Oracle Fusion Cloud Transportation and Global Trade Management

Data Management Guide

Release 26A

Oracle Fusion Cloud Transportation and Global Trade Management
Data Management Guide

Release 26A

G36016-01

Author: Oracle Transportation and Global Trade Management Product Team

Contents

Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

Get Help in the Applications

Use help icons to access help in the application.

Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, and watch events.

Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to *otm-doc_us@oracle.com*.

Thanks for helping us improve our user assistance!

2 Introduction

Intended Audience

This manual provides step-by-step instructions for importing and exporting data in DB.XML and CSV formats.

> **Note:** This manual provides examples of CSV, XML, and schema diagrams. For actual database tables and schema, refer to the latest database schema and the Integration schema.

Introduction to DB.XML

DB.XML (Database-centric XML) is an XML file format for importing and exporting Oracle Transportation Management data.

The DB.XML functionality provides the ability to query/insert/update/delete data directly from/to the Transportation and Global Trade Management Cloud database tables. As such, this functionality should only be used by privileged individuals who understand the responsibilities and capabilities that come with using this functionality. Each import or export request is limited to 100 MB.

> **Note:** Updates made directly to the Transportation and Global Trade Management Cloud through the DB.XML Import functionality can only ensure data consistency with respect to the standard database constraints, e.g. Primary Key, Foreign Key, and Check constraints. Imports don't flow through the main application logic for updates, and so cannot check that the business context of a particular change makes sense. For example, the status of a particular object (e.g. LOCATION STATUS) can be updated. Import can only check that the status GID is valid but not that the status, possibly in association with other status values, constitutes an appropriate state for the object to be in.

Why do I want to use DB.XML?

You want to use DB.XML functionality if you have Transportation and Global Trade Management data you need to move in or out of the application. The XML format follows XML standards on portability, structure, correctness, and directly maps XML element names to database table names with the XML element attribute names directly mapping to the database columns. The XML element attribute values are the values for the database column. When compared to the CSV (Comma Separated Values) format, DB.XML doesn’t require each individual value to be separated, quoted, or in the same column order. Both DB.XML and CSV support manipulation of parent-child records as a unit, however it is more straightforward with DB.XML. These give DB.XML advantages compared to CSV when inserting or updating any records especially rate information. DB.XML also supports column values that contain line breaks and commas, which cannot be imported from a CSV file.

How can I use DB.XML?

There are a few ways to perform a DB.XML export or import:

- Transportation and Global Trade Management Cloud User Interface
- HTTP POST to servlet on Transportation and Global Trade Management Cloud application server (requires authentication)
- SOAP web service

See the *DB.XML* chapter for details.

DB.XML Format

In the DB XML file, there can be more than one parent element for each business object contained within what is called a **Transaction Set**. The TRANSACTION_SET element is used to contain these **parent** elements. The parent element itself may contain one or more child element. DB XML Import and Export can work with complete parent-child table relationships all in one file by using corresponding parent-child elements. The attribute values on each element correspond to column values.

> **Note:** The convention used here is that a table is called the "child" table if it contains a foreign key to another table. The table referenced by the foreign key is called the "parent".

These parent elements typically correspond to the primary Transportation and Global Trade Management Cloud data objects – AGENT, LOCATION, etc., and child elements typically correspond to associated child tables. For example, for the LOCATION parent table, the child table could be LOCATION_REFNUM, LOCATION_STATUS, etc.

In the case where the transaction set is used for data import, each parent element will, by default, be treated as a distinct transaction, i.e. the parent element and all its child elements are saved to the database as one atomic transaction. If one child element fails, the parent element transaction fails. The failure of one parent element does not directly affect the transactions for other parent elements. Additionally, all parent elements can be treated as one unit of work i.e. if one element fails, ALL elements in the ‘set’ will fail. The details on how this is achieved are covered in the *DB.XML* chapter.

Oracle Transportation Management ignores element and attribute names that do not correspond to valid database table or column names. This allows you to comment your DB.XML file without affecting what is imported.

Performance Considerations

The primary use case for DB.XML import and export was for the setup of configuration data needed for implementation of business processes. The amount of data that can be imported and exported will be limited by default to 100 MB (megabytes) in order to protect the server and avoid excessive data retrieval or updates which could affect stability.

The limit can not increased.

Migration Project

The Migration Project is a standard way to define and manage one or more datasets for the purpose of migrating data from one Oracle Transportation Management instance to another.

Although the Oracle Transportation Management application is fully functional “out of the box”, an operational system will typically require some configuration. Best practice would be for such a configuration to be developed and tested in a pre-production environment, accepted by product and business/operational experts and then promoted to the production environment.

At a high level, an "Export" migration project is defined on source system. This migration project contains data sets which represent object groups and objects that are to be migrated. There are two ways to import this data set on to the target system.

- Migration using Zip File
- Migration using Web Service (Supported only for Oracle Logistics Cloud)

Migration Process will create or update corresponding "Import" migration project on the target system to list the imported data and the success and failures.

For more information on the Migration Project Process see the *About the Promote to Production (P2P) Process help*.

Introduction to CSV

OTM supports the ability to import and export data in the CSV format in and out of the database.

CSVUtil also exports data as a script of insert statements. This document describes how to use OTM’s CSV capabilities and shows some sample CSV files.

CSV files are compact and enable you to export/import large amounts of data into Oracle Transportation Management. For example, you can use CSVs when importing rates into Oracle Transportation Management.

> **Note:** Updates made directly to the Transportation and Global Trade Management Cloud database by CSV import can only ensure data consistency with respect to the standard database constraints, e.g. Primary Key, Foreign Key, and Check constraints. Imports do not flow through the main application logic for updates, and so cannot check that the business context of a particular change makes sense. For example, the status of a particular object (e.g. LOCATION STATUS) can be updated. Import can only check that the status GID is valid but not that the status, possibly in association with other status values, constitutes an appropriate state for the object to be in.

> **Note:** CSVUtil is deliberately non-transactional, and therefore may return inconsistent data if modifications to exported tables are made concurrently to the CSVUtil operation.

> **Note:** CSV files cannot contain column values that contain line breaks (such as a multi-line XML document). Unless the database table uses the CLOB type for the column, you cannot import or export such columns in CSV format (use DB.XML format instead). If the column is of type CLOB, additional files for the CLOB column of each record can be read or written, but only when using CSVUtil in ZIP file mode through the web interface. See below for further details.

How Can I Use CSV?

There are a few ways to use CSVUtil:

- Via the Oracle Transportation Management web interface
- Via integration transmissions

A Sample CSV File

Below are the contents of a sample CSV file:

```
ICON
ICON_GID,ICON_XID,DESCRIPTION,PATH,DOMAIN_NAME,INSERT_USER,INSERT_DATE,UPDATE_USER,UPDATE_DATE
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
"BATCH_GRID","BATCH_GRID","Reports Batch Grid","/images/icons/reports/
batch_grid.jpg","PUBLIC","DBA.ADMIN","20040310091645","DBA.ADMIN","20040630100834"
```

Line 1 must be the name of the table.

