Transmission schemas, is sent into OTM/GTM.

All future 6.4.x versions of OTM and GTM will use the namespace URL introduced in version 6.4.2, namely "`http://xmlns.oracle.com/apps/otm/transmission/v6.4`".

The next major or minor release of OTM and GTM will introduce a new namespace version, for example, "http://xmlns.oracle.com/apps/otm/transmission/v6.5" or "http://xmlns.oracle.com/apps/otm/transmission/v7.0".

> **Note:** These URLs are just for illustration purposes and do not refer to actual or planned releases.

Therefore for any schema modifications in future 6.4.x versions which could have backward compatibility issues e.g. the removal of an element, the following process will be followed:-

- Element to be removed will be marked as deprecated but will still be accepted. Unless there are exceptional circumstances, functional equivalence will be retained.
- When a new namespace URL is released, the deprecated elements will be completely removed from the new schema.

## Processing for Inbound and Outbound Compatibility

All messages valid before version 6.4.2 will continue to be supported. On receipt of an inbound Transmission which refers to the pre-6.4.2 target namespace, the XML format will be transformed internally into the corresponding 6.4.2 namespace. See the XML Interface Changes Guide for a complete list of changes in version 6.4.2.

The transformation from old to new or vice versa is performed by the Version Conversion framework. This framework uses a set of properties (default values are in `glog.integration.properties` file but can be overridden in `glog.properties`) which map to XSLT stylesheets that will transform the XML.

The default inbound and outbound transmission properties are:

```
glog.integration.versionconverter.to.PRE642_Transmission=db:TO-PRE-642-
TRANSMISSION
glog.integration.versionconverter.from.PRE642_Transmission=db:FROM-PRE-642-
TRANSMISSION
glog.integration.versionconverter.to.NONE_Transmission=db:TO-NOXMLNS-
TRANSMISSION
glog.integration.versionconverter.from.NONE_Transmission=db:FROM-NOXMLNS-
TRANSMISSION
glog.integration.versionconverter.to.NONE_PRE642_Transmission=db:TO-PRE-642-
TRANSMISSION,db:TO-NOXMLNS-TRANSMISSION
glog.integration.versionconverter.from.NONE_PRE642_Transmission=db:FROM-
NOXMLNS-PRE642-TRANSMISSION,db:FROM-PRE-642-TRANSMISSION
```

Inbound transformations are based on the `glog.integration.versionconverter.from` properties. Outbound transformations are based on the `glog.integration.versionconverter.to` properties.

The value of the property is a Stylesheet Content GID (prefixed with "db:") which will contain the XML for the stylesheet templates.

For example, if a pre-6.4.2 Transmission is sent inbound to a 6.4.2 server, the key generated for the version check will be "from.PRE642_Transmission" and the XSLT contained in the Stylesheet Content record for GID "FROM-PRE-642-TRANSMISSION" will be used to transform the message to a version 6.4.2 message.

In some scenarios multiple stylesheets can be applied. For example, when a pre-6.4.2 Transmission is sent inbound but has no XML namespaces association, two stylesheets are applied in sequence. The first stylesheet (FROM-NOXMLNS-PRE642-TRANSMISSION) transforms the original XML to have the correct XML namespace for a pre-6.4.2 version Transmission. The second stylesheet (FROM-PRE-642-TRANSMISSION) transforms the result from the first transformation to a final format which matches the version 6.4.2 namespace format.

Also, if a pre-6.4.2 Transmission is sent inbound any TransmissionAck response will also be in the same pre-6.4.2 format.

A corresponding conversion happens on outbound interface messages but obviously in the reverse order. See chapter 7, Controlling Target Namespace for details on how to declare which transformation, if any, should apply.

> **Note:** In versions prior to 6.4.2 it was possible to send in XML which would normally be considered invalid with respect to namespaces and for it to be processed. This was because the first step in processing XML was to strip all namespace information. In order to enforce tighter validation, and therefore security, namespaces are no longer ignored and therefore it is no longer acceptable to send invalid XML. An example of invalid XML would be if some elements declared a namespace while others did not. If **any** namespaces are declared they must be consistent and valid.
>
> If any existing interface has been designed to depend on an invalid format it may be necessary to modify the transformation logic to support continued operation while the external process can be modified to send valid messages. The PUBLIC stylesheet content should not be modified. The correct process will be to create new stylesheet content records and modify the relevant properties (via custom settings in the `glog.properties` file) to reference the custom GID. Preferably this custom GID should be inserted before (inbound) or after (outbound) the sequence of current GIDs.

### Deprecated or Removed Interfaces

The following interfaces will still be supported for pre-6.4.2 message formats but have been removed from the version 6.4.2 schema definitions and are therefore considered deprecated. It is not valid to include them in a version 6.4.2 message.

- **CSVFileContent**: This capability has been superseded in version 6.4.2 with the CSVDataLoad interface. Please convert any CSVFileContent integrations to use the new CSVDataLoad interface.
- **GtmShipment**: The Customs Shipment interface is now based on the GtmDeclaration interface. Any pre-6.4.2 GtmShipment interface will be internally converted to a corresponding GtmDeclaration message. Please convert any GtmShipment integrations to use the new GtmDeclaration message.

# 3. Supported Transport Protocols

There are a number of transport protocol options available when sending messages to or from Oracle Transportation Management and Global Trade Management. The available options can vary for inbound versus outbound and whether the message is a Transmission XML or a Mobile Device Message XML.

The following table shows all valid combination (Y – available, N – not available):

| Transport Protocol | Inbound | Outbound | Supports Transmission | Supports Message |
|---|---|---|---|---|
| SOAP Web Services | Y | Y | Y | Y |
| HTTP POST | Y | Y | Y | Y |
| Oracle Advanced Queue | Y | Y | Y | N |
| Direct DB Insert | Y | N | Y | Y |
| FTP | N | Y | Y | Y |

The protocol specific details e.g. resource URLs are covered in each of the following sections:

- Security
- HTTP Integration
- Oracle Advanced Queue Integration
- Direct Database Integration
- FTP Integration

**NOTE:** Mobile Device Communication protocol details are covered in Chapter 17: Mobile Device Communications. In addition to the protocols listed above it is possible to upload XML files using the Integration Manager User Interface. This is a useful tool when testing new interfaces or diagnosing functional issues that are normally triggered via integration. Please refer to the online Help for usage instructions.

# 4. Security

Access to Integration functionality is restricted to authenticated and authorized users only.

User credentials are passed in the message in a protocol specific way and so is covered in detail in each of the following sections:

- Web Service Integration
- HTTP Integration
- Oracle Advanced Queue Integration
- Direct Database Integration
- FTP Integration

**NOTE:** The capability to pass user credentials in the message payload (Transmission Header) has been deprecated as of version 6.4.2 and will be removed completely in a future version. It may be removed gradually and at different times based on transport protocol. For example, as of version 6.4.2 support for it is no longer available for inbound Web Service or Oracle Advanced Queue integration.

Access Control can be granted on a granular level e.g. to a specific service or servlet resource or can be granted to the entire ‘INTEGRATION’ function which includes all integration resources.

Please consult the Oracle Transportation Management Security Guide for details on how to configure the required access control for your integration user accounts.

# 5. Available Interfaces

This chapter lists each schema file and a description of the interfaces defined within it. The schema file will contain the detailed definition and documentation for each interface.

**Shipment.xsd**

| **Interface** | **Description** | **Inbound** | **Outbound** |
|---|---|---|---|
| ActualShipment | Third parties send actual shipments to Oracle Transportation Management, which define the working or final form of a shipment. | Y | N |
| PlannedShipment | Oracle Transportation Management builds planned shipments and can send them to involved parties, to service providers as part of the tender process or to a warehouse management system (WMS). | N | Y |
| ShipmentStatus | Service providers and other third parties transmit shipment event information to Oracle Transportation Management with the ShipmentStatus interface. Shipment status information describes activity on shipments and shipment groups. For example, you can use shipment statuses to determine whether a shipment is running on time or whether it is late. In addition, you can send vessel and flight information, and equipment service provider and order information. Based on shipment status information it receives, Oracle Transportation Management can re-plan a shipment. | Y | Y |
| SShipUnit | Queries for and updates a shipment ship unit without direct knowledge of the owning shipment. | Y | N |
| ShipmentLink | Identifies related shipments at a consolidation or de-consolidation pool for both inbound and outbound interfaces. | Y | N |
| ShipStop | Used to modify specific stop related information for a shipment as an alternative to sending the whole shipment. | Y | N |
| Equipment | Defines the equipment that can be assigned to a shipment. | Y | Y |
| ShipmentGroup | Used to define a group of shipments. | Y | Y |

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| TenderOffer | Once a shipment has been planned, Oracle Transportation Management sends a tender offer to a service provider. The tender offer provides a contract for the service provider to carry a particular shipment. | N | Y |
| TenderResponse | Service provider response to accept or decline a previous TenderOffer interface message. | Y | N |
| ShipmentGroupTenderOffer | Used to send a tender offer for a group of shipments to a service provider. | N | Y |
| BookingLineAmendment | Used to send booking line changes out of the system. | N | Y |
| Voyage | Used to send world-wide vessel schedule information to external systems. Voyage schedules define the rotation of a carrier's vessel as the vessel goes from a set of loading points (departure ports) to a set of unloading ports (arrival ports). | N | Y |
| DemurrageTransaction | Used to communicate the fees charged (demurrage) by a carrier to a customer for the use of assets beyond a contracted free time for loading and unloading. | N | Y |
| CharterVoyage | Represents an ocean transport movement by a carrier from a loading port to a discharge port. | Y | Y |
| Consol | Used to specify the shipment consolidator. A consol can be created for a charter voyage or air schedule (flight). | Y | Y |
| Driver | Sends and receives driver data to Oracle Transportation Management which includes basic personal information, CDL information, driver types, driver team data, status, remarks, etc. | Y | Y |
| PowerUnit | Sends and receives power unit (such as a tractor) data to Oracle Transportation Management. This information includes power unit name, power unit type, special services, remarks, etc. | Y | Y |
| Device | Used to send device logon and logoff messages to Oracle Transportation Management. | | |

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| DriverCalendarEvent | Specifies a driver calendar event, which describes calendar event information on a driver. | Y | N |
| WorkInvoice | Includes a record of driver activities for a shipment. This is the record sent to payroll and used to calculate driver pay. | Y | Y |

**Order.xsd**

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| TransOrder | Oracle Transportation Management receives transportation orders from external systems. These orders can include basic information such as IDs, pick-up and delivery dates, service providers, and details such as ship units or line items. It represents orders which can be shipped as one or more ‘released’ amounts called an Order Release. | Y | Y |
| TransOrderLink | Used to establish a link between order base objects. Provides the ability to maintain orders in various states along with their relationships. | Y | N |
| Release | Transmits Order Release information to and from Oracle Transportation Management. An order release represents the shippable portion of a transportation order. | Y | Y |
| TransOrderStatus | Sets order base status events. | Y | N |
| ReleaseInstruction | Allows you to specify line items and ship units, and release specific quantities of them for a particular order base. | Y | N |
| OBShipUnit | Contains information on ship units in an order base. | N | Y |
| OBLine | Contains information on lines in an order base. | N | Y |
| OrderMovement | Represents part of the routing for an order. It is also a collection of ship units that are unplanned for certain part of a route. | Y | Y |

**GTM.xsd**

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| GtmContact | Represents a Global Trade Management party. | Y | Y |
| GtmTransaction | Represents a Global Trade Management trade transaction for the screening of shipment and/or order release information. | Y | Y |
| GtmTransactionLine | Represents a Global Trade Management trade transaction line. | N | Y |
| GtmRegistration | Represents the registrations existing in external systems that are relevant for trade purposes. GTM supports different type of registrations: for parties (e.g. legal entities, company branches, customers, carriers, etc.) and for items. | Y | Y |
| GtmStructure | Represents Trade Item Structure (bill of material) to identify components that make up a single sellable product or item. | Y | N |
| GtmDeclaration | Represents a Global Trade Management Customs Declaration. Examples could be: import shipment, export shipment, inter-state shipment. The interface can be also used to pass the shipments information including charges and relevant information to an external system such as an ERP Inventory Receiving or Landed Cost Management application. | Y | Y |
| GtmDeclarationMessage | This Element maintains the information related to messages exchanged between different parties involved in the process of a Customs Declaration. | | |
| GtmBond | Represents bond or guarantee related information. | Y | N |
| ServiceRequest | Request/Reply style interface to request the execution one of a number of services:-<br>• Compliance Rule Screening<br>• License Determination<br>• Item Classification Screening<br>• Restricted Party Screening<br>• Sanctioned Territory Screening | | |
| ServiceResponse | Request/Reply style interface reply to the ServiceRequest. | N | Y |

**Finance.xsd**

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| Accrual | Used to transmit the allocated freight cost accrued or paid against orders. This can be used to communicate changes or differences in an order's allocated freight cost to other external systems. These differences, for example, could be used to establish liabilities in an accounting or general ledger system. | N | Y |
| Billing | Sends transmissions to an accounting system for customer billing. The billing information represents the amount owed by a customer to the planner of a shipment. The billing transmission includes the customer who is being charged and details of the bill such as the amount due, the date due, and any discount information. | N | Y |
| AllocationBase | Sends order release allocation cost information to an order owner. Allocation is a method for dividing the cost of a shipment among its order releases based on the line items and ship units on the shipment. Messages can be sent for planned and actual shipments. | N | Y |
| Invoice | Represents what is owed to the service provider or other third parties for transporting the shipment. The message content contains details about the payment and the specific transportation activity for which payment is owed. | Y | N |
| Voucher | Transmits payment information. A voucher represents the cost of a shipment owed to a third party such as a service provider and is designed to be sent after it has matched a third party charge to a shipment and approved the charge for payment. | N | Y |
| FinancialSystemFeed | Sends billing and shipment cost allocation information to an external financial system based on shipments (buy or sell). Can be used as an alternative to using Invoices or Bills as the source for financial information. | N | Y |
| Quote | The Quote Interface allows customer service representatives to supply their customers with transportation quotes. | N | Y |
| Claim | Used to specify information for damaged shipments. A claim consists of any freight damage that occurs to a shipment prior to taking ownership/responsibility of the shipment. | Y | Y |

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| WorkInvoice | Includes a record of driver activities for a shipment. This is the record sent to payroll and used to calculate driver pay. | Y | Y |
| ExchangeRate | Exchange Rate specifies the effective and expiration date for a set of currency conversions. | Y | Y |

**LocationContact.xsd**

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| Location | A place where transportation related activities, such as loading and unloading freight, occur. In addition, a location is a corporation, and/or a service provider. Use the location element to transmit location information, for the Transportation Orders interface. | Y | Y |
| Contact | Defines a person or party that can be contacted through Oracle Transportation Management. | Y | Y |
| ContactGroup | Represents a list of contacts used for notification | Y | Y |
| Corporation | | | |
| ActivityTimeDef | This interface is used to specify fields for allowing multiple fixed and variable stop times for each location role based on transport handling unit and commodity. | Y | Y |

## Item.xsd

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| ItemMaster | Transmits item master data to Oracle Transportation Management. Item master data represents the freight being shipped and includes :<br>• Item numbers<br>• Descriptions<br>• Packaging details<br>• Applicable NMFC, STCC, SITC, or HTS codes<br>• General ledger ID<br>• Accessorial charges<br>Item master data must exist in Oracle Transportation Management before you can create transportation orders. | Y | Y |
| Item | Transmits item data to Oracle Transportation Management. Differs from Item master interface in that it only relates to Item and not Packaging. | Y | Y |
| HazmatItem | Sends and receive records for particular hazardous items. | Y | Y |
| HazmatGeneric | Transmits hazardous material based on the shipping name for an item. | Y | Y |
| Sku | Defines a stock keeping unit including what quantities to keep in stock, and the actual amount in the warehouse. | Y | Y |
| SkuTransaction | Represents a shipment of SKUs arriving or leaving the warehouse. | Y | N |
| SkuEvent | Specifies a SKU event, which describes activities on SKU's | Y | Y |

## Planning.xsd

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| Itinerary | Define the path between two locations and specifies the constraints for building the shipment. | Y | N |
| XLane | Defines a link from a source to a destination. The source and destination may specify either general or specific geography. For example, a source could be an exact location, or an entire state. May also contain mileage information. | Y | N |

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| Mileage | Defines the distance between points for a particular lane. Lanes may contain specific or general geographic information, such as:<br>• Zip code to zip code<br>• City to city<br>• State to state<br>• Zip code to city<br>• City to state<br>• Address to address<br>• City to address<br>• State to address | Y | N |
| Schedule | Sends schedules as input to the processes for building shipments and assigning orders into batches. | Y | Y |
| RouteTemplate | Route template represents the plan for a cooperative route. A cooperative route is a linking of lanes that have been identified to have sufficient recurring volume of shipments to form a good route for a fleet or dedicated vehicle. | Y | Y |
| BulkPlan | Describes the orders that Oracle Transportation Management planned and the shipments that Oracle Transportation Management created. | N | Y |
| BulkRating | Describes rating statistics on the orders that Oracle Transportation Management planned and the shipments that Oracle Transportation Management created. | N | Y |
| BulkTrailerBuild | Describes the shipment groups created during the bulk trailer build process. | N | Y |
| BulkContMove | Describes the shipments that were selected and linked during a given run of bulk continuous move. | N | Y |
| FleetBulkPlan | This is used to provide statistics about the fleet resource assignments that were created to the planned shipments during a given run of fleet bulk plan. | N | Y |
| ServiceTime | Transmits the time it takes to go between the two locations of a Lane. Service time information can also be included as part of the Mileage interface. | Y | N |

### Configuration.xsd

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| User | Represents an Oracle Transportation Management user. | Y | N |

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| CSVDataLoad | CSVDataLoad provides the capability to embed the contents of a CSV file for insertion into the database. Each CSVDataLoad element can contain only one CSV File. This element should only be used for small sets of data. | Y | N |

### Document.xsd

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| Document | Provides a consistent way to send and receive business documents in and out of the system. Business documents are objects that contain the contents of a traditional document, like a bill of lading for example, in electronic format. | Y | Y |

### GenericTransaction.xsd

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| Topic | Raises a topic and gets Oracle Transportation Management to start processing an object. | Y | N |
| GenericStatusUpdate | Updates properties – status, remark, reference number and indicator – of various business objects including Location, Order Base, and Order Release etc. See schema documentation for a complete list. | Y | N |
| RemoteQuery | Queries Oracle Transportation Management for<br>• Rating information, based on service provider, transportation mode, quantity, locations, and/or dates of a shipment.<br>• Shipment information<br>• Transmission Report – the processing result of a previous transmission. | Y | N |
| RemoteQueryReply | Provides Oracle Transportation Management's response to a RemoteQuery interface message | N | Y |
| TransactionAck | An external system sends a TransactionAck to acknowledge receipt and processing of a transaction. | Y | N |
| DataQuerySummary | Sends a summary of the data required by an external system. | N | Y |

**Job.xsd**

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| Job | Sends data that addresses how logistics services providers and freight forwarders manage the services they provide within Oracle Transportation Management. A Job offers a workspace that brings together the objects and activities required of them. | Y | Y |

**Rate.xsd**

| Interface | Description | Inbound | Outbound |
|---|---|---|---|
| RATE_OFFERING | Represents a general contract with a service provider. It indicates what rate offering data was used to rate the shipment. | N | Y |
| RATE_GEO | Provides specific costing or rating data from one place to another. It indicates what rate record data was used to rate the shipment. | N | Y |

# 6. Inbound Transmission Processing

The internal processing of inbound transmissions is highly configurable and can support a wide range of requirements from the very simple single object Create, Retrieve, Update and Delete (CRUD) interfaces through to multi-message groups and flexible sequencing involving complex workflow.

The following steps outline the default process for a simple Transactional style transmission. The various configuration options are described in the subsequent sections.

## Processing Overview

On receipt of a Transmission XML, the integration application module performs the following steps. Note that authentication and authorization will already have been performed before the message is passed to the module:

1. Parse Transmission XML, generate a new unique Transmission Number and persist records in the I_TRANSMISSION and I_TRANSACTION tables for this transmission number.
2. Depending on transport protocol used, and if requested, create a TransmissionAck XML message and return to the sender.
3. Place a message on the internal Integration Workflow queue to say that a new transmission requires processing.
4. The Integration Workflow retrieves the message from the internal queue and:-
    a. Determines any grouping or sequencing constraints
    b. Submits group(s) to Mediator for processing.
5. When the Mediator selects a group for processing, it will process each Transaction in the group as follows:
    a. If extended validation is configured the integration module validates the following:
        i. Foreign keys, for example a Location GID must exist in the Location table.
        ii. Data Types, for example a number only contains numeric characters.
        iii. TransactionCode. If set to I, the integration module checks that the primary key does not exist. If set to U or D, the integration module checks that the primary key does exist.
        iv. Required Elements are not NULL.
    b. Saves any validation errors.
    c. If there are no validation errors or if extended validation was not configured, the corresponding business object is populated with the data from the XML and passed to the persistence module.
    d. If there are agents listening for pre-persist events, the persistence module triggers those now. An example of this is the public Order Base - Insert agent for TransOrder transmissions.
    e. The business objects are now persisted to the corresponding database tables. If extended validation had not been configured all integrity constraints are now checked and any violations would cause the transaction to fail.
    f. If there are agents listening for post-persist events, the persistence module triggers those.
6. When the Mediator has processed all transactions for all groups in a Transmission the Transmission processing is complete. The completion process optionally sends a TransmissionReport with validation and processing errors. See AckSpec for details.

**Note:** The Transmission Report is only physically transmitted when the all workflow that is considered to be part of the Transmission Process has completed (known as "child" processes). If there is a significant amount of agent workflow configured this could appear as a delay in delivering the report. If this is an issue it may be worth considering the use of custom events with the "Create New Process" option selected.

## Transmission Status

Each Transmission sent into Oracle Transportation Management has a status field that indicates the state of the Transmission at each stage in the processing. The state of the Transmission will change as it progresses through the various steps outlined in Processing Overview. You can view the status for a Transmission in the Transmission Manager UI that can be accessed via the following menu: **Business Process Automation > Integration > Transmission Manager**. The status of the transmission could be one of the following:

| Status | Description |
|---|---|
| STAGED | Indicates that the Transmission has been parsed and is stored in the transmission and transaction tables. An internal event will also be queued which, when triggered, will process the transmission content. No Transmission should stay in a STAGED status permanently. Such a scenario would normally suggest an application configuration issue. |
| RECEIVED | Indicates that a Transmission has been persisted but it has not yet been processing required to move it to STAGED status. This status should only appear temporarily for messages sent via DB Insert or OAQ until the relevant internal queue (Data Queue or App Server thread) can process it. No Transmission should stay in a RECEIVED status permanently. Such a scenario would normally suggest an application configuration issue. |
| STAGING | Indicates that the Transmission has been persisted and is waiting on a condition to be released for processing. In this status, additional transactions can be added to it prior to processing. Refer to Inbound Transmission Staging and Processing for additional details. |
| FRESH | Indicates that the Transmission is waiting to complete processing. The individual transactions may still be processing. |
| ERROR | Indicates that the Transmission had completed processing and there were errors in the processing. |
| PROCESSED | Indicates that the Transmission successfully completed processing. |
| REDO | Indicates that the Transmission is waiting for the REDO logic to initiate re-processing of it. Please refer to the online help for REDO processing. |

## Extended Inbound Validation

After Oracle Transportation Management has processed a transmission, Oracle Transportation Management sends back a Transmission Report to the external system with a list of validation and/or processing errors.

System administrators can set the default level of extended validation that Oracle Transportation Management performs. Changing the validation level can improve performance by removing unnecessary queries and logic for validating the data. The default level of validation is specified via the following "`glog.integration.validation`" property. Refer to the online help for the list of possible values for the property.

If your Oracle Transportation Management installation validates all transmissions fully, you can skip validation for certain transmissions on a case-by-case basis. Just include this processing instruction in all transmissions where Oracle Transportation Management should skip validation:

```
<?gc3-xml-process validate_required_fields="N"?>
```

If instead your Oracle Transportation Management installation never validates errors or only validates when receiving persist errors from the Oracle Database, you can get Oracle Transportation Management to validate certain transmissions on a case-by-case basis. Include this processing instruction in all transmissions Oracle Transportation Management should validate:

```
<?gc3-xml-process validate_required_fields="Y"?>
```

## Case Sensitive Data

Oracle Transportation Management provides functionality to automatically change text to upper case when processing the inbound XML. You can enable the functionality by setting the following property:

```
glog.integration.enableCaseChange=true
```

If there are some elements that you do not want to change, you can set the `glog.integration.casechange.element` property for those elements. Valid values are element names. For example, to prohibit the ArgName and ArgValue values from changing, you would define the following:

```
glog.integration.casechange.element=ArgName
glog.integration.casechange.element=ArgValue
```

## ‘Replace’ Transaction Codes

The following example Shipment XML structure illustrates the basic rules for transaction codes starting with “R”.

```
Shipment
    ShipmentHeader
            TransactionCode RC
    ShipmentStop=1
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipmentStop=2
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipUnit: ShipUnitGid=ShipUnit A
            ShipUnitContent: LineNumber=0
            ShipUnitContent: LineNumber=1
    ShipUnit: ShipUnitGid=ShipUnit B
            ShipUnitContent: LineNumber=3
            ShipUnitContent: LineNumber=4
    SEquipment: SEquipment_A
```

If the data is not found in database, the shipment is inserted into database. In this case, the RC is equivalent to IU. The use cases described in the following sections are edited from this XML.

1. Remove ShipmentStop 2 and two ShipUnitContents of ShipUnit A from the above XML.
    - Expected result: ShipmentStop 2, two shipUnitContents as well as their corresponding children are deleted from database.

```
Shipment
    ShipmentHeader
            TransactionCode RC
    ShipmentStop=1
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    <!-- ShipmentStop=2
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B -->
    ShipUnit: ShipUnitGid=ShipUnit A
            ShipUnitContent: LineNumber=0
            ShipUnitContent: LineNumber=1
    ShipUnit: ShipUnitGid=ShipUnit B
            <!--ShipUnitContent: LineNumber=3
            ShipUnitContent: LineNumber=4 -->
    SEquipment: SEquipment_A
```

2. Remove ShipmentStop 2 from the original XML and add ManagedChild=ShipmentStop.
    - Expected result: ShipmentStop 2 and all its child tables are deleted from database.

```
Shipment
    ShipmentHeader
            TransactionCode RC
            ManagedChild =ShipmentStop
    ShipmentStop=1
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    <!-- ShipmentStop=2
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B -->
    ShipUnit: ShipUnitGid=ShipUnit A
            ShipUnitContent: LineNumber=0
            ShipUnitContent: LineNumber=1
    ShipUnit: ShipUnitGid=ShipUnit B
            ShipUnitContent: LineNumber=3
            ShipUnitContent: LineNumber=4
    SEquipment: SEquipment_A
```

If the Is_permanent of the above stop in database equals true or the ManagedChild is set to a value other than ShipmentStop or ShipmentStopDetail, the stop as well as it child tables are not able to be removed from database.

3. Remove ShipmentStopDetail elements of ShipmentStop 2 and add ManagedChild = ShipmentStop from the original XML.
    - Expected result: ShipmentStopDs of ShipmentStop 2 are deleted from database.

```
Shipment
    ShipmentHeader
            TransactionCode RC
            ManagedChild =ShipmentStop
    ShipmentStop=1
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipmentStop=2
            <!-- ShipmentStopDetail: ShipUnitGid=ShipUnit A
```

```
        ShipmentStopDetail: ShipUnitGid=ShipUnit B -->
    ShipUnit: ShipUnitGid=ShipUnit A
        ShipUnitContent: LineNumber=0
        ShipUnitContent: LineNumber=1
    ShipUnit: ShipUnitGid=ShipUnit B
        ShipUnitContent: LineNumber=3
        ShipUnitContent: LineNumber=4
    SEquipment: SEquipment_A
```

You can set ManagedChild=ShipmentStopDetail in order to get the same result.

4. Remove ShipmentStopDetail elements of ShipmentStop 2 and ShipUnitContent elements of ShipUnit A with ManagedChild= ShipmentStop from the original XML.
    - Expected result: ShipmentStopDs for ShipmentStop 2 are deleted. SShipUnitLines are unchanged. In theory, SShipUnitLines should be deleted too. However, this is an exception case since ShipUnit, SEquipment and Text are not really child or grandchild nodes of shipment. They can independently exist in database. In order to delete ShipUnitContent, you have to specify the ShipUnitContent in ManagedChild as described in next section

```
Shipment
    ShipmentHeader
        TransactionCode RC
        ManagedChild =ShipmentStop
    ShipmentStop=1
        ShipmentStopDetail: ShipUnitGid=ShipUnit A
        ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipmentStop=2
        <!-- ShipmentStopDetail: ShipUnitGid=ShipUnit A
        ShipmentStopDetail: ShipUnitGid=ShipUnit B -->
    ShipUnit: ShipUnitGid=ShipUnit A
        <!-- ShipUnitContent: LineNumber=0
        ShipUnitContent: LineNumber=1 -->
    ShipUnit: ShipUnitGid=ShipUnit B
        ShipUnitContent: LineNumber=3
        ShipUnitContent: LineNumber=4
    SEquipment: SEquipment_A
```

5. Remove ShipUnitContent elements of ShipUnit A with ManagedChild=ShipUnit or ManagedChild=ShipUnitContent.
    - Expected result: The SShipUnitLines 0 and 1 are deleted from database. SShipUnit in the XML is replaced.

```
Shipment
    ShipmentHeader
        TransactionCode RC
        ManagedChild = ShipUnitContent
            OR
        ManagedChild = ShipUnit
    ShipmentStop=1
        ShipmentStopDetail: ShipUnitGid=ShipUnit A
        ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipmentStop=2
        ShipmentStopDetail: ShipUnitGid=ShipUnit A
        ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipUnit: ShipUnitGid=ShipUnit A
        <!-- ShipUnitContent: LineNumber=0
        ShipUnitContent: LineNumber=1 -->
    ShipUnit: ShipUnitGid=ShipUnit B
        ShipUnitContent: LineNumber=3
```

```
            ShipUnitContent: LineNumber=4
    SEquipment: SEquipment_A
```

6. Add two or more ManagedChild elements (ManagedChild=ShipmentStop, ManagedChild=ShipUnit) and remove ShipmentStopDetail elements in ShipmentStop 2 and ShipUnitContent elements of ShipUnit A.
    - Expected result: ShipmentStopDs of ShipmentStop 2 and SShipUnitLines of ShipUnit A are unchanged.

```
Shipment
    ShipmentHeader
            TransactionCode RC
            ManagedChild =ShipmentStop
            ManagedChild =ShipUnit
    ShipmentStop=1
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipmentStop=2
            <! -- ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B -->
    ShipUnit: ShipUnitGid=ShipUnit A
            <!-- ShipUnitContent: LineNumber=0
            ShipUnitContent: LineNumber=1 -->
    ShipUnit: ShipUnitGid=ShipUnit B
            ShipUnitContent: LineNumber=3
            ShipUnitContent: LineNumber=4
    SEquipment: SEquipment_A
```

7. Remove ShipUnit A and the ShipUnitContent 3 and 4 for the remaining ShipUnit B with ManagedChild=ShipUnit.
    - Expected result: SShipUnit corresponding to ShipUnit A is unchanged. SShipUnit corresponding to ShipUnit B is replaced. The SShipUnitLines corresponding to ShipUnitContents 3 and 4 are deleted from database.

```
Shipment
    ShipmentHeader
            TransactionCode RC
            ManagedChild = ShipUnit
    ShipmentStop=1
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipmentStop=2
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    <!-- ShipUnit: ShipUnitGid=ShipUnit A
            ShipUnitContent: LineNumber=0
            ShipUnitContent: LineNumber=1 -->
    ShipUnit: ShipUnitGid=ShipUnit B
            <!-- ShipUnitContent: LineNumber=3
            ShipUnitContent: LineNumber=4 -->
    SEquipment: SEquipment_A
```

- The same result can be achieved through specifying the transaction code R and ManagedChild = ShipUnitContent in ShipUnit B.

```
Shipment
    ShipmentHeader
            TransactionCode RC
    ShipmentStop=1
```

```
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipmentStop=2
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    <!-- ShipUnit: ShipUnitGid=ShipUnit A
            ShipUnitContent: LineNumber=0
            ShipUnitContent: LineNumber=1-->
    ShipUnit: ShipUnitGid=ShipUnit B
            TransactionCode= R
            ManagedChild = ShipUnitContent
            <!-- ShipUnitContent: LineNumber=3
            ShipUnitContent: LineNumber=4 -->
    SEquipment: SEquipment_A
```

8. Change ManagedChild = SEquipment with TransactionCode= RC in ShipmentHeader.
    - Expected result: SEquipment_A data in database will be replaced with SEquipment data.

```
Shipment
    ShipmentHeader
            TransactionCode RC
            ManagedChild = SEquipment
    ShipmentStop=1
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipmentStop=2
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipUnit: ShipUnitGid=ShipUnit A
            ShipUnitContent: LineNumber=0
            ShipUnitContent: LineNumber=1
    ShipUnit: ShipUnitGid=ShipUnit B
            ShipUnitContent: LineNumber=3
            ShipUnitContent: LineNumber=4
    SEquipment: SEquipment_A
```

9. Remove ShipmentStop 2 with ManagedChild=ShipmentStop from the original XML and change the transaction code to R.
    - Expected result: ShipmentStop 2 and all its child tables are deleted. The shipment table is replaced (This is different from transaction code RC).

```
Shipment
    ShipmentHeader
            TransactionCode RC
            ManagedChild =ShipmentStop
    ShipmentStop=1
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipmentStop=2
            ShipmentStopDetail: ShipUnitGid=ShipUnit A
            ShipmentStopDetail: ShipUnitGid=ShipUnit B
    ShipUnit: ShipUnitGid=ShipUnit A
            ShipUnitContent: LineNumber=0
    SEquipment: SEquipment_A
```

10. Change the transaction code to RP in the XML above.
    - Expected result: Only shipment table is replaced. The ManagedChild element is ignored. ShipmentStop 2 and all its child tables are unchanged.

```
Shipment
   ShipmentHeader
         TransactionCode RC
         ManagedChild =ShipmentStop
   ShipmentStop=1
         ShipmentStopDetail: ShipUnitGid=ShipUnit A
         ShipmentStopDetail: ShipUnitGid=ShipUnit B
   <!-- ShipmentStop=2
         ShipmentStopDetail: ShipUnitGid=ShipUnit A
         ShipmentStopDetail: ShipUnitGid=ShipUnit B -->
   ShipUnit: ShipUnitGid=ShipUnit A
         ShipUnitContent: LineNumber=0
   SEquipment: SEquipment_A
```

The value for element ManagedChild is defined in CHILD_ELEMENT_ALIAS column of INT_MANAGED_CHILDREN_MAP table. This value can be viewed or edited from Power Data page found by navigating to **Business Process Automation > Power Data > Integration > Managed Children Map**.

## Time Zones

### *Time Zone Override*

Generally, time information sent in the inbound XML to Oracle Transportation Management is associated with a time zone for a particular location and time zone offset. For example, the early pickup dates and late delivery dates on a transportation order line (within the TransOrder interface) are associated with the ship from and ship to locations for the order line, and each of those locations is associated with a specific time zone. The time information sent in the early pickup date and late delivery dates elements are assumed to be in the time zone of the location. So if your time is at 8 AM and the location is King of Prussia, PA, you would specify the time in the XML as 080000 (HHMISS, where HH=Hours, MI=Minutes and SS=Seconds) and Oracle Transportation Management would understand this to mean 8AM in the America/New York time zone.

For systems that only maintain their times in a single time zone, or in other circumstances when you are unable to specify times as expected in the time zone of the corresponding locations, an override can be specified to indicate the time zone for all times for the transaction. As an example, if your sending application maintains all times in a specific time zone such as San Francisco where the early pickup date is maintained as 5 AM, it would be possible to send 050000 as the time to Oracle Transportation Management and indicate an override that all the times in the transportation order are specified in the America/Los_Angeles time zone.

A valid time zone override can be specified in the TimeZoneGid element within the Transaction Header. When the TimeZoneGid is specified, it will be assumed that all the times within the transaction (GLogXMLElement) are in that time zone.

## Business Number Generator (BNG)

When creating new objects i.e. with the 'I' TransactionCode, it is possible to send a transmission without entering a value in the primary key GID element (for example, Ship Unit ID, Order Release ID, Order Base ID, etc.). Oracle Transportation Management generates values for these fields based on the default business number rules for the object type.

## Using Null Values

Null values in XML messages are usually represented by the absence of the corresponding element. However, when the desire is to change an elements current value to null a different approach is required.

In OTM XML messages, to delete (null out) values from certain fields in a record (without deleting the whole record), you can specify the (~) character in the element. For example, if a value was entered for the External System ID field in the `TransOrderHeader`, and that value needed to be removed in a subsequent TransOrder update, the following would be specified for the XML:

```
<ExternalSystemId>~</ExternalSystemId>
```

## Using Integration Saved Queries

You can select or identify objects to update or delete without using a GID by using a configurable matching integration saved query. Integration Saved Queries are defined in Power Data via the following menu: **Business Process Automation -> Power Data -> Integration -> Integration Saved Queries**. The queries are written as SQL statements that contain references to the information in the incoming XML transmission. For example, a query for a shipment GID given a shipment reference number would be as follows (e.g. if query GID is "GUEST.TEST_SAVED_QUERY_001"):

```
select s.shipment_GID from shipment_refnum s where s.shipment_refnum_qual_GID =
'{%QUAL%}' and s.shipment_refnum_value = '{%VALUE%}'
```

The inbound XML would then contain the following IntSavedQuery element to use the above query:

```
<IntSavedQuery>
   <IntSavedQueryGid>
          <Gid>
                 <DomainName>GUEST</DomainName>
                 <Xid>TEST_SAVED_QUERY_001</Xid>
          </Gid>
   </IntSavedQueryGid>
   <IntSavedQueryArg>
          <ArgName>QUAL</ArgName>
          <ArgValue>ZZ</ArgValue>
   </IntSavedQueryArg>
   <IntSavedQueryArg>
          <ArgName>VALUE</ArgName>
          <ArgValue>MY_SHIP_REFNUM_001</ArgValue>
   </IntSavedQueryArg>
<IsMultiMatch>Y</IsMultiMatch>
<NoDataFoundAction></NoDataFoundAction>
</IntSavedQuery>
```

During processing, Oracle Transportation Management would then replace the '{%QUAL%}' with 'ZZ' and the '{%VALUE%}' with ' MY_SHIP_REFNUM_001' to search for the shipment GID. And the transaction would be processed with using the shipment GID returned from the query. Note that the `IsMultiMatch` element can be used to indicate if multiple GIDs can be returned from the query and used for processing. Refer to the schema notes for additional fields that are available for the query.

The queries can also be written using XPath expressions to search for values from specific elements in the XML. An example of a query that relies on XPath expressions is as follows:

```
select ob.ORDER_BASE_GID from OB_REFNUM ob where ob.ORDER_REFNUM_QUAL_GID =
'PO' and ob.OB_REFNUM_VALUE =
'{TransOrder/TransOrderHeader/OrderRefnum[OrderRefnumQualifierGID/GID/Xid='PO'
and (not(OrderRefnumQualifierGID/GID/DomainName) or
OrderRefnumQualifierGID/GID/DomainName = '' or
OrderRefnumQualifierGID/GID/DomainName = 'PUBLIC')]/OrderRefnumValue}'
```

Note that the preferred method is to use the ArgName and ArgValue as they perform much better than the XPath expressions.

If `NoDataFoundAction` is not null and the Integration Saved Query returns no value, then the transaction code for the XML is switched to the transaction code specified by `NoDataFoundAction`, which must be a valid transaction code.

### *Default Integration Saved Queries for Updates*

There are a few interfaces that support default integration saved queries without having to specify the IntSavedQuery element in the inbound XML. This functionality is supported for Location, TransOrder, TransOrderLine, and Shipment.

The default integration saved queries are only used when the transaction code element has a value of U or D, and the primary GID for the interface is missing. If a TransOrderGid is missing, then the INT_TRANS_ORDER_GID_1 and INT_TRANS_ORDER_GID_2 saved queries are used. If a TransOrderLineGid is missing, then the INT_TRANS_ORDER_LINE_GID_1 and INT_TRANS_ORDER_LINE_GID_2 saved queries are used. If a Shipment GID is missing, then the INT_SHIPMENT_GID_1 and INT_SHIPMENT_GID_2 saved queries are used. If a Location GID is missing, then the INT_LOCATION_GID_1 and INT_LOCATION_GID_2 saved queries are used.

The defined queries must return a single GID of the element being referenced (for example, Order_Base_GID for TransOrder, Shipment_GID for Shipment, etc.). When a query returns multiple records, it will generate an error. Oracle Transportation Management supports up to two queries for each interface/record. If the first query generates an error or does not resolve to a single record, the second query will be applied. And if nothing is returned, then an error is generated.

## Character Encodings

The default character encoding for inbound XML transmissions is UTF-8. However, Oracle Transportation Management can receive transmissions in other encodings.

As per the Oracle Transportation Management Installation guidelines, the database character set must be UTF-8 which means that all the characters, irrespective of initial encoding, can be stored in the database.

To accept a different character encoding:-

- Specify it in the XML file. For example,

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  ```

- Save your XML file using that same encoding.
- If sending via HTTP post, you will also need to specify the encoding in the HTTP header. You must use the "Content-Type" attribute on the HTTP header to indicate that it’s a stream of character data with a given encoding.

Most modern text editors have the capability to save files in various encodings.

## Inbound Transmission Staging and Processing

A Transmission is ready to be processed when it is in the STAGED state, which means it has been parsed and all Transactions and Process groups have been determined.

However, for certain scenarios it is feasible that transactions that should be processed together do not all arrive in one Transmission. For example, depending on the message routing for various applications, a Tracking Event transaction could arrive from a WMS system a few seconds before the transaction from a Planning system which actually creates the shipment to which the tracking event should apply.

Oracle Transportation Management supports this scenario by using **Staging Transmissions**.

Staging Transmissions are identified by a `TransmissionType` of STAGING. This is an indication that processing can be delayed until some later time or event. The logic which should be applied to the Staging Transmission must be defined in the `StagingInfo` content. If no Staging Info content exists the Transmission is staged as normal.

### *Staging Info*

The `StagingInfo` content in the Transmission header defines the following:-

- A Staging Query - The `StagingQuery` element is an Integration Saved Query which should return one or more Transmission numbers. The first transmission number returned from the is used as the transmission number to which the transactions in the current are appended thus constructing a more complete Transmission.
- A Staging Process – The `StagingProcess` element is an Integration Saved Query which determines if the Transmission is now ready for processing. If the query returns a result, the Transmission will proceed with processing, and if no result is returned then the transmission will remain in a STAGING status until another event triggers it to be processed.

Note that initiating the processing of the Staging Transmission can also be enabled through the TRANSMISSION agent type, and the Staging Transmission Processing process management actions. Refer to the online help for details on the agents and process management actions.

# 7. Outbound Transmission Processing

All outbound transmission processing requires the configuration of an External System to which the messages will be sent. At a minimum, the External System must define the protocol to be used for the transport and any required external credentials.

The character encoding for all outbound transmissions is UTF-8.

There are various ways to trigger the sending of Transmissions to the external system from Oracle Transportation Management:

- Some messages are sent automatically as the result of workflow notification.
- Send and schedule integration transmissions in the Process Manager.
- Re-send and schedule integration transmissions in the Process Manager.
- Send transportation records to external systems from various managers using the Finder or Manager UI actions. The content of the information that is transmitted is determined by the context of the UI in Oracle Transportation Management from which you are sending it.

## Processing Overview

Once an event triggers the requirement to send a Transmission for one or more objects, the following processing takes place.

1. Construct internal objects based on SQL Queries. Out XML Profiles are applied during the generation of the objects.
2. Convert the objects into an XML String.
3. If anything other than the Current Target Namespace is selected then this conversion takes place now.
4. Apply XSL Transformations if needed.
5. Save the XML (Or Transformed output) to the I_TRANSMISSION table. If the outbound integration is designated as a message (e.g. generated via the Compose and Send UI action or dispatch actions), then an entry is stored in the I_MESSAGE table.
6. Send the XML via the indicated notify type (e.g. HTTP POST). The External System record in OTM indicates whether or not an acknowledgment is required from the external system.
7. Updates the status of the Transmission.

## External User Credentials

When sending messages outbound from Oracle Transportation Management it may be required to specify the username and password required by the receiving external server. This can be achieved by using the username related fields on the external system.

The following options are available:

- Use Credential: Oracle Transportation Management username associated with the process[2] sending the message will be present in Transmission and/or Message XML header. No password will be sent.
- Username and Password fields: Appear in Transmission and/or Message XML header.

**NOTE:** Refer to transport specific sections on outbound security for details on whether user credentials from the External System are used.

## Outbound XML Profiles

Out XML Profiles are used to exclude portions of outbound XML with a high degree of control. They reduce the size of the XML and minimize the number of queries that are used to generate the xml, thereby reducing the memory and time used and improving overall performance. There are several options for specify the elements to exclude. Please refer to the online help for additional details.

## Transform Outbound XML with XSL

Do the following to have Oracle Transportation Management transform your outgoing transmission from GLogXML to some other XML schema.

1. Define an XSL file that transforms between the schemas.
2. Upload the XSL file to create a Stylesheet Content record.
3. Create Stylesheet Profile to reference Stylesheet Content record created above.
4. Modify external system to reference Stylesheet Profile created above.

**Note:** Referencing Stylesheet files in the External System is deprecated and will be removed in a future version. Please convert to using Stylesheet Profile/Stylesheet Content as soon as possible.

## Support for Responses to Outbound Messages.

Functionality has been added to outbound integration processing to support receiving the TransmissionAck, TransactionAck, and TransmissionReport. You will be able to use these XML documents as a functional acknowledgement to receiving the Transmission and/or Transaction. Agent objects have been added for types "TRANSMISSION OUT" and "TRANSACTION OUT" to provide control on responding and handling of the response messaged.

## Integration Unit of Measure Preferences

You can specify the Unit of Measure (UOM) preference and precision to be displayed in outbound XML by defining preference for each UOM. These preferences can be defined with the Integration Preference UI that can be accessed via the following menu: **Business Process Automation > Power Data > Integration > Integration Preferences**. Integration preferences can be associated to an external system or an out XML profile. Refer to the online help for additional details.

Integration preferences can also be associated with the Rate Inquiry and Generic Query interfaces. To use the integration preference in those synchronous interfaces, you would specify the IntPreferenceGid in the RIQQuery, or the GenericQuery XML. When specified in the RIQQuery, the

[2] The workflow process determines the user and will either be the user signed in to Oracle Transportation Management whose action triggers the message or the automation agent configuration ("Run As").

RIQQueryResponse would apply the preferences before responding with the result XML. When specified in the GenericQuery XML, the preference would be applied to the resulting generated interface XML.

## Controlling Target Namespace

A new field on the External System Manager UI will specify the target namespace URL to be used for outbound messages. There are four possible values:

- **Current**: causes the latest version to be output
- **Versions prior to 6.4.2**: performs backward compatibility conversion from new to old namespace URL.
- **None**: causes the latest version to be output but with the namespace declarations removed.
- **None (Compatibility Mode)**: performs backward compatibility conversion from new to old namespace URL format and then also removes the namespace declarations.

To ensure that existing outbound integrations continue to function, External Systems will be migrated to have one of two values:

- **Versions prior to 6.4.2**: this will be used when the Include Namespace option is DEFAULT or STRICT.
- **None (Compatibility Mode)**: this will be used when the Include Namespace option is NONE.

All new External Systems will default to the Current target namespace.

Versions prior to 6.4.2 retained the compatibility logic for receiving older format GLogDate elements. This same capability is only retained if older namespace message formats are sent i.e. only the date formats specified in the 6.4.2 versions of the schemas will be accepted in version 6.4.2 messages.

See Appendix B – Backward Compatible Dates – pre 6.4.2 for more details.

# 8. Web Service Integration

## Web Service Security (WS-Security)

The Web Service Security Specification is an OASIS standard for defining security related information as part of a SOAP message. See http://www.oasis-open.org/.

Both Inbound and Outbound web services security requires the correct WSS policy to be used. Refer to the Oracle Transportation Management Security Guide for details on configuring alternative policies.

## Inbound

OTM and GTM support a number SOAP Web Services for sending messages inbound to the application. All available services are described in the following table.

| **Service Name** | **Purpose** |
|---|---|
| TransmissionService | Process Transmission documents according the Transmission Schemas. |
| IntXmlService | (**Deprecated**) Processes GLogXML transmissions. |
| IntGtmXmlService | (**Deprecated**) Processes GLogXML-GTM transmissions. |
| CommandService | Processes requests for DBXML export and import. |
| Workflow Web Services | Process agent actions against one or more existing transactional objects. See Workflow Web Service section for details. |
| MessageService | Process Mobile Device messages. See Mobile Device Communications section for details. |
| GtmRestrictedPartyService, GtmSanctionedTerritoryService | Process GTM screening messages. |

All the web services above are only deployed to the OTM/GTM Application Server, i.e. not the Web Server. If your system is configured as a cluster of more than one application server, each application server will provide a web service endpoint. At the current time, each application server has to be accessed directly and so any load balancing or failover across application servers would have to be managed external to the OTM/GTM system e.g. by a hardware load balancer.

### *Security*

All Inbound Web Services default to using a Web Service Security (WSS) policy which requires a Username Token to be passed and to be securely transported over HTTPS.

### *Accessing Web Services Description Language (WSDL)*

Each application server provides a different service endpoint URL for accessing the WSDL file. As a convenience, a new servlet has been added to Oracle Transportation Management that correctly points to the URL for the WSDL based on the application server. The servlet is accessed via the following menu path: **Business Process Automation > Integration > Integration Manager > Retrieve WSDLs** on main page. The servlet relies on the following properties to form the correct URL:

- appserver
- appserver.port.webservice.weblogic (default value is 7001)

If the application server configures a different port for use, please add the properties with the appropriate values in your glog.properties file.

The servlet also provides access to any related XSD schema files for each service.

### *Transmission Service*

The Transmission Service is the standard inbound Web Service for processing of Transmission XML documents.

> **Note:** The IntXmlService and IntGtmXmlService services are both now deprecated and will be removed in a future version.

TransmissionService is a SOAP Document Literal WRAPPED style of web service which means the SOAP Body of the SOAP message contains a Transmission XML wrapped in an element which matches the name of the operation to be performed. This style is recommended by the OASIS Web Services Interoperability Basic Profile. See https://www.oasis-open.org/ for details.

The service has two operations:-

- `publish`: This operation accepts and stages a Transmission for processing and returns a TransmissionAck as response. This operation maps to the `process` operation of the preceding IntXmlService.
- `execute`: This operation synchronously processes a Transmission and returns a reply Transmission with the processing results. For example, this operation should be used for `RemoteQuery` and `ServiceRequest` interfaces. In future, other interfaces may be supported inbound.

Refer to the TransmissionService WSDL for each operations input and output supported schema types.

## Outbound

It is also possible to configure an OTM External System to use an external web service endpoint as the destination for an outbound message.

The external system is configured to use a Web Service record created from the WSDL for the external service. See the online Web Service Manager help for details but, at a high level, the main steps are as follows:-

1. Create a Document in OTM from the external service WSDL URL.
2. Create a Web Service record in OTM using the Document created in step 1.
3. Configure External System to reference Web Service for the chosen Web Service operation and service endpoint.

   **Note**: A single service can offer multiple operations.

In step 1, the Document can be created from an uploaded text file containing the definitions (normally with the .wsdl file extension) or by specifying the URL for the WSDL resource. See the online Document Manager help for details.

**Note**: If the external service specifies a WS-SecurityPolicy (see below) then the Document must be created using the URL.

## Security

The following WS-Security policies are supported for outbound web services:-

- Username Token Profile – uses username and password configured on Web Service Endpoint.
- HTTPS Transport
- Message Encryption

***Note:*** Please refer to the Oracle Transportation Management Security Guide for configuration details for using Web Service Security for inbound and outbound messages.

## Service Call Pattern

OTM outbound Web Services support both Document Literal WRAPPED and Document Literal BARE styles. Document Literal WRAPPED is the preferred style as it offers the highest level of interoperability.

**Note:** Support for SOAP RPC style has been removed as it is rarely used due to its own interoperability issues.

By default, i.e. unless transformations have been specified, the SOAP Body will be the outbound `Transmission` document. Therefore, in versions prior to 6.4.2 the external web service 'input' parameter had to be defined as the Transmission which meant that Document Literal WRAPPED services could not be supported.

In version 6.4.2 it is now possible to select the Parameter Style which suites the WRAPPED style of service. See online Help for Web Service Manager UI on how to choose the correct style.

# 9. HTTP Integration

## Inbound

There are various options for sending integration messages into Oracle Transportation Management using HTTP:

- http://hostname/GC3/glog.integration.servlet.WMServlet
  WMServlet is the default servlet used when sending the Transmission or Message XML.
- http://hostname/GC3/glog.integration.servlet.TransformerServlet
  TransformerServlet is used to apply an XSL transformation to an XML to convert it into a valid Transmission XML. See Transform Inbound XML with XSL section for additional details.
- http://hostname/GC3/glog.integration.servlet.DirLoadServlet
  DirLoadServlet provides a faster option than WMServlet for pre-loading transactional data into Oracle Transportation Management in order to bypass application workflow. It can be used for inserting/creating data. Refer to Chapter 18: Transactional Data Pre-loading (DirLoad) for additional details.

  **Note:** The supported content types for HTTP messages (corresponding to the `Content-Type` HTTP header parameter) are `text/xml` and `application/xml`.

### *Security*

All Inbound HTTP messages require user authentication and authorization with the preferred method of passing credentials being Basic Authorization transported over HTTPS.

The methods of passing user credentials in the Transmission Header or as OTM specific HTTP Header parameters is deprecated as of version 6.4.2 and will be removed in a subsequent version.

### *Transform Inbound XML with XSL*

You can configure Oracle Transportation Management to transform your inbound transmissions from another XML schema to a valid Transmission XML. To do this:

1. Create a Stylesheet Content record in the OTM Database.
2. Include this processing instruction in the beginning of every transmission that Oracle Transportation Management needs to transform:

   ```
   <?gc3-int-translate stylesheet_name=”stylesheet_name”?>
   ```

   `stylesheet name` is the GID of the required Stylesheet Content record prefixed with “db:”. You can include multiple processing instructions in one transmission, in which case, Oracle Transportation Management will transform in the order the processing instructions appear in the transmission.
3. Post XML documents to http://hostname/GC3/glog.integration.servlet.TransformerServlet instead of http://hostname/GC3/glog.integration.servlet.WMServlet

   **Note:** The use of uploaded (file-system based) stylesheets instead of Stylesheet Content is still supported for backward compatibility but is deprecated and their use should be replaced with Stylesheet Content as soon as possible. Support for file-system based stylesheets will be removed in a future version.

## Outbound

### *Security*

The username and password credentials from the External System, if present, are used as the HTTP Basic Authorization authentication properties on the outbound HTTP connection. They will also be present in the Transmission Header.

HTTPS URLs are also supported.

# 10. Oracle Advanced Queue Integration

Oracle Advanced Queuing (OAQ) provides a Message Oriented Middleware (MOM) approach to sending and receiving XML messages to/from Oracle Transportation Management. The main benefit to using OAQ is the added level of guaranteed message delivery provided by a persistent message queue.

> **Note:** OAQ currently only supports the Transmission XML as the message content.

Both inbound and outbound queuing is supported. The message payload is the same in each case and is the Oracle User Defined Type (UDT) called INTG_QUEUE_MESSAGE.

The definition of the INTG_QUEUE_MESSAGE type is:

| ID | Type | Description |
|---|---|---|
| refnum | varchar2(101) | Can be assigned by client system for message referencing. |
| subject | varchar2(500) | Arbitrary text field definable by the client. |
| transmission_no | number | Oracle Transportation Management assigned transmission number. (only for internal use) |
| external_system_id | varchar2(101) | Overrides the external system GID in the TransmissionHeader. |
| user_name | varchar2(128) | Used for user authentication instead of specifying in the TransmissionHeader in the XML. |
| password | varchar2(128) | Used for user authentication instead of specifying in the TransmissionHeader in the XML. |
| xml | clob | Contains the XML message content (i.e. the Transmission). |

The following sections described the steps required to configure OAQ Integration.