Line 2 must be a comma-separated list of column names. Only the columns being loaded must be specified. Note that CSVUtil is case-sensitive with regards to column names; all column names must be in upper case if using standard database, or in mixed ("camel") case without underscores if using appserver mode (see chapter 6 for more details). Note also that if updating or deleting existing records, all columns in the primary key of the specified table must be included so the records to be updated or deleted can be precisely identified; CSVUtil does not permit "wildcard" updates or deletes using partial primary keys.

After line 2 may be one or more optional EXEC SQL lines, such as the one shown above, to set the date format.

Subsequent lines include the data. The number of columns of data must correspond to the number of columns specified on line 2. The ordering of the data columns must also correspond to line 2.

Character data may be surrounded with double-quotes, as shown above. If you need to include a double-quote character, use “"” instead. The tools described here to export CSV files automatically convert double-quote characters into “"”.

Numeric data should not be surrounded with double-quotes.

Multi-table CSV Files

The output produced by the xcsvw* commands is in multi-table CSV format. The various CSV import commands recognize this format also:

- The first record in a multi-format file must be "$HEADER".
- The header section contains table names and the names of the columns used in that table.
- After the header section comes the body, identified by the $BODY keyword.
- Each data record in the $BODY must be preceded by its table name on the prior line.

Here is an example:

```
$HEADER
LOCATION_ROLE_PROFILE
```

```
LOCATION_GID,LOCATION_ROLE_GID,CALENDAR_GID,FIXED_STOP_TIME, etc...
LOCATION_STATUS
LOCATION_GID,STATUS_TYPE_GID,STATUS_VALUE_GID,DOMAIN_NAME,INSERT_USER,INSERT_DATE,UPDATE_USER,UPDATE_DATE
LOCATION_CORPORATION
LOCATION_GID,CORPORATION_GID,DOMAIN_NAME,INSERT_DATE,UPDATE_DATE,INSERT_USER,UPDATE_USER
LOCATION_ADDRESS
LOCATION_GID,LINE_SEQUENCE,ADDRESS_LINE,DOMAIN_NAME,INSERT_USER,INSERT_DATE,UPDATE_USER,UPDATE_DATE
LOCATION_REFNUM
LOCATION_GID,LOCATION_REFNUM_QUAL_GID,LOCATION_REFNUM_VALUE,DOMAIN_NAME,INSERT_DATE, etc...
LOCATION
LOCATION_GID,LOCATION_XID,LOCATION_NAME,ADDRESS_LINE1,ADDRESS_LINE2,CITY,etc.
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS..'
$BODY
LOCATION
"GUEST.00621918","00621918","00621918",,,,,"TN",,"USA",,,,,"America/
New_York",,,,,,,,"N","N","COMMERCIAL",,,"GUEST","S",0,...etc
LOCATION_ADDRESS
"GUEST.00621918",1,,"GUEST","DBA.ADMIN",2001-10-07 17:53:53.0,,
LOCATION_ADDRESS
"GUEST.00621918",2,,"GUEST","DBA.ADMIN",2001-10-07 17:53:53.0,,
LOCATION_CORPORATION
"GUEST.00621918","GUEST.CUST NO","GUEST",2001-10-15 10:50:49.0,,"DBA.ADMIN",
LOCATION_REFNUM
"GUEST.00621918","GLOG","GUEST.00621918","GUEST",2001-10-25 17:13:48.0,2001-10-19
 18:23:17.0,"DBA.ADMIN","DBA.GLOGOWNER"
LOCATION_ROLE_PROFILE
"GUEST.00621918","SHIPFROM/SHIPTO",,0,0,"GUEST","S",0,"S",0,"N",,,,,,,,,,2001-10-25 14:12:38.0,2002-08-28
 19:13:05.0,"DBA.ADMIN", etc.
LOCATION_STATUS
"GUEST.00621918","GUEST.CREDIT LEVEL","GUEST.CREDIT LEVEL_UNKNOWN","GUEST","DBA.GLOGOWNER",2001-10-17
 09:38:05.0,,
```

Export via JSON through REST API

See the *REST API for Fusion Cloud Transportation and Global Trade Management Data Export guide*.

International Characters

Here are some tips for importing and exporting international characters.

Import

To be able to send data to Oracle Transportation Management containing characters outside the 7-bit ASCII character set, you must:

- Make sure your database uses an encoding that can handle all the characters you need.
- Always save your files using UTF-8 format.

XML Spy, Textpad, and Notepad (Microsoft Windows 2000 or better) can all save in UTF-8 format.

Before you edit your files, you need to ensure that you configure your text editor to use the appropriate font and script (sometimes called subset). A script is a collection of characters such as Western European, Greek or Turkish.

For example, if you need to update files containing Czech characters, then you need to select a font that supports an Eastern European script such as Arial or Arial Unicode Ms.

Export

When exporting files, Oracle Transportation Management writes files in UTF-8. Note that when you view data in your browser and then use the view source option to save your data, just save your file without specifying an encoding. Later, when editing your file, use an editor that support UTF-8.

Special Character Limitations

Lower case alphabet "x" is reserved for separating multi-part primary keys for OTM objects and should not be used as part of an object's domain name or ID (XID). As an example, a shipment stop object GID consists of a shipment GID and a stop number. They would be concatenated with a delimiting character "x". A specific example of a shipment stop GID would look like GUEST.ShipABCx1 where GUEST.ShipABC is the shipment GID and 1 is the stop number. So using the character "x" for object IDs should be avoided when using either DB.XML or CSV or any other export/import mechanism to prevent potential issues during the export of these records.

Best Practices

Whether you are using DB.XML or CSV export, you should follow some basic rules to help maintain proper version control and avoid data inconsistencies.

In order to maintain proper version control and track all changes that are being made to agents, we recommend that you **do not update existing active agents**. Instead, we recommend the current agent be disabled and that **a new agent is created** with the changes that need to be made. This will allow you to easily revert back to the previous version should you run into anything unexpected when the new agent is being used.

This can be achieved simply by adding a date, "02252013" or a version identifier 'V#', i.e. "V1", "V2", "V3" etc. to the end of the AGENT_XID when the agent is being created. If an existing agent is exported, modify the AGENT_GID and AGENT_XID before it is imported into another instance.

For instance if you want to create a new agent you may call it "SHIPMENT-CREATED V1".

If you decide you want to make changes to this agent you would create a new agent called "SHIPMENT-CREATED V2", turn off "SHIPMENT-CREATED V1", and turn on the new agent.

Now if you decide the new agent is not working as expected the new agent can simply be turned off and the original agent can be turned back on to restore the original workflow.

3 DB.XML

DB.XML Export

The DB XML Export process produces a **transaction set** which can be viewed in the UI or saved as an external XML file.

There are a number of ways to specify the data to be exported:

- Specifying a SQL Query
- Migration entity

For all methods described below there is an option to specify whether the so-called “foot print” columns are included in the export. The foot print columns are the INSERT_DATE, INSERT_USER, UPDATE_DATE, and UPDATE_USER. The values for these columns would be updated for any subsequently imported data by way of INSERT/UPDATE triggers and so their presence in exported data is largely informational.

By default, LOB data (BLOB and CLOB) is exported as a ‘base 64’ encoded string attribute value. Working with the CLOB data, for example, would require decoding the string to obtain the original character text, editing the text and then encoding again to a ‘base 64’ string.

To make this process easier, and as a way to avoid exceeding any XML attribute length limit, it is possible to export LOB data as separate elements. For example, if `parent_col2` is a CLOB column and `parent_col3` is a BLOB column:

```
<?xml version="1.0" encoding="UTF-8"?>
<xml2sql Version=”20B”>
 <TRANSACTION_SET useLOBElement="true">
 <"parent table name" parent_col1="value_1" parent_col2="CLOBID#1" parent_col3="BLOBID#1" ..etc..>
 <"child table name" child_col1="value_1" ..etc..>
 ...etc...
 </"child table name">
 <CLOB ID="CLOBID#1">
 <![CDATA[
 ...Both xml and non-xml text appears here...
 ]]>
 </CLOB>
 <BLOB ID="BLOBID#1">
 ...base 64 encoded binary value appears here...
 </BLOB>
 </"parent table name">
 </TRANSACTION_SET>
 </xml2sql>
```

The attribute value for the LOB column will then contain a unique ID for the element containing the CLOB or BLOB content. Note that BLOB content must always be a ‘base 64’ encoded string. The “useLOBElement” is generated on export and required on import for DB.XML files that contain the above format.

Using an Object Set Name

The **Object Set Name** is a named list of Object Names (described above). This allows a logical grouping of data to be exported in one file. The base install will provide the following pre-configured queries:

Object Set Names

Object Set Name	Associated Object Names
DomainReferenceData	STATUS_TYPE, WORKFLOW_TOPIC_INFO, NOTIFY_SUBJECT_CONTACT, PLANNING_PARAMETER, BN_RULE, NOTIFY_SUBJECT_STYLESHEET

New customer-defined object set names and object lists can be added by setting the associated properties. For example, the above set is defined by the property:

```
glog.integration.dbxml.objectset.DomainReferenceData=STATUS_TYPE,WORKFLOW_TOPIC_INFO,NOTIFY_SUBJECT_CONTACT,PLANNING
```

The use of a "where clause" is mandatory for the export by Object Set.

Specifying a SQL Query

The complete SQL query can also be specified directly. When this approach is used there must be an additional "Root Name" parameter given. This is used as the name of the top level parent element name for each record retrieved by the query.

Migration Entity

The Migration Entity names are a list of all the application objects available in Transportation and Global Trade Management Cloud. They are designed to support the export of a top level object, e.g. Location and all its child objects, e.g. LocationStatus, LocationRefnum, etc. However, the list also contains the respective child entity names to support fine grained export.

The specific entities are retrieved based on a comma separated list of unique object primary keys for the entity name selected. For example, if Location were the selected entity the Object ID would be a comma separated list of Location Primary Key strings.

DB.XML Import

The DB XML import process takes a **transaction set**, contained in an input XML file or message, and inserts, updates, or deletes rows in Transportation and Global Trade Management Cloud tables. It can also completely replace a current set of child records with a new set.

The valid parent/child relationships are determined by the persistence logic internal to the Transportation and Global Trade Management application server. Therefore, imports should normally only be used for data exported using the Migration Entity export type. Content exported via other DBXML execution modes in earlier releases may still be imported but any customer-defined parent/child relationship tree must be a subset of that produced by the Migration Entity export type.

DB.XML Transaction Codes

The **Transaction Code** specifies how the transaction set is to be processed and will be one of:

- **I**: Insert new records
- **II**: Insert new record or ignore (i.e. do not fail) if already existing
- **IU**: Insert new records or update if already existing
- **D**: Deletes record.
- **RC**: Replace Children. Delete existing children and replace with new.

Note: The use of the Delete transaction code should be used with care. The delete process attempts to also remove any objects which contain a foreign key to the object being deleted which may not always be the desired effect. Therefore it is strongly advised **NOT** to use this transaction code for object types that may be referenced from transaction data. An example of this is the Location object. Locations can be stand alone but they can also be ship from or ship to locations on Shipments and deleting such locations could have unintended consequences.

Replace Children

When using the **RC** transaction code the child tables that should be involved can be specified as **Managed Tables**. There are also some standard managed tables defined for some data objects which are combined with any managed tables entered as input.

Replace Children

Object Name	Child Tables
LOCATION	LOCATION_ACCESSORIAL, LOCATION_ADDRESS, LOCATION_CORPORATION, LOCATION_REFNUM, LOCATION_REMARK, LOCATION_ROLE_PROFILE, LOCATION_SPECIAL_SERVICE, LOCATION_STATUS
RATE_GEO	RATE_GEO_STOPS, RATE_GEO_ACCESSORIAL, RG_SPECIAL_SERVICE, RG_SPECIAL_SERVICE_ACCESSORIAL, RATE_GEO_COST_GROUP, RATE_GEO_COST, RATE_GEO_COST_WEIGHT_BREAK
RATE_OFFERING	RATE_OFFERING_STOPS, RATE_OFFERING_ACCESSORIAL, RATE_OFFERING_COMMENT
AGENT_EVENT	AGENT_EVENTS_INVALID_ACTION
AGENT	AGENT_EVENT_DETAILS, AGENT_ACTION_DETAILS
CORPORATION	CORPORATION_INVOLVED_PARTY
SAVED_QUERY	SAVED_QUERY_VALUES, SAVED_QUERY_SORT_ORDER
SAVED_CONDITION	SAVED_CONDITION_QUERY
MONITOR_PROFILE	MONITOR_AGENT, MONITOR_AGENT_LINK
SHIPMENT	SHIPMENT_STOP, SHIPMENT_STOP_D, SHIPMENT_STOP_REMARK, SHIPMENT_ACCESSORIAL, SHIPMENT_BILL, SHIPMENT_COST, SHIPMENT_COST_REF, SHIPMENT_INVOLVED_PARTY, SHIPMENT_REFNUM, SHIPMENT_REMARK, SHIPMENT_SPECIAL_SERVICE, SHIPMENT_STATUS
STATUS_TYPE	STATUS_VALUE
WORKFLOW_TOPIC_INFO	WORKFLOW_TOPIC_PARAM, WORKFLOW_INFO, WORKFLOW_PARAM

Object Name	Child Tables
OB_ORDER_BASE	OB_ACCESSORIAL, OB_INVOLVED_PARTY, OB_LINE, OB_LINE_ACCESSORIAL, OB_LINE_ATTRIBUTE, OB_LINE_REFNUM, OB_LINE_REMARK, OB_LINE_SPECIAL_SERVICE, OB_LINE_STATUS, OB_ORDER_BASE, OB_ORDER_BASE_STATUS, OB_REFNUM, OB_REMARK, OB_SHIP_UNIT, OB_SHIP_UNIT_CONTENT, OB_SHIP_UNIT_REFNUM, OB_SHIP_UNIT_REMARK, OB_SHIP_UNIT_SEAL, OB_SHIP_UNIT_STATUS, OB_SPECIAL_SERVICE, OB_SU_ACCESSORIAL, OB_SU_CONTENT_ATTRIBUTE, OB_SU_CONTENT_REFNUM, OB_SU_CONTENT_REMARK, OB_SU_SPECIAL_SERVICE
BS_STATUS_CODE_PROFILE	BS_STATUS_CODE_PROFILE_D
BS_EVENT_GROUP	BS_EVENT_GROUP_ATTRIBUTE
OUT_XML_PROFILE	OUT_XML_PROFILE_D, OUT_XML_PROFILE_XPATH, OUT_XML_PROFILE_CHILD
BS_REASON_CODE	BS_REASON_CODE_PROFILE_D
REMARK_QUAL	REMARK_QUAL_TEXT, REMARK_QUAL_ASSET_ATTRIBUTE
REPORT	REPORT_PARAMETER, REPORT_CONTROL
SERVPROV	SERVPROV_ALIAS, SERVPROV_CB_PROFILE
MIGRATION_PROJECT	MIGRATION_OBJECT_GROUP, MIGRATION_OBJECT_GROUP_D

Refresh Cache

The Transportation and Global Trade Management Cloud application maintains a number of in-memory cache objects to improve performance. Historically, DBXML was not able to refresh these cache objects and so occasionally required a restart of the application to pick up some modifications.