## Inbound

### *Security*

A valid Oracle DB account is required to be able to enqueue messages to an OAQ queue in the OTM database. This account must have the required privileges based on the API used to queue the message e.g. PL/SQL, OCI etc. Please refer to the Oracle Database documentation for details.

In addition to the DB account, an OTM/GTM application user and password **must** be defined in order for the message to be processed. This will be used for further authentication and authorization at the OTM/GTM application level.

**Note:** In version 6.4.2 it is no longer supported to pass the username and password as part of the Transmission XML. The credentials must be passed in the corresponding fields in the INTG_QUEUE_MESSAGE.

### Step 1 – Create Queue Table(s)

The default implementation for OAQ in Oracle Transportation Management relies on a database table called INTG_QUEUE. The table is a point-to-point (single consumer) queue table. The table should be available with the installation of the Oracle Transportation Management database.

The OAQ functionality is not restricted to a single queue table. Additional queue tables can be created as needed, although a single queue table can also support multiple queues (see Step 2 below). The following procedure is available to create additional queue tables.

```
procedure create_int_queue_table(p_table_name varchar2,
   p_comment varchar2,
   p_table_space varchar2 default 'data',
   p_multiple_consumers  boolean default false );
```

The procedure supports creating multi-consumer queue tables using the `p_multiple_consumers` argument. The only requirement for creating a queue table is inbound queues that Oracle Transportation Management will read from must be created as a point-to-point (single consumer) table.

Example:

```
Sqlplus> execute
pkg_queue_management.create_int_queue_table('queue_test_table',  'This is for
test only');
```

Alternatively, for the multi-consumer:

```
Sqlplus> execute
pkg_queue_management.create_int_queue_table('queue_test_table',  'This is for
test only', 'data', true);
```

The queue tables created this way specify the INTG_QUEUE_MESSAGE type as the payload type.

### Step 2 – Setup Required Inbound Queues

For inbound processing of the XML, a set of four queues are required. The queues are:

- Inbound Queue (default name is 'inbound_aq')
- XML Topic Queue (default name is 'xml_stage_aq')
- Ack Queue (default name is 'ack_aq')
- Exception Queue (default name is 'exception_aq')

A Query Reply AQ (default name is 'query_reply_aq') is also needed for responding to Remote Query transactions such as Rate Inquiry (RIQ).

The following diagram shows the communication from the client (e.g. a PL/SQL client) to the database, as well as a high level depiction of the processing in the database.

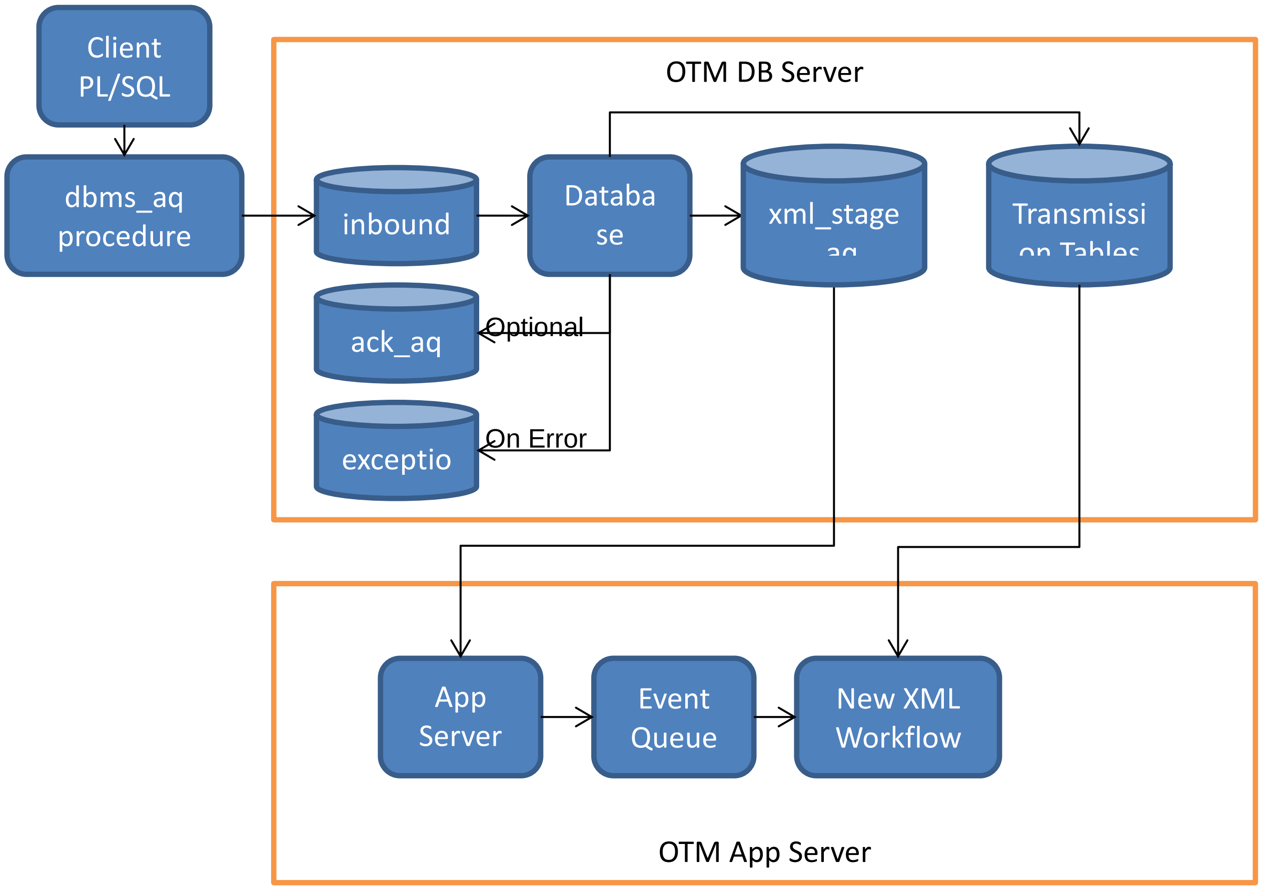


**Figure 10-1**

In the diagram above, you first send the XML to the **Inbound Queue**. The database listener reads from the queue, authenticates the OTM user credentials and stages the data to the i_transmission table. The database listener then queues the TransmissionAck message in the **Ack Queue** (unless this is suppressed in the Transmission Header). The listener also stages a message to the application server in the **XML Topic Queue** so that the application server can proceed with processing the message. If the Transmission XML is for a `RemoteQuery` transaction, the query response is placed in the **Query Reply** queue. If an error occurred in the staging, the database listener puts an exception message in the **Exception Queue**. The configuration of the application server listener is discussed later. All the queues may or may not be on the same queue table. However, the Inbound Queue and XML Topic Queue must not be multi-consumer queues. Furthermore, you are allowed to add multiple sets of inbound and outbound queues in the same queue table.

To create all queues required for inbound processing on the same queue table, use the following procedure:

```
procedure setup_inbound_queue_system(p_queue_table_name varchar2,
    p_inbound_queue_name varchar2,
    p_xmltopic_queue_name varchar2,
    p_ack_queue varchar2,
    p_exception_queue varchar2)
```

Example:

```
Sqlplus> execute pkg_queue_management.setup_inbound_queue_system
('queue_test_table',
 'another_inbound_aq',
 'raise_xml_topic',
 'acknowledgement',
 'notify_exception');
```

To create the queues on different queue table(s), use the following procedure:

```
   procedure start_queue(p_queue_name varchar2, p_queue_table_name varchar2)
```

Example - to create all the queues in example A above individually, use the following commands:

```
   Sqlplus> execute pkg_queue_management.start_queue ('another_inbound_aq',
   'queue_test_table');
   Sqlplus> execute pkg_queue_management.start_queue( 'raise_xml_topic',
   'queue_test_table');
   Sqlplus> execute pkg_queue_management.start_queue ( 'acknowledgement',
   'queue_test_table');
   Sqlplus> execute pkg_queue_management.start_queue ( 'notify_exception',
   'queue_test_table');
```

### Step 3 – Setup Database Listeners

For each inbound set of queues created above, a database listener should be created in order for the XML to be processed. The following procedure is used to setup the listener:

```
   procedure install_queue_listener(p_inbound_queue_name varchar2 ,
      p_xmltopic_queue_name varchar2 ,
      p_ack_queue varchar2 ,
      p_exception_queue varchar2)
```

Example:

```
   Sqlplus> execute pkg_queue_management.install_queue_listener  (
   'another_inbound_aq',  'raise_xml_topic', 'acknowledgement',
   'notify_exception');
```

In this case, you first send the XML to the 'another_inbound_aq' queue defined in the first parameter. The database listener reads from this queue and stages the data. If staging is successful, the database listener puts the TransmissionAck message in the 'acknowledgement' queue defined in the third parameter, and stages a message to the application server in the 'raise_xml_topic' queue defined in the second parameter. If an error occurred in the staging, the database listener puts an exception message in the'notify_exception' queue defined in the fourth parameter.

To stop the database listener, execute the following procedure on the "Inbound Queue":

```
   Sqlplus> execute
   pkg_queue_management.stop_queue_listener('another_inbound_aq');
```

### Step 4 – Setup Application Server Listeners

The application server requires a listener thread to be enabled to process the messages in the **XML Topic Queue**. The app server listener is set up through properties. The format of property entry is:

```
   glog.oaq.integration.{the_topic_queue_name}=1
```

For example, the property entry corresponding to the database listener created in Step 3 example should be:

```
glog.oaq.integration.raise_xml_topic=1
```

The value for the property must be a non-zero integer. The integer value determines the total number of threads for the listener. Since the application server listener is very lightweight, one thread should be enough to process the messages. If user desires to set up the value greater than one, a performance test should be done to determine the effects. To turn off the listener, set the value to "0" or remove the property entry. The property only takes effect during the startup.

### Auto Startup of Database Listener via Application Server

The app server has the ability to start and stop the database listener when it is being started or shut down. This is enabled through the use of the following property:

```
glog.integration.oaq.controlDbListener=true
```

When the property is true, the application server will also start the database listener when the app is starting, and will also shut down the database listener when the app server is shutting down.

**NOTE:** If multiple application servers are configured, this property should be set on only one.

## Outbound

You specify the queue to use for sending outbound XML from Oracle Transportation Management in the External System Manager in the UI. There are two approaches for creating the outbound queue, which is then used in the External System Manager. The first approach is to create the queue using the stored procedure; this enables you to specify the queue table to be used for the queue. After the queue is created, the external system can then reference the queue. The second approach is to specify the queue in the external system manager without first creating the queue. If the queue does not exist, the Oracle Transportation Management application would create the queue with the queue table defined in the property entry `glog.integration.oaq.outbound.queuetable`. By default, the queue table is **INTG_QUEUE**.

Example to create queue from procedure:

```
Sqlplus>  execute pkg_queue_management.start_queue ('outbound_example_queue',
'outbound_queue_table');
```

If the queue table is a multi-consumer queue table, the corresponding queue on the table is multi-consumer. At least one subscriber must be created for the queue; otherwise, Oracle Transportation Management will throw an exception during the enqueue process.

The following procedure will add a subscriber to the multi-consumer queue:

```
Sqlplus>   Pkg_queue_util.add_subscriber('mutlti_consumer_queue',
subscriber_name);
```

## Other Queue Management Utilities

To drop a queue:

```
execute pkg_queue_management.drop_queue ('your queue_name');
```

To delete all queue entries for a given queue:

```
execute pkg_queue_management.delete_queue_entries('your queue_name');
```

To remove every entry for all the queues in a given non multi-consumer queue table:

```
execute pkg_queue_management.empty_queue_table('queue table name');
```

To drop all queues in a given table:

```
execute pkg_queue_management.drop_all_queues('queue table name');
```

To drop a queue table as well as the corresponding queues:

```
execute pkg_queue_management.drop_queue_table('queue table name');
```

To stop all database listeners:

```
execute pkg_queue_management.stop_all_queue_listeners;
```

To stop a specific database listener:

```
execute pkg_queue_management.stop_queue_listener('inbound queue');
** The "inbound queue" is the first parameter in install_queue_listener
```

To remove database listeners:

```
execute pkg_queue_management.remove_all_queue_listeners;
```

To remove a specific database listener:

```
execute pkg_queue_management.remove_queue_listener('inbound queue');
```

## Optional Oracle Settings

The following Oracle DB parameters can be specified in init.ora or spfile. Refer to Oracle Database documentation for additional details on these parameters.

- aq_tm_process = 1 (to perform time monitoring on queue messages)
- job_queue_processes = 6 (to set the number of job queue processes started in an instance)

## Correlation of TransmissionAck to Transmission

Because of the asynchronous nature of message queues, the TransmissionAck that is placed in the "Ack Queue" or the "Exception Queue" may need to be correlated to determine the original Transmission that is referred to. There are several options available to provide this correlation:

- Use Sender Transmission Number: You must send a unique value in the TransmissionHeader.SenderTransmissionNo element in the Transmission XML before sending it to the queue. The value is echoed back in the TransmissionAck and the TransmissionReport.
- Use refnum field in INTG_QUEUE_MESSAGE: You must set the refnum field in the queue message before queuing. The value is echoed back as the SenderTransmissionNo in the TransmissionAck and the TransmissionReport.

- Limited Use of JMSCorrelationID Header Option: A column exists in the queue table for a correlation id (table column name is CORRID). This field can be populated on the inbound queue to Oracle Transportation Management, but is not populated on the queue table for the acknowledgement or the report. When this field is populated on the inbound queue, it is mapped to the Sender Transmission Number. The value is echoed back in the TransmissionAck and the TransmissionReport.

## Suppression of TransmissionAck

When using OAQ, clients can rely on the confirmation of receipt of the Transmission via the successful enqueing of the Transmission. With this approach, the TransmissionAck response is redundant. It is possible to suppress the TransmissionAck as a response by setting the SuppressTransmissionAck element to Y in the TransmissionHeader element. Errors that may occur with receiving the Transmission will still be reported in the error queue.

## TransmissionReport Sent Via QUEUE

After the application server completes processing of a Transmission, a TransmissionReport can optionally be sent to a recipient indicated the status of processing. It is also possible to place this TransmissionReport into a queue by setting the following in the TransmissionHeader:

```
AckSpecComMethodGID = 'QUEUE'
```

The AckSpec.ContactGID element should be specified. Oracle Transportation Management will look for Contact > External System > IntQueueName. The process also supports specifying a ContactGID for which the TransmissionReport should be sent. The ComMethodGID specifies the method of sending.

> **Note**: The correlation of the TransmissionReport to the original inbound Transmission is accomplished by specifying the SenderTransmissionNo in the inbound Transmission.

# 11. Direct Database Integration

## Inbound

Direct XML Insert is a way of sending inbound Transmissions and Mobile Communication Message XML to Oracle Transportation Management. This section will concentrate on Transmission XML documents. See Mobile Device Communications for details on sending Message XML.

The documents are inserted directly into the DB via a new PL/SQL procedure – `insert_transmission` – in the `pkg_integration_util` package. There is no corresponding outbound procedure.

The procedure signatures are described below:

```
insert_transmission (  p_username IN VARCHAR2,
                       p_password IN VARCHAR2,
                       p_transmission IN CLOB,
                       p_transmission_no OUT NUMBER,
                       p_data_queue IN VARCHAR2 DEFAULT NULL,
                       p_cluster_gid IN VARCHAR2 DEFAULT NULL,
                       p_priority IN NUMBER DEFAULT 0)
```

| Parameter | Description |
|---|---|
| `p_username` | **IN.** OTM user account under who's authority XML will be processed. |
| `p_password` | **IN.** Password for OTM user. |
| `p_transmission` | **IN.** Transmission XML message. |
| `p_transmission_no` | **OUT.** Unique Transmission number generated for Transmission XML message. |
| `p_data_queue` | **IN. Default NULL.** Data Queue Definition GID that event will be handled by. |
| `p_cluster_gid` | **IN. Default NULL.** Cluster that will process the events.<br><br>If this parameter is specified on insert then the property `glog.dataqueue.eventCluster` must be set equal to the Cluster GID on all Application Servers in the cluster which should process the events. |
| `p_priority` | **IN. Default 0.** Priority of event. Used by Poller to order sequence of events for Preemptive Poller. |

### *Security*

The Oracle Database user DIR_XML_USER has all required privileges and synonyms to successfully insert transmission into the database. Use DIR_XML_USER to execute this procedure.

## Internal Processing

The Transmission XMLs inserted using this procedure are processed using an internal Integration Data Queue – INTEGRATION IN DIRECT XML and can be monitored in exactly the same way as Integration Data Queue events covered in Chapter 14: Integration Data Queues.

Figure 11-1 shows the processing for directly inserted XML messages.

1. XML message passed to `insert_transmission` procedure.
2. OTM procedure stores XML in Transmission table, data queue event in Direct Insert Data Queue and returns Transmission number to caller.
3. The Poller retrieves a number of events and calls the Executor for each event.
4. The Executor retrieves the message XML and calls the New XML Insert Workflow to perform the following Transmission pre-processing:
   - Populate Transaction table data
   - Populate Process Group data if present.
   - Populate Transmission reference numbers if present.
   - Publish New XML topic or queue event to INTEGRATION IN data queue if configured.

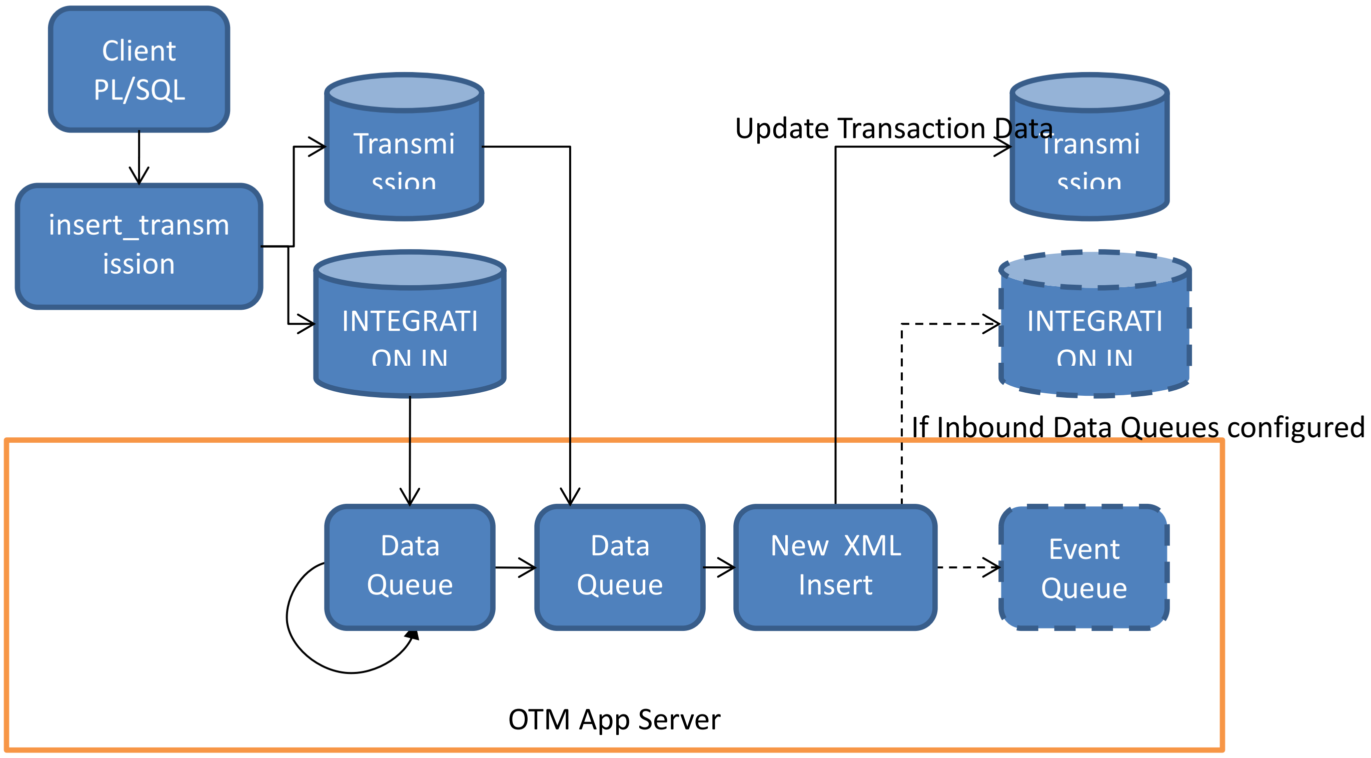


**Figure 11-1**

## Data Queue Definition Customization

The INTEGRATION IN DIRECT XML Data Queue can be customized exactly as described in the section Chapter 14: Integration Data Queues, Understanding the Integration Data Queue Definitions, with the following additions.

1. Direct XML supports multiple Data Queue Definitions

   a. Whereas the other Inbound and Outbound Data Queue Definitions to be used are determined by logic in the application server, the Direct Insert XML Data Queue GID can be selected by the client and specified as a parameter in the procedure call to insert the XML message.
   b. Multiple Data Queue Definitions allow the ability to partition a large volume of XML messages into separately controlled groups as each Definition can specify distinct values for number of threads, batch size and polling interval.

**Note:** Multiple Data Queue Definitions would also require multiple Poller Definitions because the queue name (i.e. the GID) forms part of the Poller Filter and so would need to be customized.

2. Multiple Executor options used to process events
   a. By default the executor used to process the initial event on the INTEGRATION IN DIRECT XML data queue is the INTEGRATION IN DIRECT XML executor. If the INTEGRATION IN data queue is also active, this will mean that a Transmission sent in using Direct Insert XML will pass through two data queues. An alternative executor is available - INTEGRATION IN COMPOSITE – which will instead by pass the INTEGRATION IN data queue (i.e. as if it were **not** active) and proceed directly to the New XML Workflow. This is useful in scenarios where multiple transport methods are used e.g. Web Service and Direct Insert XML but it is only desired to use one data queue for each workload.
   b. If using a prioritized poller, the use of the INTEGRATION IN COMPOSITE executor also means that the assigned priority causes the New XML workflow to be triggered in priority order. However, the priority is not used beyond this point i.e. the internal logic of the New XML Workflow no longer uses the priority after it is first triggered.
3. Method used to call New XML Insert Workflow is configurable
   a. By default, the New XML Insert Workflow is called by the INTEGRATION IN DIRECT XML Executor using the 'execute' paradigm, which means the workflow is executed using the Poller thread which retrieved the event. This method can be changed to either 'publish' or 'publishWait' by setting the property `glog.integration.dataqueue.inboundDXIType` which if not set defaults to 'execute'.
      i. **Execute**: executed using the Poller thread which retrieved the event.
      ii. **Publish**: topic is published to in-memory event queue and process at some time later by another thread. Poller thread is free to process another event without waiting.
      iii. **publishWait**: topic is published to in-memory event queue and processed at some time later by another thread. The Poller thread waits for the topic processing to complete before attempting to process another event.

# 12. FTP Integration

## Outbound

Transmission XML documents can be saved to an external FTP server.

The server name, port and remote directory should be configured on the External System record.

### *Security*

The username and password credentials from the External System, if present, are used as the FTP authentication properties.

> **Note:** the Use Credential check box is not valid for FTP.

The transport for FTP can optionally be secured using the ’Use Secure FTP’ check box on the External System. This enables the support for FTPS (FTP over TLS).

> **Note:** S/FTP often just referred to as ‘Secure FTP’ is not the same as FTP over TLS and is not supported.

# 13. Setting Up Interfaces

The following general information helps you set up your interfaces. If an interface has specific setup requirements, they are found with the pages defining each interface.

To set up interfaces, you must define where to send transmissions and what to do with the transmissions Oracle Transportation Management receives. Information throughout Oracle Transportation Management acts interdependently; one piece of information depends on another to perform an action. For some interfaces to work, data from other sources must already be present in Oracle Transportation Management. For example, before you can create a shipment, you must create itineraries.

## Define External Systems

To send transmissions to other systems, you must define the systems in Oracle Transportation Management using the External System Manager.

## User Management

You must add service providers as users and enter user associations for them. To perform user management functions, log in to the SERVPROV with a username that contains administrator (ADMIN) rights.

- Define service providers as users in Oracle Transportation Management.
- Define associations for the service providers.

## Workflow Parameters

In Power Data, define workflow parameters that determine how Oracle Transportation Management responds to inbound and outbound transmissions. You define Workflow Power Data topics to define the way the tendering shipments works.

- **Workflow Parameters**: Use the Workflow Parameters to define how Oracle Transportation Management tenders shipments. You also define shipment notification messages. For example, you define information, warning, and, fatal messages that Oracle Transportation Management sends out as the results of status information sent by service providers about particular shipments.
- **Workflow Trigger Parameters**: Use the Workflow Trigger Parameters to define how often Oracle Transportation Management performs tender activity. This topic helps you control system performance. For example, if Oracle Transportation Management is performing tender actions too frequently, your system performance may be slowed.

## Agent Manager

The Automation Agent Manager lets you construct workflow agents that are key components to automate Oracle Transportation Management. A workflow agent listens for an Oracle Transportation Management event, verifies a user-defined condition, and executes one or more actions that you choose from an action library.

# 14. Integration Data Queues

## Overview

By default the Oracle Transportation Management application server workload is managed by a set of in-memory event queues; with a configurable number of threads available to process each queue. The Oracle Transportation Management Data Queue infrastructure was introduced in version 5.5 to permit a more configurable fine-grained control of various aspects of this internal workload. The Data Queues are database resident queues where a **configurable** number of "poller" threads retrieve a **configurable** number of events at a **configurable** interval and pass them to an "executor" to be processed. See Figure 14-1 below.

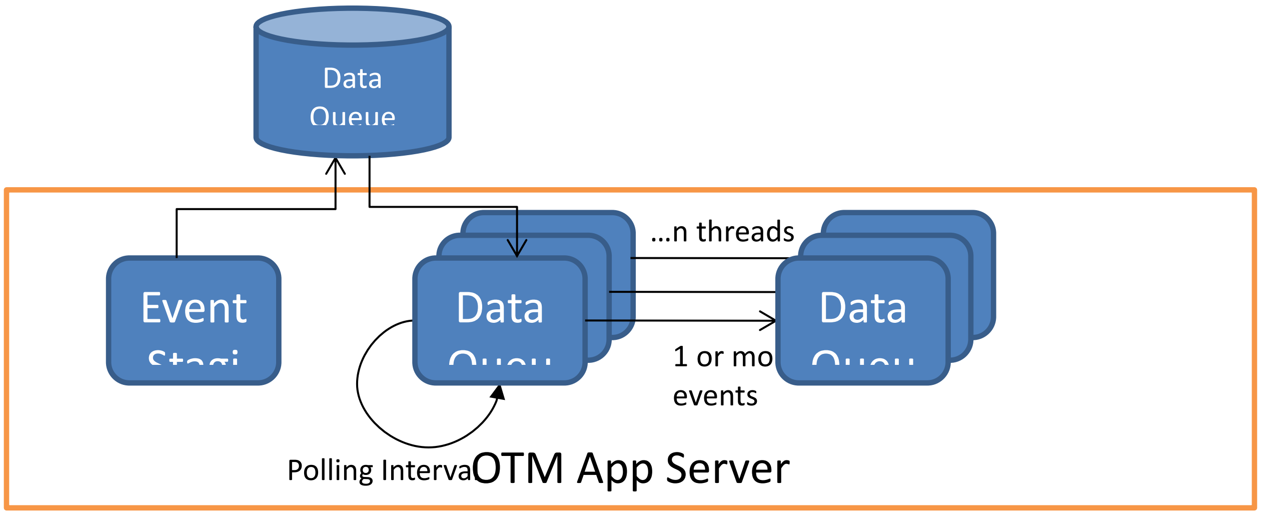


**Figure 14-1**

**Note:** Although the entity names used are similar, Oracle Transportation Management Data Queues do not use Oracle Advanced Queue objects for processing data queue events.

The internal processes for Inbound and Outbound Integration can now be configured to use this infrastructure.