When importing, the Refresh Cache flag is available to indicate that any cache objects associated with the imported data should be refreshed. For example, if the imported data contained a new workflow Agent which is 'active', this agent would automatically subscribe to its listening events and be available to be triggered by, for example, SHIPMENT – MODIFIED events.

Lifetime Events

Whenever certain objects are modified via the application, lifetime events are raised for CREATE, MODIFIED and REMOVED modifications. When the new Lifetime Events flag is used, these events will now also be raised for data imported via DBXML. For example, if a TRANSACTION_SET contains a new LOCATION, the LOCATION – CREATED event will be published.

Commit Scope

By default, each parent element in a TRANSACTION SET is treated as a separate database transaction i.e. if one failed others could potentially succeed.

There is now a new field for Commit Scope which defaults to the current behavior with the scope of 'PK'. There is a new value of 'SET' which indicates that all elements in the TRANSACTION SET must succeed or all will fail.

Custom Attributes

Certain DB XML elements support attributes not represented in the underlying table. These include:

Table	Element	Attribute	Use
GLUSER	GL_PASSWORD	PasswordType	If **TEXT**, the incoming password value is assumed to be text hashed on storage to the database. Otherwise, the value is assumed to be the hash, obtained fro export of the GL_USER record.

DB.XML User Interface

This chapter explains how to use the DB.XML user interface to export and import.

Exporting DB.XML

This section describes how to export DB.XML using the web-based user interface.

1. Log into Oracle Transportation Management.
2. Locate the DB XML Export user interface. By default this will be **Business Process Automation > Data Import/ Export > DB.XML Export**.
3. Choose an **Export Object Type**. Fields specific to the selected type will now be displayed.
    a. **DB Object**: This option is deprecated as it is not fully functional in Cloud environments and should not be used.
    b. **DB Object Set**: This option is deprecated as it is not fully functional in Cloud environments and should not be used.
    c. **Query**: The following fields are available:
        i. **SQL Query**: For example "select * from activity".
        ii. **Root Name**: the element name to be used for the parent XML element.
    d. **Migration Entity**: The following fields are available:
        i. **Migration Entity Name**: For example "Location."
        ii. **Object IDs**: For example "GUEST.MY_LOC_1, GUEST.MY_LOC_2".
4. Optionally, select to export Foot print columns or Large Objects as elements.
5. Click **Run**. Oracle Transportation Management displays the results page.

For example, the following shows an export with a SQL Query on the LOCATION table for LOCATION_GID = 'NYC'.

```
<dbxml:xml2sql xmlns:dbxml="https://xmlns.oracle.com/apps/otm/DBXML" Version="20A">
 <dbxml:TRANSACTION_SET>
 <LOCATION LOCATION_GID="NYC" LOCATION_XID="NYC" LOCATION_NAME="NEW YORK" CITY="NEW YORK"
 PROVINCE="NY" PROVINCE_CODE="NY" COUNTRY_CODE3_GID="USA" TIME_ZONE_GID="America/New_York"
```

```
LAT="40.75167" LON="-73.99417" IS_TEMPORARY="N" IS_MAKE_APPT_BEFORE_PLAN="N" DOMAIN_NAME="PUBLIC"
IS_SHIPPER_KNOWN="Y" IS_ADDRESS_VALID="Y" IS_LTL_SPLITABLE="Y" BB_IS_NEW_STORE="N"
EXCLUDE_FROM_ROUTE_EXECUTION="N" IS_TEMPLATE="N" APPT_OBJECT_TYPE="S" PRIMARY_ADDRESS_LINE_SEQ="1"
IS_ACTIVE="Y" ADDRESS_UPDATE_DATE="2013-08-13 14:14:31.0">
<LOCATION_ADDRESS LOCATION_GID="NYC" LINE_SEQUENCE="1" DOMAIN_NAME="PUBLIC"/>
<LOCATION_ADDRESS LOCATION_GID="NYC" LINE_SEQUENCE="2" DOMAIN_NAME="PUBLIC"/>
<LOCATION_CORPORATION LOCATION_GID="NYC" CORPORATION_GID="NYC" DOMAIN_NAME="PUBLIC"/>
<LOCATION_REFNUM LOCATION_GID="NYC" LOCATION_REFNUM_QUAL_GID="IA" LOCATION_REFNUM_VALUE="NYC"
DOMAIN_NAME="PUBLIC"/>
<LOCATION_ROLE_PROFILE LOCATION_GID="NYC" LOCATION_ROLE_GID="AIRPORT" DOMAIN_NAME="PUBLIC"
X_DOCK_IS_INBOUND_BIAS="N" CREATE_XDOCK_HANDLING_SHIPMENT="Y" CREATE_POOL_HANDLING_SHIPMENT="Y"
IS_ALLOW_MIXED_FREIGHT="N"/>
<LOCATION_ACTIVITY_TIME_DEF LOCATION_GID="NYC" LOCATION_ROLE_GID="AIRPORT" ACTIVITY_TIME_DEF_GID="388715"
DOMAIN_NAME="PUBLIC">
<ACTIVITY_TIME_DEF ACTIVITY_TIME_DEF_GID="388715" ACTIVITY_TIME_DEF_XID="388715"
FIXED_STOP_TIME="0" FIXED_STOP_TIME_UOM_CODE="NULL" FIXED_STOP_TIME_BASE="0" VARIABLE_STOP_TIME="0"
VARIABLE_STOP_TIME_UOM_CODE="NULL" VARIABLE_STOP_TIME_BASE="0" DOMAIN_NAME="PUBLIC"/>
</LOCATION_ACTIVITY_TIME_DEF>
</LOCATION>
</dbxml:TRANSACTION_SET>
</dbxml:xml2sql>
```

Note: Refer to the Oracle Transportation Management Data Dictionary for more information about what the objects can contain.

Note: Oracle Transportation Management does not display elements that are empty in the database.

Saving DB.XML Output to a File on Your PC

View the source for the frame containing the displayed XML using your browser and save as a file with the ".db.xml" file extension. The steps to view the source vary from browser to browser.

Note: Especially if your data contains non-ASCII characters, just save your file as-is and use an editor that supports UTF-8 when editing the file later on.

Importing DB.XML

This section describes how to import a DB.XML file using the web-based user interface.

1. Log into Oracle Transportation Management.
2. Locate the DB XML Import user interface. By default this will be **Business Process Automation > Data Import/ Export > DB.XML Import**.
3. Select the appropriate **Schema**. This defaults to GLOGOWNER.
4. Click **Browse** to specify the required **Input XML File** containing the transaction set to be uploaded.
5. The default **Transaction Code** is *I* (insert). You can change the Transaction Code from I to either II, IU, D or RC.
6. If the **Transaction Code** is *RC*, you may need to specify **Managed Tables**. This will be required when no pre-configured child table properties are setup for the parent table in the input XML file. If no managed tables are found, the RC transaction code is treated like an IU i.e. no child records will be removed.
7. Click **Run**.

Oracle Transportation Management displays summary statistics with a Success Count and an Error Count. The count is the number of transactions that were successful or in error.

DB XML Servlet

It may be convenient to export and import DB XML data remotely from the Transportation and Global Trade Management Cloud application. This can be achieved in a number of ways; by sending XML messages via HTTP POST to a servlet on the Transportation and Global Trade Management Cloud Web Server (discussed in this section) or as a SOAP message to a Web Service on the Transportation and Global Trade Management Cloud Application Server (discussed in the next section).

The HTTP POST body should use the format defined below and be sent via HTTP POST to the `glog.integration.servlet.DBXMLServlet`.

The servlet requires authentication using HTTP Basic Authentication. If the network used for communication cannot be assumed to be secure, the HTTPS protocol should be used.

Additionally, the URL **command** parameter should specify which DBXML command should be executed i.e. **xmlImport** for import and **xmlExport** for export. A complete example URL would therefore be:

```
https://<OTM URL>/GC3/glog.integration.servlet.DBXMLServlet?command=xmlExport
```

Note: It is necessary for the XML content to be specified as "text/xml". In curl this is done by adding the option: -H "Content-Type:text/xml".

Note: The at-sign “@” prefix on the filename specified with the -d option is required to inform curl that the parameter is a filename containing the XML text to be sent, rather than the literal text itself.

For example,

```
curl -k -u GUEST.ADMIN:PW -d @i_transmission.db.xml -o response.xml -H "Content-Type:text/xml" https://<OTM
 URL>/GC3/glog.integration.servlet.DBXMLServlet?command=xmlExport
```

Export Message Format

Query

The following is the format for the XML message to export XML based on a SQL query:

```
<?xml version="1.0" encoding="UTF-8"?>
<sql2xml>
 <Query>
 <RootName>{output XML tag name}</RootName>
 <Statement>{SQL select clause}</Statement>
 </Query>
 <FootPrint>{Y|N}</FootPrint>
 <UseLOBElement>{Y|N}</UseLOBElement>
</sql2xml>
```

For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<sql2xml>
 <Query>
 <RootName>Location</RootName>
 <Statement>SELECT * FROM LOCATION WHERE LOCATION_GID = 'GUEST.MY_LOC'</Statement>
```

```
 </Query>
</sql2xml>
```

Note: When using the Query option, you must account for any SQL special characters you intend to use in the Statement. Special character such as single quote (‘), ampersand (&), less than (<) and greater than (>), must be either escaped or encoded as appropriate.

Here are some examples on how to escape or encode these special characters:

- Single quote can be escaped by preceding the quote with another quote:
  - `<Statement>SELECT * FROM LOCATION WHERE LOCATION_NAME LIKE '%''S STORE'</Statement>`
- Single quote can also be encoded by using 2 `&apos;`:
  - `<Statement>SELECT * FROM LOCATION WHERE LOCATION_NAME LIKE '%&apos;&apos;S STORE'</Statement>`
- Ampersand should be encoded using `&`:
  - ```
    <Statement>SELECT * FROM LOCATION WHERE LOCATION_NAME LIKE 'JACK &
    JILL'</Statement>
    ```
- Less Than should be encoded using `&lt;`:
  - ```
    <Statement>SELECT * FROM LOCATION
    WHERE rownum < 10</Statement> -- used as an
    operator
    ```
  - ```
    <Statement>SELECT * FROM LOCATION
    WHERE LOCATION_NAME LIKE 'MIN < MAX'</Statement>
    -- used inside criteria string
    ```
- Greater Than should be encoded using `&gt;`:
  - ```
    <Statement>SELECT * FROM LOCATION
    WHERE PRIMARY_ADDRESS_LINE_SEQ > 3</Statement>
    -- used as an operator
    ```
  - ```
    <Statement>SELECT * FROM LOCATION
    WHERE LOCATION_NAME LIKE 'MAX > MIN'</Statement>
    -- used inside criteria string
    ```
- For the Not Equal To operator, <> should be encoded using `%lt;&gt;` or alternatively use `!=`:
  - ```
    <Statement>SELECT * FROM LOCATION
    WHERE PRIMARY_ADDRESS_LINE_SEQ <> 1</Statement>
    ```
  - ```
    <Statement>SELECT * FROM LOCATION
    WHERE PRIMARY_ADDRESS_LINE_SEQ != 1</Statement>
    ```

Migration Entity

The following is the format for the XML message to export based on a Migration Entity name:

```
<?xml version="1.0" encoding="UTF-8"?>
<sql2xml>
 <Entity>
 <Name>{entity name}</Name>
 <PK>{object PK}</PK>
 …more PK elements…
 </Entity>
 <FootPrint>{Y|N}</FootPrint>
 <UseLOBElement>{Y|N}</UseLOBElement>
```
```

```
</sql2xml>
```

For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<sql2xml>
 <Entity>
 <Name>Location</Name>
 <PK>GUEST.MY_LOC1</PK>
 <PK>GUEST.MY_LOC2</PK>
 </Entity>
</sql2xml>
```

The response XML will be the TRANSACTION_SET XML identical to that seen in the UI.

Import Message Format

The root element for the DB XML Import message is xml2sql and will contain the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<xml2sql Version="21C">
 <TransactionCode>{I|II|IU|D|RC}</TransactionCode>
 <SchemaOwner>{schema name}</SchemaOwner>
 <UpdateCache>{Y|N}</UpdateCache>
 <RaiseEvents>{Y|N}</RaiseEvents>
 <ManagedTables>
 <Table>{table name 1}</Table>
 <Table>{table name 2}</Table>
 </ManagedTables>
 <TRANSACTION_SET>
 <…table specific elements…>
 …
 </TRANSACTION_SET>
</xml2sql>
```

For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<xml2sql Version="21C">
 <TransactionCode>I</TransactionCode>
 <SchemaOwner>GLOGOWNER</SchemaOwner>
 <UpdateCache>Y</UpdateCache>
 <RaiseEvents>Y</RaiseEvents>
 <TRANSACTION_SET>
 <LOCATION LOCATION_GID='GUEST.MY_LOC'..etc>
 <LOCATION_CORPORATION …etc…/>
 <…etc… other child elements…/>
 </LOCATION>
 </TRANSACTION_SET>
</xml2sql>
```

The response XML will contain the counts for successful or error transactions.

```
<xml2sql>
 <SuccessCount>n</SuccessCount>
 <ErrorCount>m</ErrorCount>
 <ElapsedTime>p</ElapsedTime>
 <TimePerTransaction>q</TimePerTransaction>
</xml2sql>
```

Where `n`, `m`, `p` & `q` are integers and `p` & `q` are the number of milliseconds.