**Note:** See *Data Queue Manager* in the OTM online Help for coverage on the User Interface used to configure each data queue.

Figure 14-2 shows the application server processing of an inbound Shipment Status XML without the use of data queues. The XML message will arrive via one of the supported protocols: HTTP, SOAP Web Service or OAQ.

**Note:** the Direct Insert XML protocol is covered Chapter 11, Direct Database Integration.

1. The XML is stored in the Transmission table with a new unique Transmission number.
2. A New XML topic containing the Transmission number is placed on the in-memory event queue.
3. A listener thread which has subscribed to New XML topics removes the event and passes it to the New XML Workflow to process.
4. Workflow processes the content of the XML, which in this case results in a new status being assigned to a shipment.

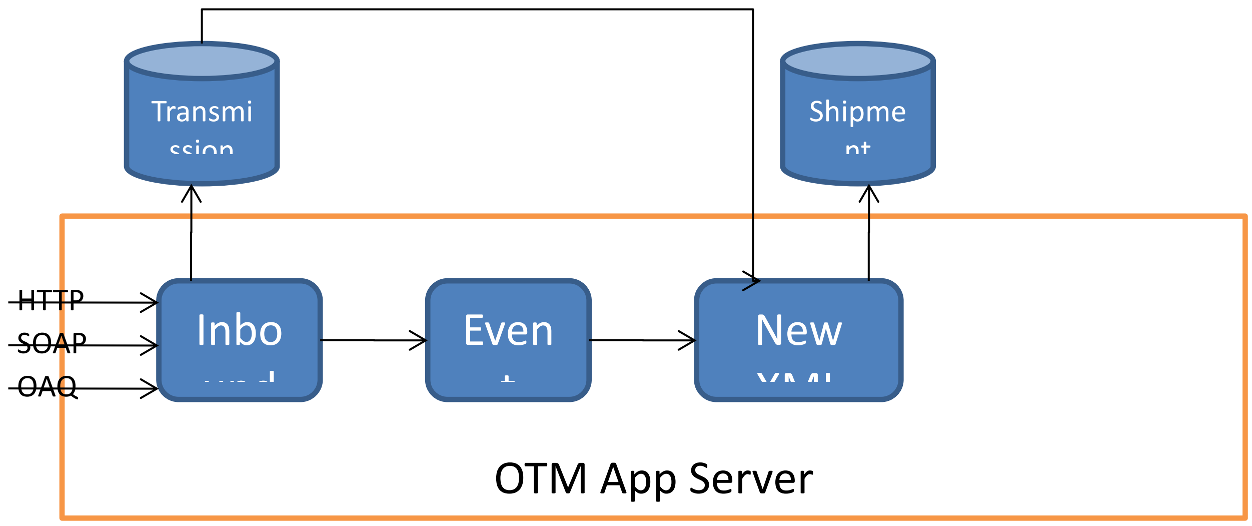


**Figure 14-2**

Imagine an Oracle Transportation Management implementation scenario that has an inbound interface with a carrier status application where shipment statuses are sent in one batch of 10000 to Oracle Transportation Management. Without Data Queues, the application server in-memory event queues would contain all 10000 Shipment Status transactions in one backlog, therefore competing with all other internal application server processes for resources and process time.

Figure 14-3 shows the application server processing where data queues were configured for inbound XML messages. The Inbound Data Queue process can instead be configured to stage the initial 10000 transactions and process in batches of 1000 every 5 minutes.

1. The XML is stored in the Transmission table and the associated Transmission number is stored in a Data Queue Event.
2. At some time within the configured interval, a Poller thread will retrieve up to 1000 data queue events and pass individually to the Executor to be processed.
3. The Executor publishes a New XML topic which, from then on, will be processed as with normal in-memory event queues.

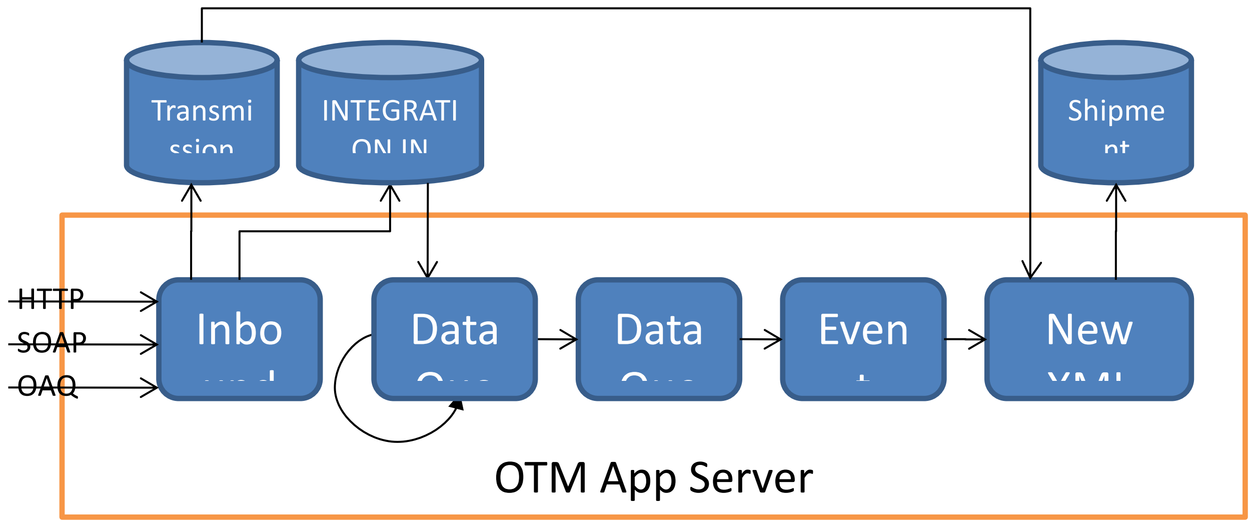


**Figure 14-3**

The application server processing for Outbound XML messages also uses in-memory event queues to manage workload; one for the building of the XML and another for the physical transport of the message. The use of Data Queues for outbound message works essentially identical to the inbound scenario described above i.e. instead of publishing new build or transport topics on in-memory event queues, data queue events are stored in outbound data queues. The same arrangement of Poller and Executor then publishes the topic to the next step in the process.

## Activating Integration Data Queues

There are several Integration Data Queues available in the PUBLIC domain:-

- **INTEGRATION IN**: Used to manage all Inbound GLogXML Transmissions sent in via UI (Upload Transmission), HTTP (WMServlet), SERVICE (IntXmlService), and OAQ (INBOUND_AQ).
- **INTEGRATION IN DIRECT XML:** (covered in Chapter 11)
- **INTEGRATION OUT XML BUILD**: Used to manage outbound process of generating Transmission content from database objects and staging in Transmission tables ready for transport.
- **INTEGRATION OUT TRANSPORT – HTTP**: Used to manage transmissions ready to be transported via HTTP to external system.
- **INTEGRATION OUT TRANSPORT – SERVICE**: Used to manage transmissions ready to be transported via web service call to external system.
- **INTEGRATION OUT TRANSPORT – FTP**: Used to manage transmissions ready to be transported via FTP to external system.
- **INTEGRATION OUT TRANSPORT – QUEUE**: Used to manage transmissions ready to be transported via Oracle Advance Queue to external system.

The inbound data queue events reside in the Q_INTEGRATION_IN Data Queue table. The outbound data queue events reside in the Q_INTEGRATION_OUT Data Queue table.

By default the PUBLIC queues are inactive. To activate the queues, login as DBA.ADMIN and edit the Data Queue records via the Data Queue Manager (under **Business Process Automation > Integration** menu).

> **Note**: It is not recommended that you activate or customize any of the public data queues. If you need to activate or customize a data queue, copy the queue and edit the copy. PUBLIC queues should be used as a template to create your own queues. PUBLIC data is not static and may change between releases. Custom changes should be made in domain-specific queues, not public ones. If you customize one of the integration queues, you must configure the appropriate `glog.integration.dataqueue` property so that it references the name of the customized data queue.

Alternatively, the PUBLIC records can be copied to DOMAIN specific records and activated there. In order for the application server to recognize the new queue names, the following properties must also be set:

```
glog.integration.dataqueue.inbound
glog.integration.dataqueue.xmlBuild
glog.integration.dataqueue.transport.http
glog.integration.dataqueue.transport.service
glog.integration.dataqueue.transport.ftp
glog.integration.dataqueue.transport.queue
```

For example,

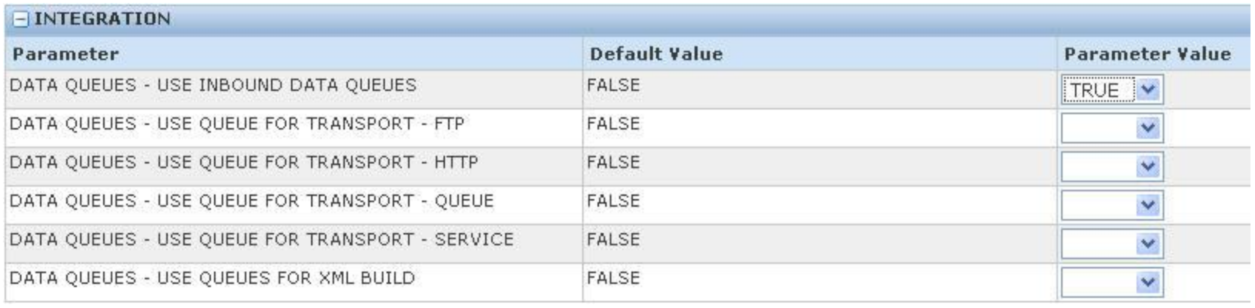
```
glog.integration.dataqueue.inbound=MYDOMAIN.MYINBOUNDQ
```

**Note:** An application server can only process one data queue of each type. For example it is not possible to configure two inbound data queues: MYDOMAIN.INBOUNDQ1 & MYDOMAIN.INBOUNDQ2 or MYDOMAIN2.INBOUNDQ1. The only exception to this is the data queue configuration for Direct Insert XML. See Chapter 11 for details.

Once one or more Integration Data Queues are active, the processing can be controlled on a domain-by-domain basis by setting one of the following PARAMETER SET parameters to 'TRUE' on the DOMAIN PARAMETER SET.

- DATA QUEUES - USE INBOUND DATA QUEUE
- DATA QUEUES – USE QUEUE FOR XML BUILD
- DATA QUEUES – USE QUEUE FOR TRANSPORT – HTTP
- DATA QUEUES – USE QUEUE FOR TRANSPORT – SERVICE
- DATA QUEUES – USE QUEUE FOR TRANSPORT – FTP
- DATA QUEUES – USE QUEUE FOR TRANSPORT – QUEUE

For example,



INTEGRATION

| Parameter | Default Value | Parameter Value |
|---|---|---|
| DATA QUEUES - USE INBOUND DATA QUEUES | FALSE | TRUE |
| DATA QUEUES - USE QUEUE FOR TRANSPORT - FTP | FALSE | |
| DATA QUEUES - USE QUEUE FOR TRANSPORT - HTTP | FALSE | |
| DATA QUEUES - USE QUEUE FOR TRANSPORT - QUEUE | FALSE | |
| DATA QUEUES - USE QUEUE FOR TRANSPORT - SERVICE | FALSE | |
| DATA QUEUES - USE QUEUES FOR XML BUILD | FALSE | |

If either the queue is not active or the domain parameter is FALSE, the default in-memory queues will be used.

**Note**: It is possible to override this default to process via in-memory queues. Set property `glog.integration.dataqueue.inbound.useMemoryQueue=false`. This will cause the inbound Transmissions to remain in 'STAGED' status.

## Understanding the Integration Data Queue Definitions

The standard installation of OTM version 6.2 has PUBLIC Data Queue Definitions configured for each of the Integration Data Queues described in the preceding section.

This section will describe these definitions so that any customization can be done without any unforeseen impact on the integrity or performance of the OTM servers.

The Data Queue Definition is a unique (based on a Data Queue Definition GID) configuration of the following properties.

| **Parameter** | **Description** |
|---|---|
| Active | Only queues which are marked as 'Active' will have Poller threads started on the application server. |

| Parameter | Description |
|---|---|
| Thread Count | The number of Poller threads that will be started on each application server. All Poller threads will synchronize their access to the events on a data queue. One thread at a time will retrieve a batch of events, mark them all as 'ACTIVE' and commit the transaction before releasing the queue for another thread to access.<br><br>**This would normally be tuned to match the expected volume of events.** |
| Batch Size | The maximum number of events that will be retrieved by each Poller thread.<br><br>**This would normally be tuned to match the expected volume of events.** |
| Polling Frequency | The duration each Poller thread will sleep between attempts to retrieve a batch of events. **Note:** if the time taken to process a batch exceeds the polling frequency the Poller thread will essentially start processing the next batch more or less immediately after completing the current batch.<br><br>**This would normally be tuned to match the expected volume of events.** |
| Data Queue Table | Corresponds to the physical table which will contain the events<br><br>**This should not require customization.** |
| Data Queue Poller | The Poller configuration specifies the behavior and attributes of the SQL statement used to retrieve the batch of events. In addition to the Class that will execute the query, the following can be customized:-<br><br>• Filter – these are additional predicates added to the basic statement (e.g. status = 'QUEUED' etc.)<br><br>• Order – this is specified as the ORDER BY criteria. This is how priority ordering is achieved.<br><br>Poller Definitions are configured via Business Process Automation Power Data.<br><br>**Custom Poller definitions may be required if additional criteria is used to further partition events placed on the same queue e.g. by joining other tables.** |

| Parameter | Description |
|---|---|
| Data Queue Executor | The Executor configuration species the Class that will process the actual event and whether events in the ACTIVE or FAILED state should be reprocessed on app server startup.<br><br>Executor Definitions are configured via Business Process Automation Power Data.<br><br>**Custom Executor definitions may be required if reprocessing of events needs to be handled differently.** |
| Finder Set | This specifies the Class which formats the queue specific information in the Data Queue Manager Events UI.<br><br>**This should not require customization.** |
| Related Queue | This is used to manage the parent/child relationship between events.<br><br>**This is not used by Integration Data Queues.** |

### *Customizing Inbound Data Queue Definitions*

There are two inbound data queue definitions created by default in the PUBLIC domain: INTEGRATION IN and INTEGRATION IN DIRECT XML. Events for both are held in the Q_INTEGRATION _IN Data Queue Table. By default both queues are inactive.

If these base configurations are deemed to be sufficient for the expected volumes, they can be activated by logging in as DBA.ADMIN and setting each queue definition to 'Active'. Additionally, as mentioned previously, the INTEGRATION IN Parameter set value also needs to be set to 'TRUE'.

#### Customized Poller Definition

The Poller Definitions for both queues are configured to use the PREEMPTIVE POLLER plug-in, which uses a 'Top N query' format to retrieve an ordered batch of events. The format of the SQL statement that would be used to retrieve a batch of events would essentially be as follows:-

```
SELECT rownum, {column list}

   FROM (SELECT {column list}

            FROM Q_INTEGRATION_IN

            WHERE q_state='QUEUED' and {filter}

            ORDER BY {order})

   WHERE rownum <= {batch size}
```

The arguments in braces are replaced at runtime as follows:-

| Parameter | Description |
|---|---|
| column list | The complete column list is controlled via the Event Class which for Q_INTEGRATION_IN events would be:-<br><br>`q_event_seq, rownum, q_start, q_process_start, domain_name, q_preemption_priority, topic_class_name, topic_parameters, topic_description, i_transmission_no, q_queue_name.` |
| filter | The PUBLIC Poller definitions for INTEGRATION IN and INTEGRATION IN DIRECT XML has a filter clause as follows:-<br><br>q_start < sys_extract_utc(current_timestamp) and q_queue_name = 'INTEGRATION IN'<br><br>or<br><br>q_start < sys_extract_utc(current_timestamp) and q_queue_name = 'INTEGRATION IN DIRECT XML'<br><br>Therefore, if a custom Data Queue Definition is to be used there will need to be a corresponding custom Poller Definition in order to override the `q_queue_name` parameter. |
| order | For Preemptive Data Queues the base ORDER BY clause will be as follows:-<br><br>q_preemption_priority, q_event_seq<br><br>That is, ordered by priority first and then event sequence number. |
| batch size | This is the batch size taken from the Data Queue Definition. |

**Customized Executor Definition**

In addition to the ability to configure redo of ACTIVE or FAILED events, it is possible to configure how the topics created by processing Direct Insert XML events are processed. See Chapter 11 for details.

### *Customizing Outbound Data Queue Definitions*

The Outbound data queue definitions are subjected to the same customization capabilities and constraints as the inbound data queues.

## Scalability & Clustering Considerations

In a Scalability environment there are additional considerations required to ensure that any required Data Queues are active and processing events as and when required.

Poller threads will only be started on Application Servers which belong to the Cluster associated to a particular data queue. Data Queues are associated to a Cluster via the Cluster Manager under Configuration and Administration, Cluster Management.

For example, assume CLUSTER-01 is defined and has app servers APP-01 and APP-02 assigned to it via the Cluster Manager. App server APP-03 is assigned to CLUSTER-02.

If data queue definition INTEGRATION IN is also assigned via the Cluster Manager to CLUSTER-01, then Poller threads will started when the APP-01 and APP-02 app server machines are started. Poller threads will not be started on APP-03 and therefore it will not process any Data Queue events for INTEGRATION IN. See "Cluster Management" in online Help for a full description.

## Monitoring Integration Data Queues

In addition to its primary task of administering the data queues themselves, the Data Queue Manager is also used to monitor the activity of events on each queue via the Events action button. This provides access to a Finder/Results screen from where all events for a queue can be listed and individual event details can be viewed. The majority of information is generic to all Data Queues e.g. Process Time, and Log Process ID. The following sections describe the Integration Data Queue specific data.

### *Inbound Events*

All inbound events have an associated transmission number. In normal processing, when a transmission is sent to Oracle Transportation Management, the XML is persisted in the transmission tables with an initial status of STAGED and a New XML event is published almost immediately. Once the New XML event execution has started the status is changed to FRESH. When the Inbound Data Queue is active, the XML is persisted in the transmission tables with a status of STAGED and the New XML event is persisted in the data queue table. Consequently, transmissions can remain in this STAGED status for much longer.

Inbound events can also be preempted. In other words, events already present in the data queue can be selected for processing ahead of other events that may have been inserted before it. By selecting an event for preemption, the Q_PREEMPTION_PRIORITY is set to the lowest value in the data queue table. This will ensure the next POLLER will retrieve the selected event.

### *Outbound Events*

Outbound events may go through two event queues: XML Build and Transport. The XML Build process happens before a transmission record has been created and so these events do not have an available transmission number. They do, however, have a Notification Context that shows the object type of the communication, such as Location, and one or more object GIDs.

The display of the context is limited to approximately 4000 characters and so if the context is greater than this it will not be displayed.

As with inbound events, outbound events can be preempted.

However, unlike inbound events, outbound event priority can be set when the data queue event is staged in the data queue table. The priority value is managed via the External System where it can be set alongside an OUT XML PROFILE for a particular GLogXML Element e.g. PlannedShipment.

# 15. Customs Info Integration

The Global Trade Management component of Oracle Transportation Management supports an interface to Customs Info for retrieving reference data used in the services. This section details the setup for accessing the data.

## Customs Info Registration

Registration with Customs Info is required for access to the Customs Info data for Global Trade Management. The registration process will give a site ID and password to the client that can be entered in the setup of Global Trade Management. This user and password combination will be used to download the actual data.

## Setup in Global Trade Management

### *Global Trade Management Properties Setup*

#### Network Connectivity to Customs Info Server

If the network Global Trade Management is installed requires a proxy server for external access to the Customs Info server, specify the following properties in the glog.properties:

```
glog.integration.http.proxyHost=proxy-address
glog.integration.http.proxyPort=proxy-port
```

Refer to the online help for additional details.

#### Properties for Data Loading Directories

The data loader relies on pre-defined directories for the source and output directories. The directories are defined via properties:

```
gtm.dataload.basedir=$temp.dir$/dataload
gtm.dataload.inputdir=$gtm.dataload.basedir$/input
gtm.dataload.outputdir=$gtm.dataload.basedir$/output
gtm.dataload.workingdir=$gtm.dataload.basedir$/output
```

These properties do not need to be changed for implementation. The logic will create the directories needed if they are not already created.

### *Setup of Customs Info User Credentials*

After retrieval of the user credentials from Customs Info, configure the credentials in Global Trade Management as follows:

- Setup an External System record in Global Trade Management with the following details:
    - **User Name**: User ID provided during registration
    - **Password**: Password provided during registration
    - **Password (Confirm)**: Same value for password
    - **URL (in For HTTP/HTTPS section)**: URL for accessing the Customs Info status document. The URL may resemble the following: https://gtm.content.descartes.com/status/statusdocument.xml
- Update the Content Source to refer to the newly created External System record as follows:

- Access menu: Trade Master Data > Power Data > Data Loading > Content Source
- Edit the Content Source with ID: "CUSTOMS INFO" which has been predefined for Customs Info
- Add an entry in the Content Source Config section at the bottom as follows:
    - **ID**: Specify an Id (e.g. CI_GTM_DATA)
    - **Comm Method**: Specify "HTTPPOST"
    - **External System**: Select the external system created above
    - **Active**: Select to enable the config.

## Processing Global Trade Management Content

The actions to initiate the Global Trade Management content loading are defined with process control. This is accessible from the following menu:

**Trade Master Data -> Process Management**

The following processes are available:

- **Download Data Content**: Used to retrieve the content from a Global Trade Management content source. For Customs Info, this process will download the latest files from the Customs Info server into the local Global Trade Management server.
- **Purge Data Content**: This is used to cleanup up the files on the Global Trade Management server used for the data loading.

# 16. Global Trade Management Screening Service

This section captures the screening services developed in Global Trade Management.

All interactions with the Global Trade Management Services through integration is made using the Service Request element (refer to the GLogXML-GTM.xsd schema for additional details). Upon receiving the ServiceRequest XML in Global Trade Management, the request is processed synchronously and a ServiceResponse xml is generated as a response to the caller. Note that the Transmission/TransmissionHeader/TransmissionType must be set to “SERVICE” for the synchronous processing to occur.

The ServiceRequest element contains several options for the screening services. A few of the available services are as follows:

| **ElementName (Service)** | **Description** |
|---|---|
| RestrictedParty | Used to indicate if the specified party is on the restricted parties list. |
| SanctionedTerritory | Used for transactional screening of territories. The screening uses the rules engine and runs all rules that have a control category of SANCTION. Users specify any number of territories and qualifiers in the LocationInfo element. |
| Classification | Used for transactional searching for global classification based on product description. |
| ComplianceRule | Used for transactional screening of personal information, product and usage information, location information, and user defined conditions. |
| LicenseDetermination | Used to find matching licenses for a license control, based on party, location and user defined code data. |

Refer to the GLogXML-GTM.xsd schema file for additional fields in the ServiceRequest and ServiceResponse, and the online help for details on the services.

# 17. Mobile Device Communications

## Message Schema

The Mobile Message infrastructure is used for mobile communication messages sent inbound and outbound to Oracle Transportation Management. These messages are intended for communication to a mobile communication device or sensor. When sent inbound to Oracle Transportation Management, these messages should be small. The message body can contain text-formatted content, which may be parsed and converted to the Transmission XML when transmission processing is required.

### *Schema Files*

The following schema files contain all definitions for the Message schema:

| **Schema Name** | **Description** |
|---|---|
| Message | Definitions for all Message XML Primary Documents. |
| Message Content Source | Definitions for Message Group implementations for the Content Source infrastructure. |

### *XML Namespaces*

| **XSD Schema File** | **Namespace** |
|---|---|
| Message.xsd | http://xmlns.oracle.com/apps/otm/msg/v1 |
| MessageContentSource.xsd | http://xmlns.oracle.com/apps/otm/cs/msg/v1 |

### *Primary XML Documents*

There are three primary XML documents in the Message XML schema that are used inbound and outbound to/from Oracle Transportation Management:

- Message
    - The Message is the primary document used for inbound to and outbound for mobile communication messages. Each message can contain either text based or XML-based content in the message body.
- MessageAck
    - The MessageAck is the response message to the receipt of the message. It contains the confirmation for the receipt of the message with an assigned ReferenceMessageId element, or an error if the message was not correctly received.
- MessageReport

- The MessageReport summarizes the errors that were detected during the processing of the message if the Message Body contains 'actionable' content. For example the content could describe a Tracking Event which is translated into a Transmission XML and processed as an Integration message. The report is optionally sent after all the transactions in the Transmission have been completed (successfully processed or generated errors). The request for the MessageReport is indicated in the message in the AckSpec element in the MessageHeader. The MessageReportBody in the MessageReport will include the TransmissionReport that will summarize the errors that were detected during the processing of the Transmission.

Each of these documents is detailed in the Message XML schema file.

In the case where the message is correctly parsed and converted to a Transmission XML for inbound transmission processing, the Transmission Report XML may also be sent as a summary of processing if the AckSpec is specified in the generated Transmission resulting from the parsing.

## *Application Level Protocol*

The following diagram shows when the documents are sent for an inbound Message XML scenario into Oracle Transportation Management.

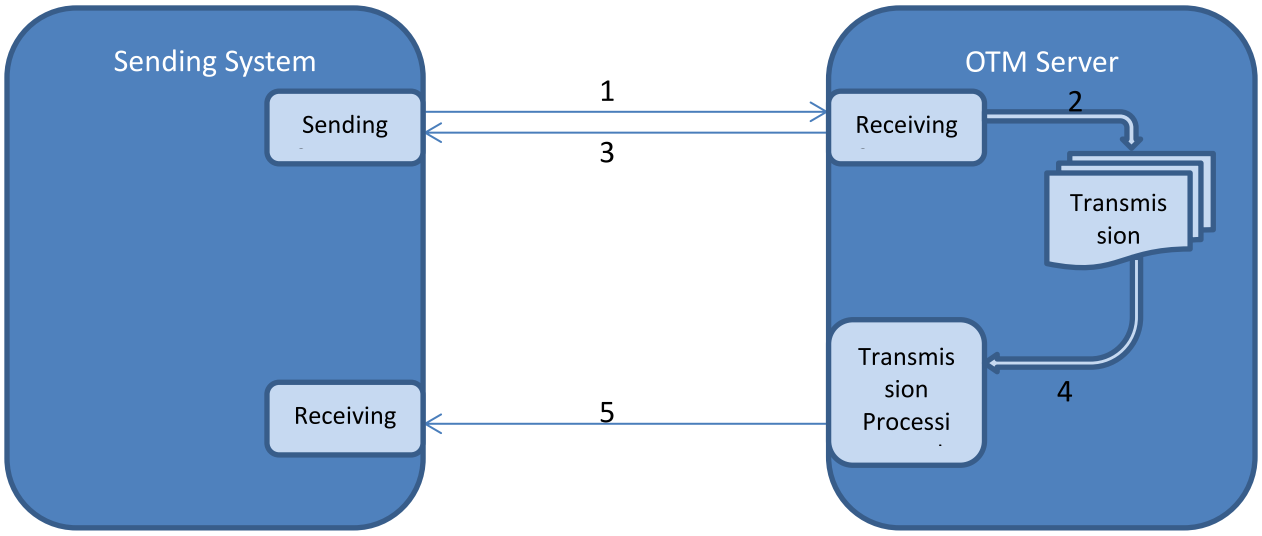


**Figure 17-1**

1. Message XML document is sent to OTM. The nature of the sending component depends on the transport protocol used e.g. it will be an HTTP client if sending message via HTTP POST.
2. Mobile message is persisted generating a unique Message Reference ID in the process. If the Message Body contains actionable content a Transmission record is created and staged.
3. A MessageAck document is returned which will contain the Message Reference ID. The nature of the response depends on the transport protocol used e.g. if using HTTP it will be a synchronous response to the initial HTTP POST.
4. If the Message Body contains actionable content a Transmission XML will have been staged for processing. The background Transmission processing daemon retrieves the Transmission at some time later and processes it e.g. creates or updates the business object using the original XML message content.
5. A MessageReport is optionally sent to confirm the status (PROCESSED or ERROR) of the actionable content. Again the report message contains the Message Reference ID for correlation.

## Processing Overview

If the servlet receives a Message XML, the integration module does the following:

1. Extracts authentication information from the MessageHeader. This information can be in the HTTP header instead.
2. Validates username and password.
3. Stores an entry in the I_message table, and the content in the I_transmission table.
4. Sends a MessageAck back to the sender as the synchronous response.
5. Checks for presence of MessageTypeGid and StylesheetProfileGid. If neither are present, checks for the MessageProfileGid, and uses the StylesheetProfileGids specified in the Message Profile to determine the correct Stylesheet Profile.
6. Using the Stylesheet Profile, the message is parsed to extract designated fields and updates the message table with extracted fields as needed.
7. Using the Message Type specified, the module performs the following:
    a. Notifies Message Center recipients specified on the Message Type
    b. Associates or disassociates devices to drivers, equipments, and power units as indicated on the Message Type
    c. Generates a Transmission XML if indicated on the Message Type. The Transmission XML would then be processed as indicate in the Transmission Processing section above.
8. Sends a MessageReport with validation and processing errors; Depending on your property settings, Oracle Transportation Management might only send a MessageReport if there are errors.

## Message Status

Each message sent into Oracle Transportation Management has a status field that indicates the state of the message. In addition, the content of the message is stored in the transmission table which also has its associated status. You can view the status for a message in the Message Hub Manager UI that can be accessed via the following menu: **Business Process Automation > Integration > Message Hub Manager**. The status of the message, and its related transmission, could be one of the following:

| Message Status | Related Transmission Status | Description |
|---|---|---|
| RECEIVED | MESSAGE | Indicates that the message is initially received in the message table, with the content stored in the transmission table. |
| PROCESSING | MESSAGE or<br><br>(dependent on Transmission processing) | Indicates the message is being processed. If the Message Type for the received message indicates a Transmission is to be generated and processed, then the status of the Transmission will be initially changed to FRESH and will be handled by normal transmission processing. |

| **Message Status** | **Related Transmission Status** | **Description** |
|---|---|---|
| PROCESSED | MESSAGE or<br><br>(dependent on Transmission processing) | Indicates that the message successfully completed processing. |
| ERROR | MESSAGE or<br><br>(dependent on Transmission processing) | Indicates that the message had completed processing and there were errors in the processing. |

## Web Service Integration

Mobile Device Communication messages can be sent inbound to OTM/GTM by calling the MessageService Web Service. The service is a SOAP Document Literal BARE style service.

The service has the following operations:

- `receiveMessage`: Accepts Message XML for staging and processing
- `receiveMessageAck`: Accepts acknowledgment MessageAck XML in response to an outbound Message XML

By default the WS-Security policy requires a Username Token and to be transported over HTTPS. Refer to the OTM Security Guide for details on how to modify this configuration.

## HTTP Integration

Mobile Device Communication messages can be sent inbound to OTM/GTM by sending via HTTP POST to the `glog.integration.servlet.WMServlet`. The servlet accepts both a Message and MessageAck XML document and returns a corresponding MessageAck in both cases.

## Direct DB Insert

Direct XML Insert is a way of sending inbound Mobile Communication Message XML to Oracle Transportation Management. The documents are inserted directly into the DB via a new PL/SQL procedure – `insert_message` – in the `pkg_integration_util` package. There is no corresponding outbound procedure.

The procedure signature is as described below:

```
insert_message (        p_username IN VARCHAR2,
                        p_password IN VARCHAR2,
                        p_message IN CLOB,
                        p_message_gid OUT VARCHAR2,
                        p_data_queue IN VARCHAR2 DEFAULT NULL,
                        p_cluster_gid IN VARCHAR2 DEFAULT NULL,
                        p_priority IN NUMBER DEFAULT 0)
```

| Parameter | Description |
|---|---|
| p_username | **IN.** OTM user account under who's authority XML will be processed. |
| p_password | **IN.** Password for OTM user. |
| p_message | **IN.** Message XML message. |
| p_message_gid | **OUT.** Unique Message GID generated for Message XML message. |
| p_data_queue | **IN. Default NULL.** Data Queue Definition GID that event will be handled by. |
| p_cluster_gid | **IN. Default NULL.** Cluster that will process the events.<br><br>If this parameter is specified on insert then the property `glog.dataqueue.eventCluster` must be set equal to the Cluster GID on all Application Servers in the cluster which should process the events. |
| p_priority | **IN. Default 0.** Priority of event. Used by Poller to order sequence of events for Preemptive Poller. |

### *Security*

The Oracle Database user DIR_XML_USER has all required privileges and synonyms to successfully execute the procedure described above.

## Internal Processing

The Message XMLs inserted using this procedure are processed using an internal Integration Data Queue – INTEGRATION IN DIRECT XML and can be monitored in exactly the same way as Integration Data Queue events covered in Chapter 11.

Figure 17-2 shows the processing for directly inserted XML messages.

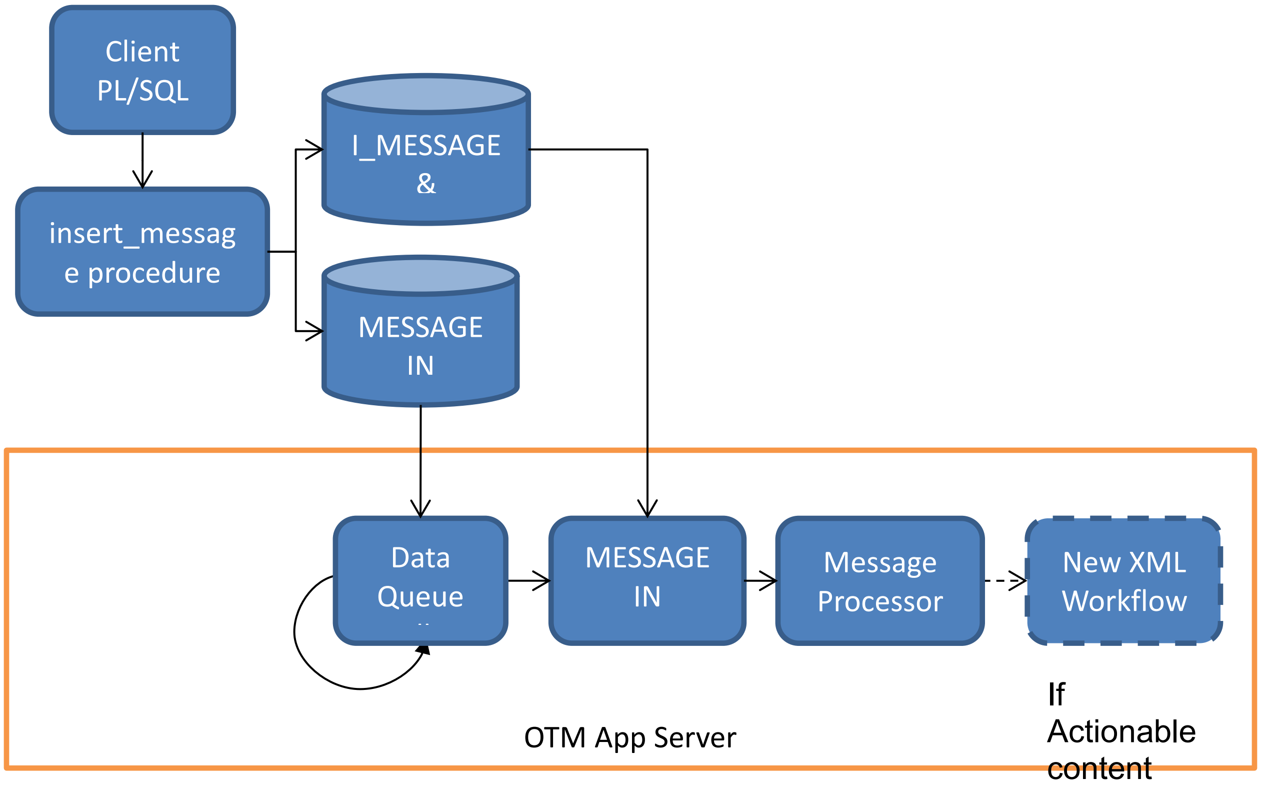


**Figure 17-2**

1. XML message passed to `insert_message` procedure.
2. OTM procedure stores data in I_MESSAGE table, I_TRANSMISSION table, data queue event in Message In Data Queue and returns Message ID to caller.
3. The Poller retrieves a number of events and calls the Executor for each event.
4. The Executor retrieves the message XML and calls the Message Processor to process the Message. If the content is "actionable", a Transmission XML is generated and New XML process is triggered.

# 18. Transactional Data Pre-loading (DirLoad)

You can get Oracle Transportation Management to load your inbound transmissions into the database faster without involving the application server. This is good when you just want to pre load data into Oracle Transportation Management and process the data later, like during setup of Oracle Transportation Management.

DirLoadServlet only supports these interfaces:

- TransOrder: You must include the GID to be able to have the application server offline.
- Shipment
- ShipmentLink
- TenderResponse
- Location
- Item
- ItemMaster
- HazmatGeneric
- HazmatItem
- ShipmentStatus
- Invoice
- Release
- ShipmentGroup
- SShipUnit
- Sku
- SkuTransaction
- Contact
- TransOrderStatus

  **Note:** The DirLoadServlet does not raise lifetime events (like shipment - created for ActualShipment) so automation agents cannot be triggered. Therefore, care must be taken to ensure that any objects created via DirLoadServlet will be complete and in the expected state.

  **Note:** Oracle Transportation Management ignores your AckSpec element. Instead, the DirLoadServlet HTTPPOSTs the TransmissionAck back to the IP address you sent your Transmission from.

To do this:

1. Make sure your transmissions only use the transaction code I.

   It is possible to use other transaction codes but with the limitation that you need to make sure that no user accesses that data through the application server while you update/delete your data. If a user accesses the data, you need to restart your application server after uploading your data to refresh its caches. To use other transaction codes with the DirLoadServlet you need to enable them in glog.properties.
2. If you load many transactions and want to increase loading speed, you can increase the number of threads assigned to load the data in glog.properties.
3. Post XML transmissions to http://hostname/GC3/glog.integration.servlet.DirLoadServlet
4. DirLoadServlet saves your data to the database.

DirLoadServlet sets default statuses for business objects you insert.

# 19. Workflow Web Service

There are several web services which provide access to trigger agent workflow for selected business objects. The capabilities include:-

- Execution of individual agent action against one or more objects of a supported type.
- Execution of multiple agent actions against one or more objects of a supported type.

Currently the supported object types are SHIPMENT (Buy Shipment), SELL SIDE SHIPMENT, ORDER RELEASE, DRIVER and ORDER MOVEMENT.

There is a separate service for each business object type:

- ShipmentService
- SellSideShipmentService
- OrderReleaseService
- OrderMovementService
- DriverService
- AgentService

The type specific services handle individual agent actions for that type and have one operation: `processAction`. The AgentService handles execution of multiple actions essentially identical to an Oracle Transportation Management Agent but with some constraints (covered later) and also has one operation: `processAgent`.

All services are implemented with the synchronous REQUEST/RESPONSE messaging model. However, with respect to Oracle Transportation Management agent action processing, the response indicates that the action has been scheduled successfully. This is due to the fact that the Oracle Transportation Management application could have a significant amount of workflow triggered by such an action. Therefore waiting for completion may require an excessive transaction timeout value.

The input and output messages for each service are specified in the service XSDs. Namely:

- AgentService.xsd
- ShipmentService.xsd
- SellSideShipmentService.xsd
- OrderReleaseService.xsd
- OrderMovementService.xsd

Agent specific message content is specified in the following XSDs:

- **Agent.xsd**: contains agent header details
- **ShipmentAction.xsd**: contains all actions related to buy shipments
- **SellSideShipmentAction.xsd**: contains all actions related to sell shipments
- **OrderReleaseAction.xsd**: contains all actions related to order releases
- **OrderMovementAction.xsd**: contains all actions related to order movements

  **Note:** Due to a change in v 6.2 in how these services are deployed in the application server at runtime, the WSDL for each service will not directly reference the XSD files mentioned above. However, the formats defined in these XSD files will be functionally equivalent to the definitions referenced in the WSDL.

### AgentService constraints

Although in theory a complete agent could be defined using the schemas, some actions will not be supported in the initial version. These actions are known as the BLOCK actions: IF, ELSE, FOR EACH, etc.

### Version Control

The service and message definition schemas are under version control starting from version 1.0, i.e. major version number is 1, and minor version number is 0. The current target namespace for each schema will contain the major version number, for example the target namespace for version 1.0 of Agent.xsd is:

`http://xmlns.oracle.com/apps/otm/agent/v1`

As of version 6.2, the current version of the Agent Action web services has increased to “v2”. However, to support backward compatibility, all “v1” messages will still be valid.

### Example Service Message

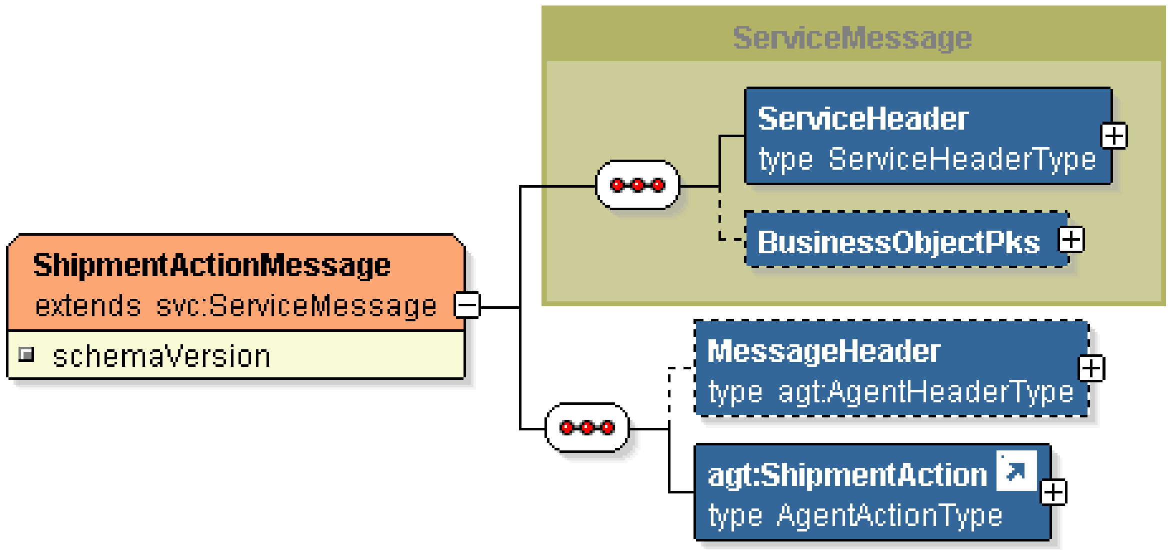


**Figure 19-1**

All service messages extend the `ServiceMessage` complexType (defined in the Service.xsd schema).

The Agent Action messages then include Agent Header information (`MessageHeader`) followed by an Agent Action, in this example `ShipmentAction`. All valid Shipment Actions are defined in the ShipmentAction.xsd schema. The following example shows the Set Indicator Action:

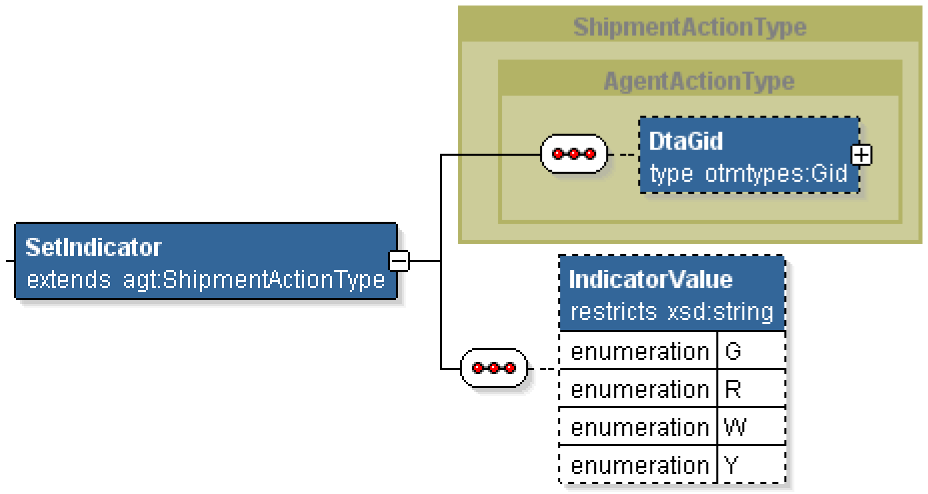


**Figure 19-2**

The `SetIndicator` element is allowed to be substituted for the `ShipmentAction` element because it is defined as an XSD **substitutionGroup** and extends the same `AgentActionType` complexType.

## *Changes between Version 1.0 and Version 2.0*

### Optional Username and Password elements

The initial version of the Service XSD schema had the username and password element as "required". Due to the added support for WS-Security this is now an unnecessary restriction.

### Removal of 'Fields' element

The agent action element part of the message e.g. "SetIndicator", in version 1.0 required the action fields to be wrapped in an outer Fields element. Due to improvements in the underlying web service standards supported (e.g. JAXB) this is now not required resulting in a smaller message size.

# 20. REST API Reference

This section describes the features available through the REST API for creating, retrieving, modifying and removing OTM & GTM persistent data content for User Interface purposes.

> **NOTE:** This API is for User Interface Integration and does not replace the current application integration services which should continue to be used for all business process based application integration.

## Authentication and Authorization

User authentication and authorization are secured via the HTTP Basic Authentication headers using the OTM username and password.

Consequently, to secure the username and password within each REST API call, the transport should be performed using a secure protocol e.g. HTTPS.

Access Control Lists exist for View and Update of each entity supported in the OTM/GTM ReST API.

Refer to the Security Guide for instructions on how to add these ACLs to the user or user role.

## Resource Reference

Data content in the REST API is associated with an *Entity Name*. The Entity Name would normally be the camel case of the primary table name, e.g. *Location, Contact or ShipmentStop* .

Page /GC3/api/RestApiList generates a list of supported entities along with links to the metadata describing the data format of these entities.

The entity itself would contain the data from the primary table and all associated tables. Tables are associated if they satisfy one of the following rules:

- Child table (the primary key contains the parent table primary key and so cannot exist without the parent).
- Joined table (the table imports the primary key of the parent table as a foreign key but can exist independently).

The associations are managed by the OTM application to ensure data integrity and optimized concurrency. The logic required for this is over and above the constraints defined in the Oracle database (internally this is managed by metadata known as Business Maps).

The current set of entities is based on best practices, as regards optimal data content, developed over many years of OTM/GTM application User Interface design.

There is also the concept of Entity Alias. This is used to support common alternative entity names for objects which may not match the actual table name. For example, the entity name BuyShipment can be used to show that the data retrieved from the SHIPMENT table is intended to be for a Buy Shipment.

> **Note:** Currently, for the above example, the 'perspective' is not checked on actual data retrieval so, in theory, it would be possible to retrieve data for a Sell Shipment ID using the BuyShipment entity name. However, the required discrimination will be added in a future version of the service and so the correct entity name should be used where possible.

**Note:** Many entities in OTM/GTM are 'statuseable' i.e. they can possess various statuses which can change throughout the life of the object to reflect the various states in a business process. For example, shipments can be ENROUTE - NOT STARTED, ENROUTE – ENROUTE and ENROUTE COMPLETED. The states are controlled by the internal workflow of the OTM/GTM application and so should **not** be modified via the REST API. The various statuses can be seen when objects are retrieved but should not be sent back to the application in any create or modify message.

The following resources are supported by this API.

| Task | REST Resource |
|---|---|
| View Data Content | GET /api/sdo/{entity name}?id={col 1}&id={col 2} |
| Create Data Content | PUT /api/sdo/{entity name}?id={col 1}&id={col 2} |
| Modify Data Content | POST /api/sdo/{entity name}?id={col 1}&id={col 2} |
| Remove Data Content | DELETE /api/sdo/{entity name}?id={col 1}&id={col 2} |
| View Metadata | GET /api/metadata/{entity name} |

## *View Data Content*

Retrieve data for a single entity instance identified by primary key.

### REST Resource

```
GET /api/sdo/{entity name}?id={col 1}&id={col 2}
```

### Parameters

| Name | Description |
|---|---|
| entity name | Identifies the name of the entity to be retrieved e.g. BuyShipment, OrderRelease etc. |
| id | Specifies the primary key column(s) to be used for retrieval of a single object. If the entity primary key has multiple columns there should be multiple id query parameters in the column order defined by the table's primary key constraint. |

### Response Body

Media Types: application/json.

The response body contains the JSON string representing the requested entity. An attribute will be present for each column of the associated table and object arrays will be present for each child object.

See View Metadata for a way to see an XML schema definition representation of the entity.

### cURL example

```
$ curl -i -X GET -u GUEST.ADMIN:CHANGEME
"http://localhost:8080/GC3/api/sdo/BuyShipment?id=GUEST.MTH-001"
```

### Example Response Header

```
HTTP/1.1 200 OK
Date: Mon, 15 Jun 2015 12:24:19 GMT
Server: Apache
X-ORACLE-DMS-ECID: 6eb06920-2fd6-4b9d-b0e0-4cf7128db22f-00000036
X-FRAME-OPTIONS: SAMEORIGIN
Set-Cookie: JSESSIONID=HXX3LXCKJTAVwhHVR5Fpj9GN73qhEJYH-Bi472IfLK3dYy2HQM5I!-
69508724; path=/; HttpOnly
Vary: User-Agent
Transfer-Encoding: chunked
Content-Type: application/json
```

### Example Response Body

The following example shows a Buy Shipment retrieved containing one child entity for the Shipment Involved Party and one child entity for Shipment Refnum.

Note that child object arrays will be structured where each child object will contain a transactionCode attribute and a *childName*BeanData attribute. On Data retrieval the transaction code will always be "NP". This attribute is intended for use by the data modification resources.

```
{"BuyShipment":
  {"shipmentGid":"GUEST.MTH-001",
   "shipmentXid":"MTH-001",
   "shipmentName":"MOD 1",
   "isTemplate":false,
   "isPrimary":false,
   "isSpotCosted":false,
   "isCreditNote":false,
   "totalActualCost":
      {"value":0.0,
       "uomCode":"USD"},
   "totalWeightedCost":
      {"value":0.0,
       "uomCode":"USD"},
   "loadedDistance":
      {"value":0.0,
       "uomCode":"MI"},
   "unloadedDistance":
      {"value":0.0,
       "uomCode":"MI"},
   "sourceLocationGid":"DEN",
   "destLocationGid":"LAX",
   "startTime":
      {"date":"2015-02-26T13:10:00.000-07:00"},
   "endTime":
      {"date":"2015-02-28T12:10:00.000-08:00"},
   "isAutoMergeConsolidate":false,
   "perspective":"B",
   …etc… ,
   "shipmentInvolvedParty":
      [{"transactionCode":"NP",
        "shipmentInvolvedPartyBeanData":
           {"shipmentGid":"GUEST.MTH-001",
            "involvedPartyQualGid":"BILL-TO",
            "involvedPartyContactGid":"GUEST.ADMIN",
            "comMethodGid":"FAX",
            "domainName":"GUEST"}
      }],
   "shipmentRefnum":
      [{"transactionCode":"NP",
        "shipmentRefnumBeanData":
           {"shipmentGid":"GUEST.MTH-001",
            "shipmentRefnumQualGid":"GLOG",
            "shipmentRefnumValue":"GUEST.MTH-001",
            "domainName":"GUEST"}
      }]
   }
}
```

## *Create Data Content*

Insert a new single entity instance into the database.

### REST Resource

```
PUT /api/sdo/{entity name}
```

### Parameters

| Name | Description |
|---|---|
| entity name | Identifies the name of the entity to be retrieved e.g. BuyShipment, OrderRelease etc. |

### Request Body

Media Types: application/json.

The request body contains the JSON string representing the entity to be created. An attribute will be present for each column of the associated table and object arrays will be present for each child object to be inserted. Each child object must have the transaction code attribute value of "I" for insert.

> **NOTE:** Some entities auto-generate certain child objects e.g. the Reference Number with Qualifier "GLOG". No attempt should be made to create or modify these qualifiers and the transaction code for these elements should remain as "NP" for Not Persisted.

See View Metadata for a way to see an XML schema definition representation of the entity.

### Response Body

There is no response for this request.

### cURL example

```
$ curl -i -X PUT -u GUEST.ADMIN:CHANGEME -d @shipment.json
"http://localhost:8080/GC3/api/sdo/BuyShipment"
```

### Example Request Body

```
{"BuyShipment":
  {"shipmentGid":"GUEST.MTH-002",
   "shipmentXid":"MTH-002",
   "shipmentName":"ORIGINAL SHIPMENT NAME",
   "isTemplate":false,
   "isPrimary":false,
   "isSpotCosted":false,
   "isCreditNote":false,
   "totalActualCost":
      {"value":0.0,
       "uomCode":"USD"},
   "totalWeightedCost":
      {"value":0.0,
       "uomCode":"USD"},
   "loadedDistance":
      {"value":0.0,
       "uomCode":"MI"},
   "unloadedDistance":
      {"value":0.0,
       "uomCode":"MI"},
   "sourceLocationGid":"DEN",
   "destLocationGid":"LAX",
   "startTime":
      {"date":"2015-02-26T13:10:00.000-07:00"},
   "endTime":
      {"date":"2015-02-28T12:10:00.000-08:00"},
   "isAutoMergeConsolidate":false,
   "perspective":"B",
   …etc… ,
   "shipmentInvolvedParty":
      [{"transactionCode":"I",
        "shipmentInvolvedPartyBeanData":
           {"shipmentGid":"GUEST.MTH-002",
            "involvedPartyQualGid":"BILL-TO",
            "involvedPartyContactGid":"GUEST.ADMIN",
            "comMethodGid":"FAX",
            "domainName":"GUEST"}
      }],
   "shipmentRefnum":
      [{"transactionCode":"NP",
        "shipmentRefnumBeanData":
           {"shipmentGid":"GUEST.MTH-002",
            "shipmentRefnumQualGid":"GLOG",
            "shipmentRefnumValue":"GUEST.MTH-002",
            "domainName":"GUEST"}
      }]
   }
}
```

### Example Response Header

```
HTTP/1.1 200 OK
Date: Mon, 15 Jun 2015 12:24:19 GMT
Server: Apache
X-ORACLE-DMS-ECID: 6eb06920-2fd6-4b9d-b0e0-4cf7128db22f-00000036
X-FRAME-OPTIONS: SAMEORIGIN
```

```
Set-Cookie: JSESSIONID=HXX3LXCKJTAVwhHVR5Fpj9GN73qhEJYH-Bi472IfLK3dYy2HQM5I!-
69508724; path=/; HttpOnly
Vary: User-Agent
Transfer-Encoding: chunked
Content-Type: application/json
```

## *Modify Data Content*

Modify one or more attributes or child objects for a new single entity which already exists in the database.

### REST Resource

```
POST /api/sdo/{entity name}?id={col 1}&id={col 2}
```

### Parameters

| Name | Description |
|---|---|
| entity name | Identifies the name of the entity to be retrieved e.g. BuyShipment, OrderRelease etc. |
| id | Specifies the primary key column(s) to be used for retrieval of a single object. If the entity primary key has multiple columns there should be multiple id query parameters in the column order defined by the table's primary key constraint. |

### Request Body

Media Types: application/json.

The request body contains the JSON string representing the requested entity. An attribute will be present for each column of the associated table and object arrays will be present for each child object.

See View Metadata for a way to see an XML schema definition representation of the entity.

### cURL example

```
$ curl -i -X PUT -u GUEST.ADMIN:CHANGEME -d @shipment.json
"http://localhost:8080/GC3/api/sdo/BuyShipment?id=GUEST.MTH-001"
```

### Example Request Body

```
{"BuyShipment":
  {"shipmentGid":"GUEST.MTH-002",
   "shipmentName":"NEW SHIPMENT NAME"
  }
}
```

Only attributes to be modified need to be present. All other attributes will remain unchanged.