DB XML Web Service

DB XML Export and Import can also be performed by calling a SOAP Web Service running on the Application server. The WSDL for the service will be found under:

```
https://<server:port>/GC3Services/CommandService/call?WSDL
```

The server and port are specific to the host and port configured for the WebLogic server running Transportation and Global Trade Management Cloud (aka the Transportation and Global Trade Management Cloud application server).

The service is secured via Web Service Security in common with all other Transportation and Global Trade Management Cloud Web Services and so by default requires the WSS Username Token Profile over HTTPS for authentication.

The SOAP messages (defined in the WSDL) are essentially identical to the messages used for the DB XML Servlet but will be 'wrapped' in the corresponding command/operation name, that is:

```
<xmlExport>
 <sql2xml>
 <…elements as described previously>
 </sql2xml>
</xmlExport>
```

and

```
<xmlImport>
 <xml2sql>
 <…elements as described previously>
 </xml2sql>
</xmlImport>
```

Editing DB.XML Files

This section describes how you edit an exported DB.XML file before importing it again.

A Sample DB.XML File

An exported DB.XML file might look like this. Note that the content is wrapped in a pair of <TRANSACTION_SET> tags.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xml2sql Version=”21C”>
<TRANSACTION_SET>
<CORPORATION CORPORATION_GID="ACL" CORPORATION_XID="ACL" DOMAIN_NAME="PUBLIC" INSERT_DATE="2001-10-05
 19:03:37" INSERT_USER="DBA.ADMIN" IS_DOMAIN_MASTER="N" UPDATE_DATE="2001-10-06 12:43:46"
 UPDATE_USER="DBA.GLOGLOAD" dbObjectName="CORPORATION" />
</TRANSACTION_SET>
</xml2sql>
```

You can edit the values and add new objects.

When editing date and time values, be sure to keep the following format: YYYY-MM-DD HH:MM:SS.

If you miss an element in the exported file this is probably because Oracle Transportation Management does not export elements that are empty in the database. This means that you will have to add the tag to the DB.XML file yourself. Refer to the Oracle Transportation Management Data Dictionary for more information about what objects and tables exist.

Oracle Transportation Management ignores element names that do not correspond to the database table. This allows you to comment your DB.XML file without affecting what is imported.

As you edit the file, keep all element and attribute names in uppercase.

4 CSV

CSV Export

Exporting CSV Files via the Interface

CSV Export Screens

An initial screen prompts for certain information so the system can determine what additional information is required on subsequent screens.

Note: To avoid accidental open queries potentially filling up the disk space, CSV export sizes are limited to 512 MB.

Exporting Data as a Zip File

This section illustrates how to export a zip file containing one or more CSV files.

1. First, create a `csvutil.ctl` file containing the commands for exporting your files.

   A csvutil.ctl file may contain the following commands:

   ```
   -dataFileName activity_out.csv -command xcsv -tableName ACTIVITY
   -dataFileName location_out.csv -command xcsv -tableName LOCATION -whereClause "rownum < 10"
   ```

2. Next, create a zip file containing the csvutil.ctl file.
3. Once your zip file is created, you can upload the zip file as you would upload any other file to Oracle Transportation Management through the **Business Process Automation > Integration > Integration Manager > Upload an XML/CSV Transmission**.
4. Press **Save** to save the "results" zip file to your local workstation.
5. Open the zip file to see that the zip file contains two CSV files in this case, one corresponding to each command in the csvutil.ctl file. If the exported table(s) include a column of type CLOB, subdirectories named after the tables will exist in the zip file and will contain the individual external files for the CLOB values.
6. The zip file also contains a log file containing information regarding the execution of CSVUtil.

Exporting Large Zip Files in the Background

When exporting a large zip file, you may prefer to export it in the background to avoid the browser timing out. Here is a sample request zip file:

Here are the contents of the csvutil.ctl file within test2.bg.zip:

```
-dataFileName activity_out.csv -command xcsv -tableName ACTIVITY
-dataFileName location_out.csv -command xcsv -tableName LOCATION -whereClause "rownum < 10"
-mailTo customername@example.com -mailFrom customername@example.com -subject zipFileProcessDone -message
 hello -smtpHost mail.example.com
```

Here is another example csvutil.ctl file that exports all the rate_geo records in a given domain, along with all parent and child data, but not public data:

```
-dataFileName rate_geo_out.csv -command xcsvwpcd -tableName RATE_GEO -whereClause "domain_name = 'MDIETL'"
```

```
-mailTo customername@example.com -mailFrom customername@example.com -subject zipFileProcessDone -message
 hello -smtpHost mail.example.com
```

Here is the same example, but this time with referenced public data:

```
-dataFileName rate_geo_out.csv -excludePublic N -command xcsvwpcd -tableName RATE_GEO -whereClause
 "domain_name = 'MDIETL'"
-mailTo customername@example.com -mailFrom customername@example.com -subject zipFileProcessDone -message
 hello -smtpHost mail.example.com
```

Note: Exporting with parent and child data is a very time consuming process since the system has to repeatedly chase after foreign key references. Expect the export to run overnight for as long as 8 hours.

Running CSVUtil in the Background

CSVUtil supports running in the background. To do so:

1. Select what you want to export from the **Export Object Type** drop-down list.
2. Select where you want the data to be placed from the **Output Destination** drop-down list.
3. If you would like this process to run in the background, select **Y** from the **Run Job in Background** drop-down list.
4. Click **Run**. The next page is displayed. The fields on this page vary depending on your selections for the above fields.
5. If you selected "Y" for the **Run Job in Background** field, enter an **Email Address** and **SMTP Host** so you can be notified when the process is complete.

In this example, the following content was emailed:

```
<CSVUtil>
<Command>xcsv</Command>
<DataFileName>null</DataFileName>
<ExcludePublic>true</ExcludePublic>
<Write>
<Table>ACTIVITY</Table>
<WhereClause>null</WhereClause>
<DomainName>null</DomainName>
<Sql>null</Sql>
<![CDATA[
ACTIVITY
ACTIVITY_GID,ACTIVITY_XID,ACTIVITY_NAME,DOMAIN_NAME,INSERT_DATE,UPDATE_DATE,INSERT_USER,UPDATE_USER
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
"RECEIVE","RECEIVE","RECEIVING
 FREIGHT","PUBLIC","20011005190259","20021008201735","DBA.ADMIN","DBA.GLOGLOAD"
"LOAD","LOAD","LOADING FREIGHT","PUBLIC","20011005190259","20021008201735","DBA.ADMIN","DBA.GLOGLOAD"
"LIVELOAD","LIVELOAD","LIVE TRAILER
 LOADING","PUBLIC","20011005190259","20021008201735","DBA.ADMIN","DBA.GLOGLOAD"
"DISPATCH","DISPATCH","DRIVER
 DISPATCHING","PUBLIC","20011005190259","20021008201735","DBA.ADMIN","DBA.GLOGLOAD"
"ACTIVATE","ACTIVATE","ITINERARY
 ACTIVATED","PUBLIC","20011005190259","20021008201735","DBA.ADMIN","DBA.GLOGLOAD"
"PICKUP","PICKUP","WAREHOUSE PICKING","PUBLIC","20011005190259","20021008201735","DBA.ADMIN","DBA.GLOGLOAD"
"CLOSED","CLOSED","WAREHOUSE CLOSED
 DOOR","PUBLIC","20011005190259","20021008201735","DBA.ADMIN","DBA.GLOGLOAD"
"OFFICEHOURS","OFFICEHOURS","OFFICE
 HOURS","PUBLIC","20011005190259","20021008201735","DBA.ADMIN","DBA.GLOGLOAD"
"BATCH SORT","BATCH SORT","SORTATION AT
 DC","PUBLIC","20020125162107","20021008201735","DBA.GLOGLOAD","DBA.GLOGLOAD"
"BATCH DOCK LOAD","BATCH DOCK LOAD","DOCK LOAD AT
 DC","PUBLIC","20020125162107","20040308170536","DBA.GLOGLOAD","DBA.ADMIN"
"GUEST.BLAH","BLAH",,"GUEST","20030425012307","20031104125706","DBA.GLOGOWNER","DBA.ADMIN"
"RUSHHOURS","RUSHHOURS","RUSH HOURS","PUBLIC","20030717003037","20040308170536","DBA.ADMIN","DBA.ADMIN"
```

```
"GUEST.DLI1","DLI1","DLI1","GUEST","20030717144513",,"GUEST.DLI",
"GUEST.DLI2","DLI2","DLI2","GUEST","20030717144528",,"GUEST.DLI",
"GUEST.TEST","TEST","1","GUEST","20030728200219",,"GUEST.ADMIN",
"GUEST.ABCD","ABCD","VDSFDS","GUEST","20040605190045",,"GUEST.ADMIN",
"GUEST.DTB_SECOND_ACTIVITY","DTB_SECOND_ACTIVITY","DAWN'S SECOND
 ACTIVITY","GUEST","20040611120516",,"GUEST.ADMIN",
"GUEST.DTB_FIRST_ACTIVITY","DTB_FIRST_ACTIVITY","DAWN'S FIRST
 ACTIVITY","GUEST","20040611120313",,"GUEST.ADMIN",
"GUEST.DTB_NUMBER_3","DTB_NUMBER_3","NUMBER 3","GUEST","20040611121927",,"GUEST.ADMIN",
"ALL","ALL","ALL ACTIVITIES","PUBLIC","20040910173537","20041213180312","DBA.ADMIN","DBA.ADMIN"
"DEPOT","DEPOT","DEPOT","PUBLIC","20040910173537","20041213180312","DBA.ADMIN","DBA.ADMIN"
"OTHER","OTHER","OTHER ACTIVITIES","PUBLIC","20040921094353","20041213180312","DBA.ADMIN","DBA.ADMIN"
]]>
</Write>
</CSVUtil>
```

Normally, you use background processing when initiating lengthy jobs.

Exporting Referenced PUBLIC Data during Multi-table Exports

CSVUtil provides the ability to export referenced PUBLIC data during the multi-table export operations (xcsvwcd, xcsvwpd, xcsvwpcd). This feature is especially important when exporting data from a source database where the PUBLIC data has been modified.

Export Data Control Options

Exporting with Parent Data

To export a data record with its parent data, you can specify the xcsvwpd command. This command exports the specified records, along with all the referenced non-public foreign key records associated with the specified records. The generated CSV file is in multi-table format.

> **Note:** All the xcsvw* commands are far more expensive in terms of CPU usage than the plain xcsv command. Using them to export a large data set will take a long time, since many foreign keys must be found. Use the commands with a restrictive where-clause, as shown in the examples, to limit the running time.

Exporting with Child Data

To export a data record with its child data, you can specify the xcsvwcd command. This command exports the specified records, along with all the subordinate child table records corresponding to the selected records.

Exporting with both Parent and Child Data

To export a data record with both its parent and child data, you can specify the xcsvwpcd command. This command should be used with care since it can take a while to run.

CSV Import

CSVUtil Transaction Codes

When importing CSVUtil with any of the methods described in this document, the following transaction codes available:

- **I** (Insert): Only inserts are performed. If the data already exists in the database, you will get primary key errors.
- **IU** (Insert/Update): Attempts to insert data. If a primary key violation occurs, it updates the data. No delete statements are generated.
- **RC** (Replace Children): Deletes all child data corresponding to the top level parent, updates the top level parent, and reinserts the child data. This mode allows for a complete replacement of a data object. This command is available after you upload a CSV file via the integration manager.
- **II** (Insert/Ignore Duplicates): Attempts to insert a new record but only if it does not exist already.
- **D** (Delete): Deletes the object and all associated child data.

> **Note:** As this is essentially a "cascade delete" of an object this should only be used with particular care. For example, deleting a LOCATION may result in deleting SHIPMENTs which are related to that LOCATION.

CSVUtil supports a “replace children” (rc) command when processing multi-table CSV files. The rc command will recursively delete all child records and re-insert them from the CSV file. This is useful when you want to completely replace the rows that comprise a complex multi-table business object.

The “C.” table sets are used to determine the hierarchical parent/child relationships.

For example:

```
TABLE_SET_DETAIL
TABLE_SET,TABLE_NAME
C.GEO_HIERARCHY,GEO_HIERARCHY_DETAIL
C.GEO_HIERARCHY_DETAIL,HNAME_COMPONENT
```

The C.GEO_HIERARCHY table set indicates that the GEO_HIERARCHY_DETAIL table is a child of geo_hierarchy.

The C.GEO_HIERARCHY_DETAIL table set indicates that the HNAME_COMPONENT table is a child of geo_hierarchy_detail.

Examples:

If you submit the following multi-table CSV file with the “rc” command, all rows in the GEO_HIERARCHY_DETAIL table relating to the GUEST.COUNTRY hierarchy would be deleted (since there are none to replace those records in the CSV file).

```
$HEADER
GEO_HIERARCHY_DETAIL
GEO_HIERARCHY_GID,HNAME_COMPONENT_GID,HLEVEL,DOMAIN_NAME,INSERT_USER,INSERT_DATE,UPDATE_USER,UPDATE_DATE
GEO_HIERARCHY
GEO_HIERARCHY_GID,GEO_HIERARCHY_XID,RANK,COUNTRY_CODE3_GID,DOMAIN_NAME,INSERT_USER,INSERT_DATE,UPDATE_USER,UPDATE_DA
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS..'
$BODY
GEO_HIERARCHY
"GUEST.COUNTRY","COUNTRY",10,,"GUEST","DBA.ADMIN",2001-08-30 11:01:56.0,"DBA.ADMIN",2005-10-26 14:44:50.0
```

If you submit the following multi-table CSV file with the “rc” command, there will be two records in the geo_hierarchy_detail table relating to the GUEST.COUNTRY hierarchy, regardless of how many rows were there previously.