### Example Response Header

```
HTTP/1.1 200 OK
Date: Mon, 15 Jun 2015 12:24:19 GMT
```

```
Server: Apache
X-ORACLE-DMS-ECID: 6eb06920-2fd6-4b9d-b0e0-4cf7128db22f-00000036
X-FRAME-OPTIONS: SAMEORIGIN
Set-Cookie: JSESSIONID=HXX3LXCKJTAVwhHVR5Fpj9GN73qhEJYH-Bi472IfLK3dYy2HQM5I!-
69508724; path=/; HttpOnly
Vary: User-Agent
Transfer-Encoding: chunked
Content-Type: application/json
```

## *Remove Data Content*

Remove a single entity instance from the database. Due to the often complex inter-relationships between transactional objects, it is not recommended to use this API for removal of shipments, orders, invoices etc.

Use of this API should be considered carefully and only used where the full impact of the transaction is known.

### REST Resource

```
DELETE /api/sdo/{entity name}?id={col 1}&id={col 2}
```

### Parameters

| **Name** | **Description** |
|---|---|
| entity name | Identifies the name of the entity to be retrieved e.g. BuyShipment, OrderRelease etc. |
| id | Specifies the primary key column(s) to be used for retrieval of a single object. If the entity primary key has multiple columns there should be multiple id query parameters in the column order defined by the table’s primary key constraint. |

### cURL example

```
$ curl -i -X DELETE -u GUEST.ADMIN:CHANGEME
"http://localhost:8080/GC3/api/sdo/BuyShipment?id=GUEST.MTH-001"
```

### Example Response Header

```
HTTP/1.1 200 OK
Date: Mon, 15 Jun 2015 13:24:02 GMT
Server: Apache
Content-Length: 0
X-ORACLE-DMS-ECID: 6eb06920-2fd6-4b9d-b0e0-4cf7128db22f-000000f2
X-FRAME-OPTIONS: SAMEORIGIN
Set-Cookie: JSESSIONID=AGD3ZBoHx4JneOt2DAkNKP2VkI5bTP5TmGFNusr30mUf7YWgf0tK!-
69508724; path=/; HttpOnly
Vary: User-Agent
Content-Type: text/html; charset=ISO-8859-1
```

## *View Metadata*

Retrieve XML Schema Definition describing the JSON content for a named entity.

It is anticipated that users of the View and Modify REST API resources will want to serialize and deserialize the JSON string content to/from application classes to make UI development more straightforward.

For example, an XML schema can be used to generate JAXB annotated classes that can then be used by the EclipseLink JAXB & JSON implementation (MOXy) to serialize and deserialize to/from JSON strings and Java classes.

See http://www.eclipse.org/eclipselink/ for details.

### REST Resource

```
GET /api/metadata/{entity name}
```

### Parameters

| Name | Description |
|---|---|
| entity name | Identifies the name of the entity to be retrieved e.g. BuyShipment, OrderRelease etc. |

### Response Body

Media Types: application/xml.

The response body contains an XML schema definition. An attribute will be present for each column of the associated table and an anonymous complex type definition will be present for each child entity type.

The response body can be quite a large document but is really only required during API development and not as part of the runtime execution.

### cURL example

```
$ curl -i -X GET -u GUEST.ADMIN:CHANGEME
"http://localhost:8080/GC3/api/metadata/BuyShipment"
```

### Example Response Header

```
HTTP/1.1 200 OK
Date: Mon, 15 Jun 2015 16:52:31 GMT
Server: Apache
X-ORACLE-DMS-ECID: 6eb06920-2fd6-4b9d-b0e0-4cf7128db22f-00000357
X-FRAME-OPTIONS: SAMEORIGIN
Set-Cookie: JSESSIONID=OND4Ivr8NRMqK9DV61oNiHmTRdMEpXi0p3D82kAHDlT23UX6oAKM!-
69508724; path=/; HttpOnly
Vary: User-Agent
Transfer-Encoding: chunked
Content-Type: application/xml
```

## Example Response Body

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:otm="http://xmlns.oracle.com/apps/otm"
xmlns:cil="http://xmlns.oracle.com/apps/otm/cil"
xmlns="http://xmlns.oracle.com/apps/otm/cil"
targetNamespace="http://xmlns.oracle.com/apps/otm/cil">
<xsd:element name="BuyShipment">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="shipmentGid" type="xsd:string"/>
         <xsd:element name="shipmentXid" type="xsd:string"/>
         <xsd:element name="shipmentName" type="xsd:string"/>
         <xsd:element name="transportModeGid" type="xsd:string"/>
         <xsd:element name="isTemplate" type="xsd:boolean"/>
         <xsd:element name="isPrimary" type="xsd:boolean"/>
         <xsd:element name="isSpotCosted" type="xsd:boolean"/>
         <xsd:element name="isCreditNote" type="xsd:boolean"/>
         <xsd:element name="cmPrevDestLocationGid" type="xsd:string"/>
         <xsd:element name="cmNextSourceLocationGid" type="xsd:string"/>
         <xsd:element name="totalActualCost" type="MeasureType"/>
         <xsd:element name="totalWeightedCost" type="MeasureType"/>
         <xsd:element name="totalTransCost" type="MeasureType"/>
         <xsd:element name="loadedDistance" type="MeasureType"/>
         <xsd:element name="unloadedDistance" type="MeasureType"/>
         <xsd:element name="sourceLocationGid" type="xsd:string"/>
         <xsd:element name="destLocationGid" type="xsd:string"/>
         <xsd:element name="startTime" type="DateType"/>
         <xsd:element name="endTime" type="DateType
         …etc…
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

The common types of *MeasureType* and *DateType* are defined in each schema.

## REST Resource

```
GC3/api/auth
```

This REST service is used to authenticate the Mobile user in OTM calls and it is attached in the Security setup in an Oracle Mobile Application.

### Parameters

| **Name** | **Description** |
|---|---|
| Encoded Authorization | After the user logs in while using the Oracle Transportation Mobile application, the auth API is invoked by attaching the cookies with the credentials to the Rest call header. A special setup is needed in the security section of the maf-application.xml |

### Response

```
AUTH: {"userId": "GUEST.ADMIN",
"roles": ["SERVPROV"]}
```

### REST Resource

```
GC3/api/login
```

Uses the REST header to send the user credentials. Returns the authenticated user.

### Parameters

| Name | Description |
|---|---|
| None | |

### REST Resource

```
GC3/api/shipment
```

This is the first call to OTM to bring all the shipments for the user down to the local Database on the user's device. In addition to shipments it brings all the shipment's children tables such as Shipment_Stops, Remarks, events, etc.

### Parameters

| Name | Description |
|---|---|
| None. | |

### Request Body

Media Types: `application/json.`

The request body contains the JSON string representing the user logon in credentials to retrieve shipments for the user sent in the request header.

### Response Body

The response body includes OTM default value for the Maximum Session Timeout and shipments with shipment's children tables. Finally, Quick Codes are also retrieved during this call.

```
{"userid":"GUEST.ADMIN",
"maxSessionTimeout":2,
"shipments":[{"shipmentGid":"GUEST.01008",
"transactionNumber":4872089,
"tenderType":"Ordinary",
"isTendered":true,
"tenderedAction":"A",
"servprovGid":null,
"isException":true,
"progressStatus":"I",
"indicator":"R",
"respondByDate":1471877801000,
"respondByDateTz":"UTC",
"respondByDateOffset":0,
"equipmentGid":"28FT DRY VAN",
"totalWeightUP":23900.0,
"totalWeightUomUP":"LB",
"totalWeightAE":23900.0,
"totalWeightUomAE":"LB",
"totalVolumeUP":0.0,
"totalVolumeUomUP":"CUFT",
"totalVolumeAE":0.0,
"totalVolumeUomAE":"CUFT",
"freightCostUP":600.0,
"freightCostUomUP":"USD",
"freightCostAE":600.0,
"freightCostUomAE":"USD",
"contactName":"ORACLE GUEST",
"contactPhone":"850-687-6860",
"contactEmail":"oracle.guest@oracle.com",
"isHazardous":true,
"totalDistanceUP":1167.99,
"totalDistanceUomUP":"MI",
"totalDistanceAE":1167.99,
"totalDistanceUomAE":"MI",
"totalTime":185832548,
"delayedTime":185676453,
"firstStopNumber":1,
"lastStopNumber":2,
"isMultiStop":false,
"isPreShipmentEvent":false},
{"shipmentGid":"GUEST.01025",
"transactionNumber":4894340,
"tenderType":"Ordinary",
"isTendered":true,
"tenderedAction":"A"
,"servprovGid":null,
"isException":true,
"progressStatus":"C",
"indicator":"R",
"respondByDate":1472746276000,
"respondByDateTz":"UTC",
"respondByDateOffset":0,
"equipmentGid":"28FT DRY VAN",
"totalWeightUP":3050.0,
"totalWeightUomUP":"LB",
"totalWeightAE":3050.0,
"totalWeightUomAE":"LB",
"totalVolumeUP":0.0,
"totalVolumeUomUP":"CUFT",
"totalVolumeAE":0.0,
"totalVolumeUomAE":"CUMTR",
"freightCostUP":600.0,
"freightCostUomUP":"USD",
"freightCostAE":600.0,
"freightCostUomAE":"USD",
"contactName":null,
"contactPhone":null,
"contactEmail":null,
"isHazardous":false,
"totalDistanceUP":1170.5,
"totalDistanceUomUP":"MI",
"totalDistanceAE":1170.5,
"totalDistanceUomAE":"MI",
"totalTime":0,
"delayedTime":1946,
"firstStopNumber":1,
"lastStopNumber":2,
"isMultiStop":false,
"isPreShipmentEvent":false}],
"quickCodes":[{"quickCodeGid":"GUEST.1PU",
"quickCodeDesc":"PICKUP"},
{"quickCodeGid":"GUEST.ARRIVED",
"quickCodeDesc":"ARRIVED AT LOCATION"},
{"quickCodeGid":"GUEST.CA",
"quickCodeDesc":"Carrier Arrived"},
{"quickCodeGid":"GUEST.CDPU",
"quickCodeDesc":"Carrier Departed Pickup Location"},]}
```

### REST Resource

```
GC3/api/shipment/tender
```

### Request Body

Media Types: `application/json.`

The request body contains the JSON string representing the shipmentGid to accept or decline and the user accepting or declining it.

```
{"tenderType":"Ordinary","shipmentGid":"GUEST.01060","transactionNumber":486604
8,"type":"A","servprovGid":"GUEST.3939"}
```

### Parameters

| Name | Description |
|---|---|
| tenderType | OTM Tender type’s such as ordinary |
| shipmentGid | ShipmentGid to accept or decline |
| transactionNumber | |
| type | A for accepted or D for decline |
| ServprovGid | The transportist user accepting or declining the shipment |

### Response Body

Response includes OK or 500 if an error occurred while trying to process the acceptance or decline in OTM.

### REST Resource

```
GC3/api/shipment/add_event
```

### Request Body

Media Types: `application/json.`

The request body contains the JSON string representing the Event data Entered in the Oracle Transportation Mobile Add Event UI, including the shipment related data e.g. shipmentGid as well as any image associated with the event such as signature or pictures. Images such as signatures and pictures should be passed as Base64 encoded strings. The header request adds the user credentials to the REST service call.

```
{"stopNumber":1,"eventDateTz":"America/New_York","shipmentGid":"GUEST.01059","e
ventDateOffset":0,"latitude":39.77264,"quickCode":"GUEST.PUC","eventDateMillis"
:1450626780000,"longitude":-
105.04914,"images":[{"fileEncoded":[47,57,106,47,52,65,65…] }]}
```

### Parameters

| Name | Description |
|---|---|
| shipmentGid | The shipment GID |
| stopNumber | The stop where the event is occurring. Current Location or actual stop number. Current Location is identified as -1 |
| Latitude | The position of the stop |
| longitude; | The position of the stop |
| eventDateMillis | Date Entered in UI sent in milliseconds |
| eventDateTz | Date Time Zone |
| eventDateOffset | Date format |
| quickCode | OTM |
| remarkText | Remark entered in the UI |
| List<Image> images | Pictures attached to the event.<br><br>Should be passes as Base64 encoded String. |
| signature | The signature attached to the event<br><br>Should be passes as Base64 encoded String. |

| Name | Description |
|---|---|
| name | The name of the person who signed the event. |

## Response Body

Response includes the JSON String that contains the current shipment in addition to all the shipment’s children table including the events. Or 500 if an error occurred while trying to process the event in OTM. Or 401 if the user is not authenticated.

```
{"userid":null,"maxSessionTimeout":0,"shipments":[{"shipmentGid":"GUEST.01008",
"transactionNumber":4872089,"tenderType":"Ordinary","isTendered":true,"tendered
Action":"A","servprovGid":"GUEST","isException":true,"progressStatus":"C","indi
cator":"R","respondByDate":1471877801000,"respondByDateTz":"UTC","respondByDate
Offset":0,"equipmentGid":"28FT DRY
VAN","totalWeightUP":23900.0,"totalWeightUomUP":"LB","totalWeightAE":23900.0,"t
otalWeightUomAE":"LB","totalVolumeUP":0.0,"totalVolumeUomUP":"CUFT","totalVolum
eAE":0.0,"totalVolumeUomAE":"CUFT","freightCostUP":600.0,"freightCostUomUP":"US
D","freightCostAE":600.0,"freightCostUomAE":"USD","contactName":"GUEST
ORACLE","contactPhone":"850-687-
6860","contactEmail":"GUEST@oracle.com","isHazardous":false,"totalDistanceUP":1
170.5,"totalDistanceUomUP":"MI","totalDistanceAE":1170.5,"totalDistanceUomAE":"
MI","totalTime":234385332,"delayedTime":234229659,"firstStopNumber":1,"lastStop
Number":2,"isMultiStop":false,"isPreShipmentEvent":false}],"shipmentStops":[{"s
hipmentGid":"GUEST.01008","stopNumber":1,"locationGid":"GUEST.DENVER
LOCATION","locationName":"DENVER","locationCity":"DENVER","locationProvinceCode
":"CO","locationPostalCode":"80212","addressLine1":"3939 OSCEOLA
ST.","addressLine2":null,"addressLine3":null,"addressLine4":null,"addressLine5"
:null,"latitude":39.77264,"longitude":-105.04914,"contactName":"JOHN
DENVER","contactPhone":"303-867-
5309","contactEmail":"GUEST@oracle.com","isPickup":true,"isDelivery":false,"isO
ther":false,"appointmentStartTime":0,"appointmentStartTimeTz":null,"appointment
StartTimeOffset":0,"appointmentEndTime":0,"appointmentEndTimeTz":null,"appointm
entEndTimeOffset":0,"appointmentActivityType":null,"appointmentConfirmationNumb
er":null,"appointmentRemarkText":null,"plannedArrivalTime":1213646992000,"plann
edArrivalTimeTz":"America/Denver","plannedArrivalTimeOffset":0,"plannedDepartur
eTime":1213646992000,"plannedDepartureTimeTz":"America/Denver","plannedDepartur
eTimeOffset":-
21600000,"estimatedArrivalTime":1213646992000,"estimatedArrivalTimeTz":"America
/Denver","estimatedArrivalTimeOffset":-
21600000,"estimatedDepartureTime":1213646992000,"estimatedDepartureTimeTz":"Ame
rica/Denver","estimatedDepartureTimeOffset":-
21600000,"actualArrivalTime":0,"actualArrivalTimeTz":null,"actualArrivalTimeOff
set":0,"actualDepartureTime":0,"actualDepartureTimeTz":null,"actualDepartureTim
eOffset":0,"stopDisplayTime":1213646992000,"stopDisplayTimeTz":"America/Denver"
,"stopDisplayTimeOffset":-
21600000,"shipUnitCount":10,"weightUP":23900.0,"weightUomUP":"LB","weightAE":23
900.0,"weightUomAE":"LB","volumeUP":0.0,"volumeUomUP":"CUFT","volumeAE":0.0,"vo
lumeUomAE":"CUFT","progressStatus":"N","delayedTime":0,"indicator":"G","distanc
eFromPreviousUP":0.0,"distanceFromPreviousUomUP":"MI","distanceFromPreviousAE":
0.0,"distanceFromPreviousUomAE":"MI","timeFromPrevious":0,"isLastStop":false},{
"shipmentGid":"GUEST.01008","stopNumber":2,"locationGid":"GUEST.EUGENE
LOCATION","locationName":"EUGENE","locationCity":"EUGENE","locationProvinceCode
":"OR","locationPostalCode":"97402","addressLine1":"811 SPRUCE
ST.","addressLine2":null,"addressLine3":null,"addressLine4":null,"addressLine5"
:null,"latitude":44.05827,"longitude":-
123.18787,"contactName":null,"contactPhone":null,"contactEmail":null,"isPickup"
:false,"isDelivery":true,"isOther":false,"appointmentStartTime":0,"appointmentS
tartTimeTz":null,"appointmentStartTimeOffset":0,"appointmentEndTime":0,"appoint
mentEndTimeTz":null,"appointmentEndTimeOffset":0,"appointmentActivityType":null
,"appointmentConfirmationNumber":null,"appointmentRemarkText":null,"plannedArri
valTime":1213803087000,"plannedArrivalTimeTz":"America/Los_Angeles","plannedArr
ivalTimeOffset":0,"plannedDepartureTime":1213803087000,"plannedDepartureTimeTz"
:"America/Los_Angeles","plannedDepartureTimeOffset":-
25200000,"estimatedArrivalTime":1213803087000,"estimatedArrivalTimeTz":"America
/Los_Angeles","estimatedArrivalTimeOffset":-
25200000,"estimatedDepartureTime":1399479540000,"estimatedDepartureTimeTz":"
```

# 21. Appendix A – Integration Messages

This appendix lists integration messages, describes why the message occurs, and describes what you need to do as a result of receiving the message.

You might find these error messages in a TransmissionReport element.

| Heading | Data |
|---|---|
| Message: | public final static String Invalid Date Format Text = "THE DATE ELEMENT {0} WITH VALUE {1} IS NOT OF FORMAT YYYYMMDDHHMMSS"; |
| Occurs When: | An invalid date format error occurs when the date is not provided in the format YYYYMMDDHHMMSS. |
| Corrective Action: | Enter the date using the required format. |

| Heading | Data |
|---|---|
| Message: | public final static String dataConversionErrorText = "DATA CONVERSION FAILED FOR THE ELEMENT {0} WITH VALUE {1}"; |
| Occurs When: | A data conversion error occurs when character data cannot be converted to an internal data type. |
| Corrective Action: | Eliminate extraneous characters from the data element. |

| Heading | Data |
|---|---|
| Message: | public final static String duplicateKeyErrorText = "THE ELEMENT(S) {0} WITH VALUE(S) {1} IS A DUPLICATE PRIMARY KEY"; |
| Occurs When: | A duplicate Key error occurs when the primary key for a given element already exists in the G-Log database. |
| Corrective Action: | Change the Transaction Code element from I to IU. |

| Heading | Data |
|---|---|
| Message: | public final static String PKNotFoundErrorText = "THE PRIMARY KEY ELEMENT(S) {0} WITH VALUE(S) {1} COULD NOT BE FOUND IN TABLE {2} COLUMN(S) {3}"; |

| Heading | Data |
|---|---|
| Occurs When: | A PKNotFoundError most often occurs when an attempt is made to update or delete data that does not exist in the G-Log database. |
| Corrective Action: | If your Transaction Code is U, then use UI instead. If your Transaction Code is D, then there is no corrective action. |

| Heading | Data |
|---|---|
| Message: | public final static String FKNotFoundErrorText = "THE FOREIGN KEY ELEMENT {0} WITH VALUE {1} COULD NOT BE FOUND IN TABLE {2} COLUMN {3}"; |
| Occurs When: | A FK Not Found Error occurs when a referenced primary key does not exist in the GLog database. |
| Corrective Action: | Correct the XML data value such that it refers to a primary key that does exist in the G-Log database. |

| Heading | Data |
|---|---|
| Message: | public final static String missing RequiredElementErrorText = "THE REQUIRED ELEMENT {0} IS MISSING"; |
| Occurs When: | A missing required element error occurs when a required element has been omitted from a GLogXML element. |
| Corrective Action: | Provide the missing required element in your XML data. |

| Heading | Data |
|---|---|
| Message: | public final static String ITransactionNoNotFoundErrorText = "THE I_TRANSACTION_NO WITH VALUE {0} DOESN'T EXIST IN THE DATABASE"; |
| Occurs When: | An ITransactionNoNotFoundError occurs when a GLogXMLElement, such as a Tender Response, refers to a transaction number that does not exist in the G-Log I_TRANSACTION table. |
| Corrective Action: | Refer to an existing transaction number. |

| Heading | Data |
|---|---|
| Message: | public final static String invalidTransactionCodeErrorText = "THE TRANSACTION CODE {0} is not valid. Valid codes are I,U,D,IU,UI,NP"; |
| Occurs When: | An invalid transaction code error occurs when the Transaction Code element is specified with an invalid value. |
| Corrective Action: | Specify a valid Transaction Code in your XML data. |

| Heading | Data |
|---|---|
| Message: | public final static String transactionCodeNotSupportedText = "THE TRANSACTION CODE {0} IS NOT SUPPORTED IN THIS INTERFACE. VALID CODES ARE {1}"; |
| Occurs When: | A transactionCodeNotSupported occurs when the TransactionCode element is specified with an unsupported value. |
| Corrective Action: | Correct the XML  data to specify a valid TransactionCode. |

| Heading | Data |
|---|---|
| Message: | public final static String conflictingElementErrorText = "THE ELEMENT {0} AND THE ELEMENT {1} CANNOT BOTH BE SPECIFIED"; |
| Occurs When: | A conflicting element error occurs when two elements have been provided in a G-Log XML Element, when only one out of the two may be used. |
| Corrective Action: | Eliminate one of the two conflicting elements in your XML data. |

| Heading | Data |
|---|---|
| Message: | public final static String invalidNumberFormatErrorText = "THE NUMBER ELEMENT {0} WITH VALUE {1} IS NOT IN A NUMBER FORMAT"; |
| Occurs When: | An invalid number format error occurs when non-numeric characters are specified in a numeric element. |
| Corrective Action: | Eliminate the non-numeric characters in your XML data. |

| Heading | Data |
|---|---|
| Message: | public final static String missingElementErrorText = "WE REQUIRE A {0} ELEMENT WITH VALUE {1} IN THE {2} ELEMENT"; |
| Occurs When: | A missing element error occurs when an element with a particular value is required. |
| Corrective Action: | Provide the element with the required value as indicated in the error message. |

| Heading | Data |
|---|---|
| Message: | public final static String invalidBooleanErrorText = "THE BOOLEAN ELEMENT {0} WITH VALUE {1} MUST BE EITHER Y or N"; |
| Occurs When: | An invalid Boolean error occurs when a Boolean element is provided with a value other than Y or N. |
| Corrective Action: | Provide either Y or N in the indicated Boolean element in your XML data. |

| Heading | Data |
|---|---|
| Message: | public final static String invalidActionCodeErrorText = "THE ACTION CODE {0} is not valid. Valid codes are A, D"; |
| Occurs When: | An invalid action code error occurs when an action code is specified with a value other than A or D. |
| Corrective Action: | Provide either A or D in your XML data. |

| Heading | Data |
|---|---|
| Message: | public final static String invalidCodeErrorText = "THE ELEMENT {0} WITH VALUE {1} is not valid. Valid codes are {2}"; |
| Occurs When: | An invalidCodeError occurs when a code is specified that is not valid for that element |
| Corrective Action: | Correct the XML  data to provide a valid value. |

| Heading | Data |
|---|---|
| Message: | public final static String invalidActivityErrorText = "THE ACTIVITY {0} is not valid. Valid codes are D, P or O"; |
| Occurs When: | An invalidActivityError occurs when an activity is specified with a value other than P, D or O. |
| Corrective Action: | Correct the XML  data to provide anyone of D, P or O. |

| Heading | Data |
|---|---|
| Message: | public final static String transactionProcessorExceptionText = "CAUGHT THE FOLLOWING EXCEPTION WHILE PROCESSING TRANSACTION: {0}"; |
| Occurs When: | A transactionProcessorException occurs when the integration layer validation method has not checked for a given error condition, and the condition is caught by the underlying database. Most often this error occurs when an attempt is made to update the Oracle Transportation Management database so that one or more database referential integrity constraints are violated. For example, this error occurs if you attempt to delete a transportation order after one or more releases have been created. |
| Corrective Action: | No particular correction can be defined. Analyze the error based on the requirements of the data in the underlying database. |

| Heading | Data |
|---|---|
| Message: | public final static String maxLengthExceededErrorText = "ELEMENT {0} VALUE {1} HAS LENGTH {2} WHICH EXCEEDS THE MAX LENGTH {3} OF TABLE {4} COLUMN {5}"; |
| Occurs When: | The length of an element value exceeds the length of the corresponding database column |
| Corrective Action: | Correct the XML data to provide a value which does not exceed the maximum length. |

| Heading | Data |
|---|---|
| Message: | public final static String maxLengthExceededErrorText3 = "ELEMENT {0} VALUE {1} EXCEEDS THE MAX LENGTH {2}"; |

| Heading | Data |
|---|---|
| Occurs When: | The length of an element value exceeds the length of the corresponding database column |
| Corrective Action: | Correct the XML data to provide a value which does not exceed the maximum length. |

| Heading | Data |
|---|---|
| Message: | public final static String validateMaxLengthErrorText = "ERROR VALIDATING MAX LENGTH OF ELEMENT {0} VALUE {1} - CHECK TABLE NAME {2} COLUMN NAME {3}"; |
| Occurs When: | The specified table/column information could not be found. |
| Corrective Action: | Call Support. |

| Heading | Data |
|---|---|
| Message: | public final static String transactionSuccessText = "TRANSACTION NUMBER {0} APPLICATION {1} PRIMARY KEY {2} TRANSACTION CODE {3} SUCCESSFULLY PROCESSED"; |
| Occurs When: | A transaction success informational message occurs when a transaction has been successfully processed. |
| Corrective Action: | No action needed. |

| Heading | Data |
|---|---|
| Message: | public final static String matchMultipleShipmentErrorText = "UNABLE TO PROCESS DUE TO MULTIPLE SHIPMENTS MATCHED ON {0}"; |
| Occurs When: | A matchMultipleShipment error message occurs when ShipmentRefnums/EquipmentNumber match different Shipment_GID. This type of error happens when receiving a TenderResponse or a ShipmentStatus. |
| Corrective Action: | Correct the XML data to provide ShipmentRefnum values which correspond to the same Shipment. |

| Heading | Data |
|---|---|
| Message: | public final static String missingRequiredDataErrorText = "WE REQUIRE A {0} ELEMENT WITH A VALUE MATCHING A {1} IN THE {2} TABLE"; |
| Occurs When: | A missingRequiredDataError occurs when an element with a particular value is missing from the database table. |
| Corrective Action: | Correct the database data to provide the value as indicated in the error message. |

| Heading | Data |
|---|---|
| Message: | public final static String orderNotModifiableText = "ORDER {0} IS NOT MODIFIABLE AND SO COULD NOT BE MODIFIED OR DELETED."; |
| Occurs When: | An orderNotModifiable informational message occurs when the status on an order is not "WKFLW_ORDER_OB_MODIFIABLE". This can occur if an order is in a state that restricts it from being modified, or an agent is setup to restrict modification. |
| Corrective Action: | If you want to be able to modify the order, you may have to change the state of the order or modify the agent that handles order modifications. |

| Heading | Data |
|---|---|
| Message: | public final static String reDoTransmissionErrorText = "UNABLE TO raiseNewXMLTopicsForRedoTransmissions, STACK TRACE: {0}"; |
| Occurs When: | |
| Corrective Action: | None. |

| Heading | Data |
|---|---|
| Message: | public final static String savedQueryNoDataFoundErrorText = "THE SAVED QUERY {0} RETURNS NO DATA"; |
| Occurs When: | The saved query in the SShipUnit element did not return any values. |
| Corrective Action: | Verify that the integration saved queries are correct and that the desired shipunits exist. |

# 22. Appendix B – Backward Compatible Dates – pre 6.4.2

In version 5.5, all date/time XML elements were updated to include the time zone and time zone offset. The following section describes the logic retained for backward compatibility of subsequent versions. This conversion only applies to XML messages using the pre-6.4.2 namespace URL "`http://xmlns.oracle.com/apps/otm`".

The previous XML Date elements as strings are replaced with new elements to include GLogDate, TimeZone ID and TimeZone Offset. These new date elements are renamed with a suffix "Dt" for uniformity. Also, the original XML Date elements with GLogDate are modified to include TimeZone ID and TimeZone Offset.

For example:

| **Old XML Element** | **New XML Element** |
|---|---|
| <StartDate>20071012173600</StartDate> | <StartDt> <GLogDate>20071012173600</GLogDate> <TZId>America/New_York</TZId> <TZOffset>-4</TZOffset> </StartDt> |
| < ActivityDate> <GLogDate>20071012173600</GLogDate> < /ActivityDate> | < ActivityDate> <GLogDate>20071012173600</GLogDate> <TZId>America/New_York</TZId> <TZOffset>-4</TZOffset> < /ActivityDate> |

For inbound integration the old date elements are compatible and remain same, Oracle Transportation Management will convert and persist them. Backward compatibility logic is used to support inbound transmissions by enabling the property `glog.integration.enableTimeZoneCompatibility=true`.

Properties are used to specify the compatibility of old date elements to new date elements as `glog.integration.timeZone.{old element name}={new element name}`. For example: `glog.integration.timeZone.StartDate = StartDt`

XSL Transformation Files can be used for mapping elements if needed, such as `(GLogXML_v55_to_v60_DateTime.xsl, GLogXML_v60_to_v55_DateTime.xsl)`

# 23. Appendix C – Uploading XSL Stylesheets

Using XSL Stylesheet files for inbound or outbound XML transformation is deprecated as of version 6.4.2 and will be removed in a future version. Any current use should be converted to use Stylesheet Content or Stylesheet Profile that refers to Stylesheet Content, as appropriate. For inbound transformation via HTTP see Transform Inbound XML with XSL in chapter 9. For outbound transformation via all transport protocols see Transform Outbound XML with XSL in chapter 7.

The following process captures the steps required to upload an XSL file.

1. Sign in using a using with DBA.ADMIN privileges.
2. Go to Business Process Automation, Integration, Integration Manager, Upload an XML/CSV Transmission
3. Browse to select XSL file (must have ".xsl" file extension)
4. Upload
5. Ensure response shows file has been uploaded to correct directories in both the web and application servers.

# 24. Appendix D – Interface Additional Detail

## Actual Shipments

If the service provider sends a new order release as part of the actual shipment, Oracle Transportation Management creates an order release and order base for the new release.

An actual shipment is required to print shipment documentation such as a bill of lading or Domestic Packing List.

### Actual Shipment Workflow Considerations

Changes to an existing shipment using the Actual Shipment interface can be configured to trigger additional business logic by way of agent workflow. When the shipment contains multiple orders or is one of a 'graph' of multiple shipments for different modes or service providers, this workflow can be quite complex and involve modification of other objects. Consequently, although simple modifications **CAN** be achieved using the Actual Shipment interface, e.g. addition of a new Shipment Reference Number, these cases should instead make use of the Generic Status Update interface.

### Structural Changes to Shipment Ship Units in Release 6.0

Prior to Oracle Transportation Management release 6.0, shipment ship units were shared among different shipments/legs of a multi-leg movement. With the multi-tier execution enhancement in release 6.0, this has been changed to not allow sharing of the shipment ship units across multiple shipments. As a result of this change, updates to Shipment Ship Unit details (quantities or attributes) will not automatically be reflected across all legs when a change is made to a shipment ship unit. There are several options for making the appropriate changes to each of the impacted legs:

- Update Each Shipment Manually
    - Update each of the shipments impacted with a separate ActualShipment xml. Each ActualShipment xml would update the appropriate ship units on that shipment.
- Leverage Integration Saved Query
    - Leverage the IntSavedQuery within the Shipment/ShipmentHeader and Shipment/SShipUnit to search for the GIDs of the shipment(s) and ship unit(s).
    - This option is beneficial when the shipment and/or ship unit GIDs are not all known in the integration layer.
    - This option is limited in that any other fields set in the ActualShipment would be applied to all shipments identified by the query. A similar situation exists for the Ship Units on the shipments.
- Use Propagation for Ship Unit Changes
    - This is useful to propagate the count and quantity changes to upstream or downstream shipments.
    - Initiated by setting the `Shipment/ShipmentHeader/IntCommand/IntCommandName = "PropagateShipUnitChanges"`. The propagation option (upstream, downstream, or both) is specified by the IntCommand/IntArg element. Refer to the GLog XML Schema for the specific values to be used.
    - This option has limitations on the types of changes it will handle and propagate. Several are listed as follows:
        - Cannot Add SEquipment From Another Shipment: You cannot add a SEquipment that already exists on another shipment to a shipment with the propagation. The SEquipment addition should be done in a separate ActualShipment without the propagation.

- Cannot Change SEquipment for Existing Ship Unit: If there are more than one SEquipment on a Shipment, you cannot change the SEquipment for an existing Ship Unit with the propagation.
- Cannot Add New SEquipment for Existing Ship Unit:  You cannot add a new SEquipment and assign to an existing Ship Unit with the propagation. The SEquipment changes should be done in a separate ActualShipment without the propagation.
- Limited Support to Add New SEquipment and New Ship Unit: You can add a new SEquipment and assign to a new Ship Unit. The logic will determine the new SEquipment and persist it before initiating the propagation logic. The ability to support adding the new SEquipment is controlled by the following property: glog.integration.shipment.persistNewSEquipmentForShipmentActualAPI=true
- Can Add a New Ship Unit:  The propagation supports adding a new Ship Unit to the Shipment. The Ship Unit must be assigned to an existing SEquipment on the shipment.
- Cannot Use DR Transaction Code In Ship Unit: You cannot use the DR transaction code in the Ship Unit to dereference the Ship Unit from the Shipment. Since the Ship Units are no longer shared among shipments, you should consider using the D transaction code to delete the ship unit. If there is a need to use the DR transaction code, it should be done in a separate ActualShipment without the propagation.

**Note**: When migrating to Oracle Transportation Management 6.0 from a previous release, review the implementation of the Actual Shipment integration into Oracle Transportation Management to determine the impacts of this change.

### Updating Parts of a Shipment

When sending an actual shipment to Oracle Transportation Management you often want to update parts of an existing shipment. Generally the TransactionCode of the shipment (ShipmentHeader/TransactionCode) provides the guiding rule for the child elements. Here are some examples:

| **Element** | **Description** |
|---|---|
| Shipment/ShipUnit/ShipUnitContent | If TransactionCode=IU, a LineNumber that does not exist will be added, otherwise updated.<br><br>Currently, you cannot delete an individual line. |
| Shipment/ShipmentHeader/ShipmentRefnum | If TransactionCode=IU, a new QualifierValue pair will be added.<br><br>You can delete (and replace) using the GenericStatusUpdate interface. |

| Element | Description |
|---|---|
| ShipmentHeader/Remark | If TransactionCode is IU and the RemarkSequence does not exist, then Oracle Transportation Management will automatically generate a sequence number and add the remark.<br><br>If TransactionCode is IU and the RemarkSequence does exist, Oracle Transportation Management updates with a new RemarkQualifier and RemarkText.<br><br>If you supply neither a RemarkSequence nor a RemarkQualifier, Oracle Transportation Management adds the RemarkText as new Remark.<br><br>You can delete (and replace) using the GenericStatusUpdate interface. |

In the Shipment element, if you set the transaction code to RC and set the ReplaceChildren/ManagedChild element to "ShipmentStop", Oracle Transportation Management deletes all shipment stops for that shipment and replaces the deleted shipment stops with the shipment stops from your transmission.

**Note:** This does not apply to shipment stops marked IsPermanent (same as Permanent check box in Oracle Transportation Management web interface).

In the Shipment element, if you set the transaction code to RC and set the ReplaceChildren/ManagedChild element to ShipmentStopDetail, Oracle Transportation Management replaces the existing ship units with the ship units in your transmission.

**Note:** This does not apply to existing ShipmentStopDetails marked IsPermanent.

**Note:** Within ShipmentStopDetail, the removal of the reference to the ShipUnitGID(s) will not remove the S_Ship_Unit from the system. Only the reference to the object is removed.

In the Shipment element, if you set the transaction code to U and the Shipment ID is missing from either the transaction or the database, you will receive an error.

When a new shipment referencing a ship unit is added with missing ship unit data, then the ship unit data is pulled from the database. When a new shipment referencing a ship unit is added with new data, then the ship unit data passed in through integration is used.

### Adding Stops

There is no way to insert a new stop to a shipment via shipment actuals unless the new stop has a stop number that does not already exist on the shipment (like adding stop #3 to a 2-stop shipment, or adding stop #2 to a shipment with stops 1 and 99).

### Adding Ship Units

An added ship unit should be linked to an order release that is on a shipment (this order release must be planned on the initial shipment) and should be linked to the initial pickup location. If the check box on the Shipment Header indicates “Propagate Updates,” the Oracle Transportation Management integration layer will call business logic to add the new ship unit to subsequent stop on the initial shipment and all affected succeeding shipments.

To add a ship unit to a shipment, the following must be done in the ActualShipment XML interface:

1. Specify a flag to indicate that new ship unit should be applied to downstream shipments.

   ```
   AcutalShipment.Shipment.ShipmentHeader.IntCommand.IntCommandName =
   "PropagateShipUnitChanges"
   ```

2. Indicate that DropOff stop should be determined for the shipment, but not to propagate the ship unit changes.

   ```
   ActualShipment.ShipmentHeader.IntCommand.IntCommandName =
   "DetermineShipUnitDropoff"
   ```

3. Specify the new ship unit.

   ```
   ActualShipment.ShipmentHeader.Shipment.ShipUnit.ShipUnitGID =
   ```

4. Specify the Transaction Code (optional).

   ```
   ActualShipment.ShipmentHeader.Shipment.ShipUnit.TransactionCode = "I" or
   "IU"
   ```

5. Specify the pickup stop for the ship unit.

   ```
   ActualShipment.ShipmentHeader.Shipment.ShipmentStop.ShipmentStopDetail.Activ
   ity = "P"
   ActualShipment.ShipmentHeader.Shipment.ShipmentStop.ShipmentStopDetail.ShipU
   nitGID = ShipUnit.ShipUnitGID
   ```

6. ShipmentFrom and ShipmentTo Locations can in the Shipment.ShipUnit element are ignored. They will be based on the release.
7. Assign an SEquipment for the new ShipUnit via one of the following options:
8. Using the ShipUnit.SequipmentGID element
9. Allow integration to assign it by using ActualShipment.ShipmentHeader.Shipment.ShipUnit.SEquipmentGIDQuery.SequipGIDMatchOption = "Any"
10. Query the SEquipmentGID using the ActualShipment.ShipmentHeader.Shipment.ShipUnit.SEquipmentGIDQuery.IntSavedQuery
11. Have the business logic assign it by not specifying the element in the ship unit.

**Updating Ship Units**

The updating of ship units means packaged items can be deleted or added to those ship units. Quantities from existing items can also be changed.

The following options are available via integration:

In the SShipUnit XML interface:

```
SShipUnit.TransactionCode = "RC"
SShipUnit.ReplaceChildren.ManagedChild = "ShipUnitContent"
```

In the ActualShipment XML interface:

```
Shipment.ShipmentHeader.TransactionCode = "RC"
Shipment.ShipmentHeader.ReplaceChildren.ManagedChild = "ShipUnitContent"
```

In the Shipment.ShipUnit.SShipUnit XML interface:

```
ActualShipment.Shipment.ShipUnit or ActualShipment.Shipment.SShipUnit
SShipUnit adds the ability to query for the ShipUnitGID if it’s not known
```

### Deleting Ship Units

Deleting a ship unit only removes the link between the shipment stop and the ship unit, as well as the link between the ship unit and equipment. The actual ship unit will not be deleted from the database. integration will also attach the ship unit remark, “Ship Unit Not Picked Up” to the ship unit.

A ship unit can be marked for removal from the shipment via the TransactionCode as follows:

```
ActualShipment.Shipment.ShipUnit.TransactionCode = “DR”
```

where “DR” corresponds to “Delete Reference.” The ship unit will be removed from the shipment, but not deleted from Oracle Transportation Management.

Alternatively, you can delete ship units from a shipment using the IntCommand via integration. You can either delete all the ship units from the shipment, or only those that are marked as non-permanent. When used, the ship unit record, its shipment stop detail record, and any corresponding equipment, is deleted.

Specify the integration command as follows:

To remove all ship units:

```
<IntCommand>
   <IntCommandName>RemoveAllShipUnits</IntCommandName>
</IntCommand>
```

To remove only non-permanent ship units (where IsPermanent = 'N'):

```
<IntCommand>
   <IntCommandName>RemoveNonPermanentShipUnits</IntCommandName>
</IntCommand>
```

To remove orphaned ship units, use the command below. This specifies that the ship units that have been removed from the shipment via the DR transaction code should be deleted if no other shipments refer to them. Without this command, those ship units are left in the system and can later be added to other shipments.

```
<IntCommand>
   <IntCommandName>DeleteOrphanedShipUnits</IntCommandName>
</IntCommand>
```

### Alternative Interfaces For Updating Ship Units

For alternatives to using this interface to update ship unit information, see SShipUnit and TransOrder.

## Tips For Shipments As Work or SAWs

| Element | Description |
|---|---|
| ShipmentHeader2 | In Shipment/ShipmentHeader2, the most important element is ShipmentAsWork and it should almost always be set to "Y". The exception is when:<br><br>1) The shipment is new<br>AND<br>2) There is at least one release associated with<br>the shipment<br>OR<br>the shipment.ShipmentHeader2.<br>AutoGenerateRelease = "Y"<br><br>To avoid confusion, set the Perspective element to "B." If Perspective is not specified, it will default to "B". All other elements in this branch are completely optional from a schema and business perspective. |
| ShipmentHeader, use correct rate | In the Shipment/ShipmentHeader, if you can get them, it is good to provide the RateOfferingGID and the RateRecordGID. This helps Oracle Transportation Management use the rate for the service provider that actually took the load. If you can't get information for these elements, the ServiceProviderGID would be the next best thing to use. |
| sPlannedTimeFixed | If you want to send in old shipments (past dates) and want Oracle Transportation Management to rate with correct rates (pertaining to correct effective/expiration dates), then you may want to use the Shipment/ShipmentStop/ArrivalTime/EventTime/IsPlannedTimeFixed flag set to "Y". You only need to insert the ArrivalTime element at the first stop (only) and that the date here should be the same as the StartDate of the shipment. |
| SEquipment | Oracle Transportation Management requires at least one SEquipment object. When you insert a new SAW, Oracle Transportation Management creates a default SEquipment if you do not provide one. If there are several ship units, the same (created equipment) is specified for each ship unit. The only thing really required in SEquipment is the SEquipmentGID. To avoid problems later, include the SEquipment element and set the SEquipmentGID to the same value as the ShipmentGID. This makes it easier to identify and manage the SEquipment, if there is ever a need in the future to specify multi-equipment, or update the shipment with additional ship unit information.<br><br>**Note:** When you update a SAW with a new ship unit, you must include the SEquipmentGID. Oracle Transportation Management cannot create one for you. |

| Element | Description |
|---|---|
| TransOrder | The Shipment/TransOrder element is outbound only so you cannot include it. Any TransOrderHeader info should be specified in the Release/TransOrderHeader element. |
| Locations | When specifying source and destination locations in the Shipment/Release/ShipFromLocationRef and Shipment/Release/ShipToLocationRef elements, refer to locations already defined in Oracle Transportation Management instead of defining new ones (using Shipment/Release/ShipFromLocationRef/LocationRef/Location). This saves you the effort of providing all the Location elements. To refer to existing locations, use the Shipment/Release/ShipFromLocationRef/LocationRef/LocationGID element. |
| ShipUnits | Set ShipmentHeader2/AutoGenerateRelease to "Y" to avoid having to populate both Shipment/Release/ShipUnit and Shipment/ShipUnit.<br><br>**Note:** Oracle Transportation Management generates an error if AutoGenerateRelease is set to "Y" and you still include the Shipment/Release element. |

## *Order-Centric Modifications*

Most modifications via this interface are based around shipment ship units (SShipUnit). In these cases, all weight, volume, quantities, and rating are based on shipment ship units. If you would like to modify shipments based on order information, you can do so by using the following sub-elements in the ShipmentHeader element:

- ShipmentModViaOrderLine
- ShipmentModViaOrderSU

When these two elements are used in the ShipmentHeader, the following logic will be used instead of the standard Shipment Interface logic:

1. Oracle Transportation Management will only interact with the order line level or order ship unit information instead of the shipment ship unit level information.
2. The logic addressing shipment modifications will change the number of order ship units involved and allocate the delta in the ship unit counts across multiple ship units.
3. The modified order ship unit count will be properly propagated and the related business objects (shipments and order movements) will be updated across legs.
4. The modified gross weight and volume will be updated per ship unit. This would then be reflected in the shipment total gross weight and volume, which impacts the shipment cost. This should only be applied when the AffectsCurrentLegOnly element is set to 'N'.

Both of these elements will only be included once on the shipment. There is no need to repeat this data for both the pickup stop and the delivery stop. Since the Shipment Interface is defined the same on the inbound and the outbound, you can only specify one way for the modification to happen, either at the order line level or the ship unit level.

### ShipmentModViaOrderLine

The ShipmentModViaOrderLine element will contain all of the counts, weights, and volumes for that order (order release, order release line, or order base) that is being shipped on this shipment across all the shipment ship units.

#### ShipmentModViaOrderSU

The ShipmentModViaOrderSU element, Oracle Transportation Management will loop through all the ship units that are on the shipment that have the same order ship unit GID (ob_ship_unit_gid or or_ship_unit_gid).

The inbound XML will accept this data into Oracle Transportation Management when you are doing a modify transaction. When the integration brings in this modification it will call business logic that will apply allocation rules and perform the appropriate updates.

### Data Requirements

To send shipments and perform planning actions, you must make decisions about the way you want Oracle Transportation Management to perform certain actions.

#### Sending Shipments (Shipment as Work)

Send shipments that do not have orders associated with them to Oracle Transportation Management for processing using the Shipment interface. This type of shipment is known as a shipment as work or manual shipment; it can include order level information, but not necessarily. These shipments are not bundled, re-consolidated, or re-sequenced.

A shipment as work must have at least one pickup and one delivery location. A shipment as work is not associated with an itinerary.

> **Note:** To indicate that the shipment you are sending to Oracle Transportation Management is a shipment as work, enter Y in the ShipmentAsWork element.

When a Shipment as Work is received, Oracle Transportation Management can be set to automatically perform certain actions defined in public workflow agents in the Agent Manager.

To ensure best possible performance, you should let Oracle Transportation Management process your actual shipments in parallel. To do this, either send only one actual shipment per transmission or follow these steps:

- In the TransmissionHeader, set IsProcessInSequence to N.
- Send all the actual shipments in one Transmission.

See the Shipment Manager help for a detailed description of manual shipments.

If you insert a new shipment and omit the end_date, Oracle Transportation Management sets the end_date to the same date as the start_date.

## Accrual Interface

When the ALLOCATION GENERATES ACCRUALS parameter is set to TRUE and the shipment status is ACCRUAL_ALLOWED, the allocation logic will generate an accrual record. The accrual record contains the difference between the current allocated freight cost and the previously transmitted freight cost. The delta between the two is used because a single order may be on multiple shipments, which are approved for payment in different time periods. These accrual records are sent as part of this interface.

## Invoice Interface

Invoices can be automatically matched to shipments based on the Service Provider ID and Shipment Reference Number fields. If more than one shipment is found for an invoice, the invoice must be reviewed manually and assigned to a shipment, it must be rejected. After an invoice is approved, a voucher gets created. A voucher represents what the planner agrees to pay for the shipment.

Use the Financials managers to create and modify invoices and customer bills.

In some cases, you may need to send the Invoice interface outbound. This is true when sending a bill to yourself for internal invoice or billing purposes. When the Invoice interface is used outbound, then it will include all shipment details, as well as any associated order information.

### Consolidated Invoices

1. You must send each invoice, parent and child, as a separate transaction.
2. Parent invoices must enter Oracle Transportation Management before any child invoices.
3. Child invoices may be sent inbound referencing a parent in one of two ways:

- Populate the invoice number on the child to be that of the parent and integration will lookup the parent id based on the invoice number.
- Populate the parent invoice ID on the child.

## Job Interface

A job offers a workspace that brings together the objects and activities required of them, including:

- The ability to group all objects related to a job and perform existing functions/actions against those objects, including buy shipments, sell shipments, non-freight related charges, and customer bills.
- The ability to manage jobs from various perspectives depending on responsibility. For example, export, import, both, or consolidations.
- The ability to manage settlement functions at the job level, including profitability, expenses, revenues, and billing.

**Note: The interface is supported on the outbound only.**

The primary business objects in the Job Interface are:

1. Order releases
   - For each order release related to the job, JOB_ORDER_RELEASE_JOIN will be added to the XML. Although orders are not required to create a job, at least one order should be related to the job to send out the interface.
2. Buy side costs
   - The Buy Side Costs wrapper element contains two sub elements, Buy Shipments and Buy Allocation.
   - Buy shipments: Select all related orders. For each order, select all related buy shipments where the shipment job GID equals null or it equals the current job GID. There may be zero or more buy shipments.
   - Buy allocation: Each allocation will be selected for each order, where the allocation Shipment Job GID is equal to the current job or it is equal to null. There may be zero or more allocations.
3. Sell side costs
   - The sell side costs works exactly the same as the buy side, except the selection criteria is based on sell side perspective.
4. Bills
   - Each customer bill related to the job will be included in the XML. Bills can be found in the JOB_BILL table. Zero or more bills are required.

- Because this interface can potentially be large, redundant data has been reduced across multiple data elements included in the interface. This includes:
- The ability to only include the Order Release GID in the shipment, allocation, and bill elements.
- The ability to only include the Shipment GID in the allocation and bill elements.

## Release Interface

An order release contains the following information:

- Order release ID that is automatically generated by Oracle Transportation Management.
- Order release name and type.
- Order base ID that references the base order from which the order release was created.
- Source and destination locations.
- Early/late pick up dates.
- Assigned or fixed itinerary.
- Current status.
- Package or non-package data attributes.

#### Release Method

Populate the ReleaseMethodGid to tell Oracle Transportation Management how to release ship units. The  ReleaseMethodGid (also called Order Configuration) tells Oracle Transportation Management how to build ship units from order lines, or how to calculate ship unit information if ship units are entered. If ReleaseHeader/ReleaseMethodGid is not populated, the default can be property controlled. See the Order Management Guide for more information.

#### Business Number Generator (BNG)

You can send a transportation order to Oracle Transportation Management, without the entering values in the TransOrder GID, Ship Unit ID, or Order Release ID elements in the XML Transmission. Oracle Transportation Management generates values for these fields based on the default business number rule in place when the order comes into the system. You can set up the BNG to create numbers that fit your needs.

## ShipmentStatus Interface (INE)

You can also send received ship unit quantities with this interface using the SStatusShipUnit element.

### *Insert New Shipment Status into Oracle Transportation Management*

#### Required Data

Before you can send ShipmentStatus transmissions to Oracle Transportation Management, you must set up the following:

- User accounts for service providers in Oracle Transportation Management
- Shipment Status Codes
- Shipment Status Reason Codes
- Shipment Event Groups
- Shipment Reason Groups
- Corporations for service providers

#### What Data Goes into the Transmission?

1. To ensure that Oracle Transportation Management processes multiple ShipmentStatus transactions in the order you intend, set IsProcessInSequence to Y in the TransmissionHeader.
2. Identify which object (shipment, shipment group etc) the status applies to. Set StatusLevel, ShipmentStatusType, ServiceProviderAlias, and ShipmentRefnum or IntSavedQuery.
3. ShipmentStatusType must be set to one of Shipment or ShipmentGroup. Note that it is case sensitive. The integration logic assumes that it is a ShipmentGroup if the value is not matched to Shipment.
4. Optionally, identify which equipment the shipment status refers to with SStatusSEquipment element.
5. Include the time when the event occurred. To be sure that Oracle Transportation Management can interpret the time correctly, include the TimeZoneGID element. Alternatives to doing this is:
   - If you cannot include the TimeZoneGID, Oracle Transportation Management can set the time zone to the time zone of the Location where the event occurred.
   - If you cannot do this either, set the TimeZoneGID to Local. In this case, Oracle Transportation Management saves and displays the event date as entered, ignoring user preferences.
6. Enter your status information. In some cases, shipments can only have events added to them if they are of a certain status.
7. Identify at what SSStop (number or location name) the event (shipment status) occurred. The LocationID = Location Reference Number and the LocationRefnumQualifierGID = Location Reference Qualifier in the Location Manager.
8. If you have a Shipment Agent Type with the Recalc Estimated Stop Times action, then you must include a RATE_GEO element for Oracle Transportation Management to be able to recalculate your estimated stop times and/or re-drive your shipment. If you omit the RATE_GEO element, Oracle Transportation Management only resets the stop times you provide.
9. See the ShipmentStatus Diagram in the XML schema to learn which elements are required. To view the diagrams use an XML application like XML Spy.

### *Send Shipment Status from Oracle Transportation Management*

You can forward a received ShipmentStatus transmission to an external system with an agent. See the agent action called SEND SHIPMENT STATUS XML.

### *Match Events to an Object*

You can use one of these methods:

- If you need the object (shipment or shipment group) to match many reference numbers, use IntSavedQuery.
- If you need the object to only match one out of a set of reference numbers, use ShipmentRefnums, or ShipmentGroupRefnums.

#### IntSavedQuery

If you specify the IntSavedQuery element, only that query is applied. You can define a query to search for shipments or shipment groups that shipment status applies to. To do this, set:

- IntSavedQueryGID to specify which query you want to use. If the query you specify here does not return any results, Oracle Transportation Management generates an error message. No other queries are applied. You must have created this query in Power Data beforehand.
- IntSavedQueryArg to arguments that can be referred to in the queries. For example, BM=YELLOW-0000007. If you omit this element, your IntSavedQueryGID must point to a query that uses XPath instead.
- IsMultiMatch to N to forbid multiple records to be returned from the query. If your query happens to return multiple records, Oracle Transportation Management generates an error message.

**Refnums**

If you omit the IntSavedQuery element, Oracle Transportation Management tries to match your shipment status with:

- The ShipmentRefnum elements to the shipment_refnum table in the database.
- The SSEquipment/ EquipmentIdentificationNum element to the S_Equipment.Equipment_Number field in the database
- The standard integration saved query INT_SHIPMENT_STATUS_GID_1
- The standard integration saved query INT_SHIPMENT_STATUS_GID_2

You can optionally enforce a rule that a give shipment may have only one shipment reference number with a given qualifier. The update_flag column in the shipment_refnum_qual table indicates if the rule is in effect or not. The valid values for the update_flag are:

- UPDATE_OK: Only one value is allowed for a give qualifier, the value of which can be modified.
- UPDATE_NOT_OK: Only one value is allowed for a give qualifier, the value of which cannot be modified.
- MANY: a given shipment can have multiple values for the same qualifier.

### *Match Events to a Shipment Stop*

For Oracle Transportation Management to match an event to a stop on a shipment, you must include the SSStopSequenceNum element.

Another way of matching event to shipment stop is to include the LocationID where the event occurred and the LocationRefnumQualifierGID in SSStop/SSLocation. This only works if you have enabled this feature in your glog.properties file. As long as Oracle Transportation Management can match your LocationID to a stop number, your shipment status saves as if you had supplied a stop number.

> **Note:** If Oracle Transportation Management cannot match the event to a location, Oracle Transportation Management still saves the information but not for a specific stop and only as informational. You can also have Oracle Transportation Management send you a TransmissionReport if the LocationID is missing altogether (controlled by glog.properties). Oracle Transportation Management also set the time zone for the event to local.

Correspondingly, if the event is not related to a shipment stop to begin with, Oracle Transportation Management saves the event as informational with a local time zone.

A single stop related shipment event can be applied to multiple shipments, regardless of whether their stop numbers or location IDs are the same. This will allow for situations where you want to apply a single shipment stop event to stop 2, but stop 2 of shipment 1 and stop 1 of shipment 2 are both Philadelphia. Stop related events are applied to all the shipments specified in the ShipmentStatus interface. To work successfully, the ShipmentStatus XML must include an IntSavedQuery element that will return two shipments. Logically, this is similar to having specified the ShipmentStatus message multiple times in the Transmission XML.

## TransOrder Interface (INO)

Create, modify, or delete order information through the TransOrder interface.

### *Insert New Order and Release Order Line*

This procedure shows you how to:

- insert a new order
- release all or part of an order line
- build shipments from the order release

#### Required Data

To send an order to Oracle Transportation Management, certain information related to the order must already exist in Oracle Transportation Management. For example, you must have a valid itinerary, rate, locations, and so on.

#### Setup

You control validation of incoming transmissions with the glog.integration.validation property.

#### What Data Goes into the Transmission?

1. Set the TransactionCode to I. A transaction code of UI or IU works too.
2. If you do not want to enter values for the TransOrderGID, TransOrderLineGID, ShipUnitGID, or OrderReleaseGID elements, you can have Oracle Transportation Management automatically generate GIDs. Automatic generation of GIDs only works for a transaction code of I.

   **Note**: If a transaction code of IU is used, then a TransOrderGID must be provided.
3. Populate the ProcessingCodeGID to tell Oracle Transportation Management how to plan the shipments from the order release.
4. Populate the TransOrderLineDetail element, including the PackagedItemCount, WeightVolume/Weight, and WeightVolume/Volume under TransOrderLineDetail/TransOrderLine/ItemQuantity/ to specify the order lines. You can set all but one of them to 0, if your setup uses the same kind of quantity to release.
5. Set TransOrderLineDetail/TransOrderLine/ItemQuantity/IsShippable = N.
6. Populate the amount to release in the TransOrderHeader/ReleaseInstruction/QuantityToRelease element.

   If you omit the ReleaseInstruction, set TransOrderLineDetail/TransOrderLine/ItemQuantity/IsShippable = Y to have Oracle Transportation Management create an order release for all of your order lines.
7. If your order base is coded in a format other than GLogXML you need to transform your TransOrder transmission into the GLogXML schema, you can use Oracle Transportation Management's transform feature to do this.
8. If you want to change the level of validation for this transmission, you can include a processing instruction to set the desired level. By default, TransOrder validation is turned on.
9. See the online help for a description of the fields.
10. See the TransOrder Diagram in the XML schema to learn which elements are required. To view the diagrams use an XML application like XML Spy.

#### Transmission Results

11. Oracle Transportation Management receives your transmission and starts to process it internally.
12. Oracle Transportation Management starts the public Order Base - Insert agent.

    If the current date is outside the effective date/expiration date window of your TransOrder, the agent cannot create order releases. You must release the TransOrder via the process manager. There you can release all orders which have release instructions, but whose release has not been processed.

    If you use the UI or IU transaction codes and the record exists already, Oracle Transportation Management starts the public Order Base - Modify agent instead.
13. Oracle Transportation Management raises events that in turn can trigger Notifications to be sent.

#### Error Messages

If you receive a TransmissionReport, check for integration messages.

### *Insert New TransOrder and Release ShipUnit*

Auto-releasing ship units is consistent with order base lines. This procedure shows you how to:

- insert a new order
- release all or some ship units on the order base
- build shipments from the order release

#### Required Data

To send an order to Oracle Transportation Management, certain information related to the order must already exist in Oracle Transportation Management. For example, you must have a valid itinerary, rate, locations, and so on.

#### Setup

You control validation of incoming transmissions with the glog.integration.validation property.

#### What Data Goes into the Transmission?

1. Set the TransactionCode to I. A transaction code of UI or IU works too.
2. If you do not want to enter values for the TransOrderGID, ShipUnitGID, or OrderReleaseGID elements, you can have Oracle Transportation Management automatically generate GIDs. Automatic generation of GIDs only works for a transaction code of I.

   **Note:** If a transaction code of IU is used, then a TransOrderGID must be provided.

3. Populate the ProcessingCodeGID to tell Oracle Transportation Management how to plan the shipments from the order release.
4. Populate the TransOrder/ShipUnitDetail element.
5. To be able to track your ship unites as they propagate through Oracle Transportation Management as order release ship units and shipment ship units, you might want to include a unique ID in the ShipUnitDetail/ShipUnit/ShipUnitContent/ItemQuantity/ItemTag1 element. Also, there is a TransOrderShipUnitGID element in Release/ShipUnit that can help you track ship units.

6. Set ShipUnitDetail/ShipUnit/IsShippable to Y to have Oracle Transportation Management create an order release for all your order base ship units.
7. If you omit the IsShippable element or set it to N, you need to populate the amount to release in the TransOrderHeader/ReleaseInstruction/QuantityToRelease element. With this option, you can specify the number of ship units to be released in the ReleaseInstruction/ShipUnitReleaseCount element.
8. You can override all dates and locations from the ShipUnitDetail with other settings in the ReleaseInstruction.
9. If you want to change the level of validation for this transmission, you can include a processing instruction to set the desired level. By default, TransOrder validation is turned on.
10. See the Order Base Manager online help for a description of the fields.
11. See the TransOrder Diagram in the XML schema to learn which elements are required. To view the diagrams use an XML application like XML Spy.

#### Transmission Results

1. Oracle Transportation Management receives your transmission and starts to process it internally.
2. Oracle Transportation Management starts the public Order Base - Insert agent.
3. It finds the unprocessed release instructions with a release date <= the current date.
4. If the current date is outside the effective date/expiration date window of your TransOrder, the agent cannot create order releases. You must release the TransOrder via the process manager. There you can release all orders which have release instructions, but whose release has not been processed.
5. If you use the UI or IU transaction codes and the record exists already, Oracle Transportation Management starts the public Order Base - Modify agent instead.
6. Oracle Transportation Management raises events that in turn can trigger notifications to be sent.

### *Modify Order Base With Lines*

In this scenario, you can just update the information in an order base or you can update and release the full amount specified for the TransOrderLine.

#### Required Data

#### Setup

You control validation of incoming transmissions with the glog.integration.validation.orderinterface property.

#### What Data Goes into the Transmission?

1. Set the TransactionCode to U.

   A transaction code of UI or IU works too.
2. If your update is to delete only a couple of fields in the OrderBase, use the Value to Null Field symbol.

3. If you want Oracle Transportation Management to release all your order lines, set IsShippable = Y.

   With IsShippable=Y, you should omit the ReleaseInstruction element, Oracle Transportation Management releases the full weight, volume, or count (depending on glog.properties). If you set IsShippable=Y and include a ReleaseInstruction, Oracle Transportation Management releases your order twice. One full order release based on the parameter in glog.properties and another order release based on the ReleaseInstruction element.
4. If you want to change the level of validation for this transmission, you can include a processing instruction to set the desired level. By default, TransOrder validation is turned on.
5. See the Order Base Manager online help for a description of the fields.
6. See the TransOrder Diagram in the XML schema to learn which elements are required. To view the diagrams use an XML application like XML Spy.
7. To update date fields with NULL values, submit a value of '~' in the date element(s) of the inbound TransOrder XML.

**Transmission Results**

1. Oracle Transportation Management receives your transmission and starts to process it internally.
2. Oracle Transportation Management starts the public Order Base - Modify agent.

   If you use the UI or IU transaction codes and the record does not exist already, Oracle Transportation Management starts the public Order Base - Insert agent instead.
3. Oracle Transportation Management raises events that in turn can trigger Notifications to be sent.

**Error Messages**

If you receive a TransmissionReport, check for integration messages.

### *Modify ShipUnits*

There are three ways to update the ship unit information (quantities, weights, volumes, etc) on a shipment via integration:

- Use ActualShipment. This interface provides complete control of all the fields in the Shipment.
- Use SShipUnit.
- Send another TransOrder with the IsUpdateShipmentOnly element.

The TransOrder interface together with the IsUpdateShipmentOnly element supports uploading a slightly modified TransOrder and has it update only the Shipment/SShipUnit. IsUpdateShipmentOnly indicates that the TransOrder should update the shipment only, and not the order base information.

> **Note:** To update date fields with NULL values, submit a value of '~' in the date element(s) of the inbound TransOrder XML.

Using the IsUpdateShipmentOnly element can help you reduce the need to implement a separate SShipUnit or ActualShipment interface. The use of this flag with the TransOrder interface is restricted as follows:

- The original order base should have been created using the ShipUnitDetail (not the TransOrderLineDetail).
- The information you can update is restricted to the SShipUnit element. The TransOrderHeader is ignored, and none of the other Shipment related information is updated.

- The specific S_Ship_Unit(s) to be modified are identified by using the ShipUnitGID in the new TransOrder and searching for the related Release/ShipUnit (via the OB_SHIP_UNIT_GID on SHIP_UNIT table) and then the  Shipment.ShipUnit(s) (via the SHIP_UNIT_GID field in the S_SHIP_UNIT table). The search requires those reference pointers to exist.

## *Delete Orders*

### Required Data

### Setup

You control validation of incoming transmissions with the glog.integration.validation.orderinterface property.

### What Data Goes into the Transmission?

1. Set the TransactionCode to D.
2. If you do not know the GID of the record you want to delete, you can use integration saved queries instead.
3. If you want to change the level of validation for this transmission, you can include a processing instruction to set the desired level. By default, TransOrder validation is turned on.
4. See the Order Base Manager for a description of the fields.
5. See the TransOrder Diagram in the XML schema to learn which elements are required. To view the diagrams use an XML application like XML Spy.

### Transmission Results

1. Oracle Transportation Management receives your transmission and starts to process it internally.
2. Depending on what kind of record you are deleting an agent might start. For example, if you are deleting an Order Base, the public Order Base - Delete agent starts.
3. Oracle Transportation Management raises events that in turn can trigger notifications to be sent.

### Error Messages

You cannot delete an order that is assigned to a shipment after a service provider accepts a tender on the shipment. If you try to do this, Oracle Transportation Management might send a TransmissionReport stating the problem.

If you receive a TransmissionReport, check for integration messages.

## *Bulk Plan Orders*

### Required Data

You must create a saved query that points out the order releases you want to include.

### Setup

You control validation of incoming transmissions with the glog.integration.validation.orderinterface property.

### What Data Goes into the Transmission?

If you can keep all your TransOrders within one transmission follow these steps:

1. Set IsProcessInSequence=N.

   This ensures maximum performance because Oracle Transportation Management can process TransOrders belonging to different order bases in parallel.
2. Include all TransOrders that should be bulk planned.
3. Create an order release for every TransOrder either with IsShippable=Y and omit the ReleaseInstruction, or with IsShippable=N and include a ReleaseInstruction.
4. Include a Topic element as the last element in the transmission to start the bulk planning. Set TopicArgName to 'savedQuery' and TopicArgValue to a Query_Name. The saved query must point out the Order Releases you want to include.
5. In the GLogXMLElement holding the Topic element, include a ProcessInfo element with WhenToProcess=END_OF_TRANSMISSION.

   This tells Oracle Transportation Management to wait to start the bulk planning until the end of the transmission.

   **Note:** Oracle Transportation Management plans all order releases that match the saved query, not just the ones within the transmission.

   **Note:** The Topic element must be the last element in the transmission. If it is not, Oracle Transportation Management will plan incorrectly.

If you cannot keep all your TransOrders within one Transmission follow these guidelines:

1. For every transmission with TransOrder, set IsProcessInSequence=N.
2. Create an order release for every TransOrder either with IsShippable=Y and omit the ReleaseInstruction, or with IsShippable=N and include a ReleaseInstruction.
3. For every TransOrder that Oracle Transportation Management should bulk plan later, set the ProcessingCodeGID to NOPLN.

   If you instead set the ProcessingCodeGID to PLN on each TransOrder in a transmission, Oracle Transportation Management bulk plans these orders on each transmission. Also, Oracle Transportation Management cannot supply a bulk plan history in this case.
4. When Oracle Transportation Management has received all TransOrders to be bulk planned, send a Topic element as the last element in the transmission to start the bulk planning. Set TopicArgName to 'savedQuery' and TopicArgValue to a Query_Name. The saved query must point out the Order Releases you want to include. If you want to supply your own bulk plan ID, in addition, set TopicArgName to 'bulkPlanID' and TopicArgValue to your desired bulk plan ID.

   To be reasonably sure that Oracle Transportation Management has received all your transmissions, allow sufficient amount of time between sending the last TransOrder Transmission and sending the Topic element.

   **Note:** Oracle Transportation Management plans all order releases that match the saved query, not just the ones within the last transmissions.

   **Note:** The Topic element must be the last element in the transmission or group. If it is not, Oracle Transportation Management will plan incorrectly.

### Transmission Results

When Oracle Transportation Management completes the bulk planning, Oracle Transportation Management sends the results of the bulk plan in a BulkPlan element.

#### Error Messages

If you receive a TransmissionReport, check for integration messages.

### *Incrementally Release TransOrder Line from Existing TransOrder*

In this scenario, you already have a TransOrder with a large amount of goods in a TransOrderLine in Oracle Transportation Management but now you want to release small amounts of that TransOrderLine with multiple subsequent TransOrders.

To incrementally release TransOrderLines from an order already in Oracle Transportation Management via an integration transmission, do the following:

#### Required Data

#### Setup

You control validation of incoming transmissions with the glog.integration.validation.orderinterface property.

#### What Data Goes into the Transmission?

1. Make sure the public Order Base - Modify - Incremental Release agent is active.
2. Send a transmission of the record and enter the transaction code U in the TransactionCode element.
3. All your TransOrderLines must be marked IsShippable=N.
4. Always keep IsShippable=N between all these TransOrders.
5. Include a TransOrderHeader/ReleaseInstruction to release a fraction of the amount specified on the original TransOrderLine. If you omit the ReleaseInstruction element, Oracle Transportation Management only saves your order base since you have IsShippable set to N.
6. For each modified TransOrder you send, update the ReleaseInstruction/SequenceNumber and make it unique. If you do not, Oracle Transportation Management keeps the old releases but replaces the content of the release instruction.
7. See the Order Base Manager for a description of the fields.
8. See the TransOrder Diagram in the XML schema to learn which elements are required. To view the diagrams use an XML application like XML Spy.

#### Transmission Results

1. Oracle Transportation Management receives your transmission and starts to process your transmission internally.
2. Oracle Transportation Management starts the public Order Base - Modify - Incremental Release agent.
3. Oracle Transportation Management raises events that in turn can trigger notifications to be sent.

#### Error Messages

If you receive a TransmissionReport, check for integration messages.

### *Send TransOrder from Oracle Transportation Management*

#### Required Data

#### Setup

You must have created the external system you want to send to.

#### How To Send the Transmission?

In the Order Base Manager.

In the Process Manager, Send Integration Page.

#### What Data Goes into the Transmission?

Oracle Transportation Management includes all data specified for the order base.

#### Transmission Results

#### Error Messages

### *Processing Codes*

When you send an order to Oracle Transportation Management, you can indicate whether you want Oracle Transportation Management to perform planning functions on it. If Oracle Transportation Management plans orders, it creates shipments from the orders and then executes the shipments as soon as it receives them. If you want to execute orders into shipments at a particular time or after you receive a certain number of orders, do not run the planning function.

Control the details of planning orders in Oracle Transportation Management in the Agent Manager.

In the ProcessingCodeGID element, enter one of the following values:

- **NOPLN**: Instructs Oracle Transportation Management not to plan shipments form the order. This is the default if you omit this element.
- **PLN**: Instructs Oracle Transportation Management to plan shipments from the order release. Oracle Transportation Management will plan multi-stop shipments if appropriate. You must have your TransOrder set up to create an order release for this to work.
- **MSPLN**: Obsolete.

## Voucher Interface

Oracle Transportation Management sends voucher transmissions to the URL you specify in glog.properties.

You can create an agent that send the Voucher interface using the agent action Send Voucher Interface. You can also send this interface from the Invoice Manager.

**Note:** You can send a voucher transmission that cancels or edits a previous voucher.

Oracle Transportation Management determines to whom a payment is due based on the involved parties defined on the order release.

The Shipment element is only included when generating vouchers for parent invoices. When generating a voucher for a child invoice, the Shipment element is not included.

You can optionally use the ShipmentGID element instead of the full Shipment element in order to reduce the size of the Voucher XML.

## DataQuerySummary Interface

Contains the GID of a business object record.

Some external systems may not be prepared for Oracle Transportation Management to send large amounts of data. The DataQuerySummary interface provides a mechanism to send only a summary of the data. The external system can request the individual records from Oracle Transportation Management at appropriate times (e.g. idle times, overnight) by referencing the GID.

### *Send DataQuery Summary from Oracle Transportation Management*

#### Required Data

#### Setup

You must have created the external system you want to send to.

#### How To Send the Transmission?

Mark the Send Summary check box when sending from one of these managers:

- Order Base Manager
- Order Manager (use for order releases)
- Buy Shipment Manager
- Sell Shipment Manager
- Shipment Group Manager
- Bill Manager
- Service Provider Manager
- Location Manager
- Rate Offering Manager
- Rate Record Manager
- Item Manager
- Invoice Manager
- In the Process Manager, Send Integration Page.

#### What Data Goes into the Transmission?

Oracle Transportation Management includes only the GID of the record.

#### Transmission Results

#### Error Messages

### *The Role of the Transaction Code*

The TransactionCode specifies whether the information should be inserted, or updated/replaced. For the Refnum objects that have the qualifier and value as part of the primary key, the TransactionCode indicates whether the new qualifier/value pair should be added (Insert), or used to replace all of the current records with the same qualifier (Update).

For example, the Shipment_Refnum table has a composite primary key made up of the ShipmentGID, RefnumQualifier, and RefnumValue. Assume a shipment has the following ShipmentRefnum Qualifier/Value pairs in the system: CO/A-12345, CO/B-89387, CN/C-83920. If you send a new refnum qualifier/value of CO/D-23849 using the GenericStatusUpdate interface, the TransactionCode would affect the change as follows:

- TransactionCode = I: The new refnum would be added, resulting in all of the following being present in the table: CO/A-12345, CO/B-89387, CN/C-83920, CO/D-23849
- TransactionCode = U: The current reference numbers with the same qualifier would be deleted, and replaced by the new one. In this case, the result would leave the following in the table: CN/C-83920, CO/D-23849

The TransactionCode is only applicable for the Refnum and Remark elements. It is not used for the Status or Indicator elements, which are only intended to be updated.

## Topic Interface

This inbound interface allows you to raise a topic and get Oracle Transportation Management to start processing an object. Currently Oracle Transportation Management supports BuildBuySideShipments and BuildSellSideShipments that allows you to start bulk planning. Oracle Transportation Management also supports clearing caches using the interface.

> **Note:** Make sure Oracle Transportation Management has released all your TransOrders before sending the Topic element to Oracle Transportation Management.

> **Note:** When including other transactions in the same transmission as the Topic transaction, make the Topic transaction the last in the transmission.

The table lists what each element should contain.

| **TopicAliasName** | **TopicArgName** | **TopicArgValue** |
|---|---|---|
| BuildBuySideShipments | savedQuery | query_name, e.g. YELLOW_ORDER_REL |
| BuildSellSideShipments | savedQuery | query_name, e.g. YELLOW_ORDER_REL |
| glog.server.workflow.adhoc.ClearCaches | cache | partial or full string matching the cache name, e.g. RateOffering |
| | zone | zone name, e.g. Rating or Business |
| | exactMatch | true, if the cache match should be exact |

## SKU Interface

### How to Structure Your Data

If you have large amounts of highly complex, nested inventory information, you should use the XML column in the SKU_DESCRIPTOR table to store that information, rather than using nested SKU_DESCRIPTOR records. However, it will not be possible to search for SKUs using the XML info. You will only be able to search on the non-XML columns in the SKU and SKU_DESCRIPTOR tables.

If you have small or medium amounts of less complex inventory information, you can use nested SKU_DESCRIPTOR records instead. Using this method, it will be possible to find a SKU by a sub-descriptor.

### SKU Table

The following sample SKU record shows a part used to make Novelty phones. The Novelty stock code is 2002, which is used to form the XID. This corresponds to the packaged_item novelty.8946. The warehouse is novelty.wh1. The supplier is General Electric, who also currently owns the inventory.

```
SKU_GID = novelty.2002-wh1
SKU_xid = 2002-wh1
Packaged_item_GID = novelty.8946
Warehouse_location_GID = novelty.wh1
Supplier_corporation_GID = novelty.ge
Owner_corporation_GID = novelty.ge
Quantity_on_hand = 1800
Min_level = 100
Max_level = 2000
Domain_name = novelty
```

### SKU_DESCRIPTOR table - BLOB

Oracle Transportation Management cannot show BLOBs in tree view of the inventory manager.

This section illustrates the relational approach to storing SKU descriptor and sub-descriptor information in using the SKU_DESCRIPTOR table. This method would be used when it is necessary to use standard SQL to search SKU descriptor data.

The example below shows a top-level SKU_DESCRIPTOR record. Notice that the parent_sku_descriptor_seq is null.

```
SKU_GID = novelty.2002-wh1
SKU_descriptor_seq = 1
SKU_descriptor_type = status
SKU_desriptor_value = held
SKU_descriptor_quantity = 1000
Parent_sku_descriptor_seq = null
Domain_name = novelty
```

The example below shows a level-2 SKU_DESCRIPTOR record. The parent_sku_descriptor_seq is set to 1, pointing to the previous example.

```
SKU_GID = novelty.2002-wh1
SKU_descriptor_seq = 2
SKU_descriptor_type = reason
SKU_descriptor_value = damaged
SKU_descriptor_quantity = 600
```

```
Parent_sku_descriptor_seq = 1
Domain_name = novelty
```

The example below shows a level-3 SKU_DESCRIPTOR record. The parent_sku_descriptor_seq is set to 2, pointing to the previous example.

```
SKU_GID = novelty.2002-wh1
SKU_descriptor_seq = 3
SKU_descriptor_type = batch
SKU_descriptor_value = 001
SKU_descriptor_quantity = 250
Parent_sku_descriptor_seq = 2
Domain_name = novelty

SKU_GID = novelty.2002
SKU_descriptor_seq = 4
SKU_descriptor_type = batch
SKU_descriptor_value = 002
SKU_descriptor_quantity = 300
Parent_sku_descriptor_seq = 2
Domain_name = novelty

SKU_GID = novelty.2002
SKU_descriptor_seq = 5
SKU_descriptor_type = batch
SKU_descriptor_value = 003
SKU_descriptor_quantity = 50
Parent_sku_descriptor_seq = 2
Domain_name = novelty
```

### SKU_DESCRIPTOR Table - XML

If the SKU descriptor information need not be fully searchable using standard SQL, then the XML column in the SKU_DESCRIPTOR table may be used to represent the information at level 2 and below. In other words, it would be possible to use standard SQL to search for a SKU descriptor, but not for a SKU sub-descriptor.

An example situation where the XML method may not be appropriate would be where the top level SKU is a combination of shoes of different styles. The top level SKU_DESCRIPTOR records would have one row for each style. The level 2 SKU_DESCRIPTOR would have counts of sizes within each style. A query to determine the total inventory of size 9 shoes across all styles would not be possible using the XML method. You can think of similar examples for the auto industry, i.e. find the inventory of all cars with anti-lock brakes, etc.

When using the XML method for representing detailed SKU_DESCRIPTOR information, each client implementation will be responsible for developing their own industry-specific XML schema for that information. By default, the UI will display this information in a nicely formatted manner. The UI provides a mechanism whereby you can install custom XSL for formatting information. However, this XSL file is purely optional.

Below is a snippet of how the information from the previous section might appear in the database if the XML approach is used instead of the nested SKU_DESCRIPTOR method. In this case, the parent_sku_descriptor_seq column is always null, and the XML column is used instead. In this case, the top level status information is available relationally. However, the lower level descriptors within that status are represented inside the XML.

```
SKU_GID=novelty.2002-wh1
SKU_descriptor_seq = 1
```

```
SKU_descriptor_type = status
SKU_descriptor_value = held
SKU_descriptor_quantity = 1000
Domain_name = novelty
Xml =
<SkuDescriptor>
  <type>damaged</type>
  <value>001</value>
  <quantity>600</quantity>
<SkuDescriptor>
  <type>batch</type>
  <value>001</value>
  <quantity>250</quantity>
</SkuDescriptor>
<SkuDescriptor>
  <type>batch</type>
  <value>002</value>
  <quantity>300</quantity>
</SkuDescriptor>
… etc …
</SkuDescriptor>
```

## CharterVoyage Interface

The CharterVoyage interface is used to specify the charter voyage for creating a consol shipment. It is supported on both the inbound and the outbound.

A charter voyage represents an ocean transport movement by a carrier from a loading port to a discharge port. Within a charter voyage, there are several Stowage Modes, which represent, at a conceptual level, separate "compartments" within the charter voyage. There is also capacity associated with each stowage mode as defined on a consol that you create for each stowage mode defined on the charter voyage. This capacity controls the orders that can be booked on the charter voyage.

For a charter voyage, and each of its defined stowage modes, you can create a consol that has a single empty shipment attached. For each voyage, one consol is automatically created for each stowage mode defined on the voyage. A shipment is also created for each consol at the same time.

Most of the elements included in the CharterVoyage interface follow the fields available in the Charter Voyage manager and the Charter Voyage Stowage Details.

## Consol Interface

The Consol interface is used to specify the shipment consolidator. It is supported on both the inbound and the outbound. A consol can be created for a charter voyage or air schedule (flight).

A charter voyage consol represents the weight, volume, FEU/TEU capacities of a specific stowage mode on a specific charter voyage. It captures the allocated, maximum, committed, booked, and produced capacity values when the status of a consol is changed as a result of booking orders on a shipment that is related to the consol.

For a freight forwarder, the consol is considered a group of house bills or a set of sell shipments. All actions related to manipulating a consol should be performed from the perspective of a sell shipment. For example, adding freight to a consol would be performed by selecting sell shipments to add to consol.

For example, a freight forwarder starts with a group of house bills or a set of sell shipments. They have also reserved flights. For each flight reservation, there is a consol for defining the reserved capacity of the flight. The sell shipments are then booked to consols to create buy shipments.

## Generic Status Update Interface

The Generic Status Update interface is used to modify one of a number of simple properties of a set of business objects. The properties that can be modified are:

- Status
- Indicator
- Reference Number
- Remark

The following table shows which properties are supported by each valid business object (the `GenericStatusObjectType` is the value that must be passed in the transaction XML):

| **Business Object (GenericStatusObjectType)** | **Status** | **Indicator** | **Refnum** | **Remark** |
|---|---|---|---|---|
| INVOICE_LINEITEM | | | x | x |
| INVOICE | x | x | x | x |
| LOCATION | x | | x | x |
| OB_LINE | x | | x | x |
| OB_ORDER_BASE | x | x | x | x |
| OB_SHIP_UNIT | x | | x | x |
| ORDER_RELEASE | x | x | x | x |
| SHIPMENT | x | x | x | x |
| SHIPMENT_STOP | | | x | x |
| SHIP_GROUP | x | x | x | x |
| S_SHIP_UNIT | | | x | x |
| S_SHIP_UNIT_LINE | | | x | x |
| VOUCHER | | x | x | x |
| SCHEDULE | x | | | |
| CHARTER_VOYAGE | | | x | |
| CLAIM_LINE_ITEM | | | x | |
| CLAIM | x | | x | x |

| **Business Object**<br><br>**(GenericStatusObjectType)** | **Status** | **Indicator** | **Refnum** | **Remark** |
|---|---|---|---|---|
| CONSOL | x | | | |
| ITEM | x | | x | x |
| JOB | x | | x | x |
| ORDER_RELEASE_LINE | | | x | x |
| DOCUMENT | x | | | |
| CONTAINER_GROUP | x | | | |
| DRIVER | x | x | x | x |
| DRIVER_TYPE | | | | x |
| POWER_UNIT | x | x | x | x |
| POWER_UNIT_TYPE | | | | x |
| OR_STOP | | | | x |
| EQUIPMENT | x | | x | x |
| EQUIPMENT_TYPE | | | | x |
| WORK_INVOICE | x | | | |
| SHIP_STATUS_SPCL_SERVICE | | | | x |
| ORDER_MOVEMENT | x | | x | x |
| CONTACT | x | x | x | x |
| GTM_TRANSACTION | x | | x | x |
| GTM_TRANSACTION_LINE | x | | x | x |
| GTM_REGISTRATION | x | | x | x |
| GTM_STRUCTURE | | | x | x |