```
$HEADER
GEO_HIERARCHY_DETAIL
GEO_HIERARCHY_GID,HNAME_COMPONENT_GID,HLEVEL,DOMAIN_NAME,INSERT_USER,INSERT_DATE,UPDATE_USER,UPDATE_DATE
GEO_HIERARCHY
GEO_HIERARCHY_GID,GEO_HIERARCHY_XID,RANK,COUNTRY_CODE3_GID,DOMAIN_NAME,INSERT_USER,INSERT_DATE,UPDATE_USER,UPDATE_DA
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS..'
$BODY
GEO_HIERARCHY
"COUNTRY","COUNTRY",10,,"PUBLIC","DBA.ADMIN",2001-08-30 11:01:56.0,"DBA.ADMIN",2005-10-26 14:38:33.0
GEO_HIERARCHY_DETAIL
"COUNTRY","COUNTRY_CODE3",1,"PUBLIC","DBA.ADMIN",2001-08-30 11:01:56.0,,
GEO_HIERARCHY_DETAIL
"COUNTRY","CITY",2,"PUBLIC","DBA.ADMIN",2001-08-30 11:01:56.0,,
```

CLobs in CSV Files

CSVUtil supports inserting, updating, and deleting CLobs. You can:

- Include the CLob in the CSV file (each CLob<1Mb, no newline characters or quotes)
- In the importing CSV file, refer to an external file holding the CLob. (no size restrictions on the CLobs, newline characters allowed). Note that CLOB columns are always exported to external files.

Note: CSVUtil can only handle one CLob per record. Also, the DB column must be of type CLOB; this will not work if the column type is VARCHAR or VARCHAR2, no matter how large the field is allowed to be.

Here is a sample CSV file that inserts a CLob using the in-line method:

```
CLOB_TEST
SEQ,DESCR,XML
9,"LINE1",<test1>test1</test1>
10,"LINE2",<test2>test2</test2>
```

In this case, the "XML" column is of type CLob. When using the in-line method, each CLob:

- Must be specified on a single line (no newline characters).
- Must be smaller than 1 megabyte.

Here is a sample CSV file that inserts two CLobs using the external file method:

```
CLOB_TEST
SEQ,DESCR,EXT_FNAME,XML
11,"LINE1",myxmlfile.xml
12,"LINE2",myxmlfile2.xml
```

When using the external file method, you must specify a special "pseudo column" called "EXT_FNAME ". The EXT_FNAME pseudo column must be specified immediately to the left of the CLob column. In this case, you will have an extra column on line 2. So, in this case, line 2 has 4 columns, but there are only 3 columns in the data lines.

The external file method must be used when inserting CLobs containing newline characters, or when inserting CLobs greater than 1 megabyte.

Note: To include the external files with the main CSV file, the ZIP encapsulation (described in more detail below) must be used so all the files may delivered to the Oracle Transportation Management server in a single HTTP transfer.

GL_User Table

CSVUtil supports adding and deleting records in the GL_USER table. This table stores the Oracle Transportation Management users and their passwords.

When the GL_USER table is specified in the header of a CSV file, special processing is done.

If you are a Transportation and Global Trade Management Cloud user authorized to directly access the GL_USER table, you may add and delete records in the GL_USER table. As an exception for this table, you can only use the commands: i, ii, d, or dd.

Note: The u, uu, and iu commands are not supported when loading the GL_USER table.

Loading CSV Data

Loading CSV Data via the User Interface

This section describes how to import a CSV file using the Oracle Transportation Management user interface.

Note: There are limits on importing CSVs. When importing via the user interface, there is a 50MB limit. However, when importing via integration, there is a 100MB limit.

1. Sign in to Oracle Transportation Management.
2. Choose **Business Process Automation > Integration > Integration Manager**.
3. Click **Upload an XML / CSV Transmission**.
4. Select the file to upload. The upload will transfer files from your local machine to the server.

   Note: You must select a CSV file. This CSV file cannot make reference to external files for CLOB columns. If you need to use external CLOB files, use the zip file format described in the *Uploading a Zip File* and *CSV Export Screens* sections.
5. Click Upload and Oracle Transportation Management displays the page for importing the file. If you select a file other than a CSV file, a different page will open.
6. If it is not already selected, select *i* from the **command** list.
7. Leave the **dataDir** as is.
8. Leave the **dataFileName** as is.
9. The **xvalidate** drop-down list allows you to turn off verbose diagnostic messaging. To leave messaging on, the value in the field should be Y, which is the default.
10. In the encoding field, select the appropriate encoding type for your CSV file. If your file contains standard ASCII characters, then it can be encoded as ISO-8859-1. If it contains non-standard, international characters, then it should be encoded as UTF-8.
11. Click **Run** and Oracle Transportation Management displays a results page.

To read more about interpreting error messages, see the *Introduction to CSVUtil Response Messages* section.

Loading CSV Data via the Application Server

Introduction to Loading CSV Data via the Application Server

Oracle Transportation Management allows importing of CSV files via the application server. This feature is called "AppServer CSV" or AS.CSV.

If you upload a file whose name ends in "as.csv" instead of just ".CSV", it will be interpreted as an application server CSV file, instead of a database-centric CSV file. AppServer CSV files have the following features:

- The first line must be the name of an Entity such as Location, ObOrderBase, OrderRelease, etc. Entity names are derived from database table names, except they omit the underscores and use mixed case. For example, the entity name for the ob_order_base table is ObOrderBase.
- The second line must be a comma-separated list of attribute names. Attribute names are like database column names, except they omit the underscores and use mixed case. For example, a column called location_gid corresponds to the attribute **locationGid**. Note that the first character is in lower-case for attribute names, but upper case for entity names.
- The third line may be an optional UOM line, which provides UOM values for any UOM attributes. This line may be provided instead of providing UOM qualifiers every time a UOM value occurs.
- The remaining lines are data lines. Each value in a data line must correspond to an attribute name from line2.
- If a particular attribute is typically populated via an Oracle Database sequence, it can be specified in the attribute name list (second line described above), but the data lines can leave the value for that attribute empty (two commas next to each other without any other text between them).

Here's small sample file. This example omits the optional UOM line.

```
Location
locationGid,locationXid,countryCode3Gid,domainName,locationName
"GUEST.MYLOC8","MYLOC8","USA","GUEST","myloc8"
```

Here's another small sample file showing how to specify a UOM line.

```
SShipUnit
domainName,unitWidth,sShipUnitGid,isSplitable,unitNetVolume,unitNetWeight,shipUnitCount,unitWeight,unitVolume,unitHe
UOM:,,,,CUFT,LB,,LB,,,CUFT,LB,,
GUEST,,GUEST.001,false,0,10,1,10,,,0,0,,001
```

Here's the same sample, but with the UOM line omitted and the units of measure specified with each data attribute instead. (Note the use of "false" for the boolean isSplitable field).

```
SShipUnit
domainName,unitWidth,sShipUnitGid,isSplitable,unitNetVolume,unitNetWeight,shipUnitCount,unitWeight,unitVolume,unitHe
GUEST,,GUEST.001,false,0 CUFT,10 LB,1,10 LB,,,0 CUFT,0 LB,,001
```

Here's an example that will result in errors. You can't specify a UOM line of you also specify UOMs within the data attributes.

> **Note:** The example below represents what not to do. Do NOT copy the example below. The following example would produce an error because a UOM line was specified, but UOMs were also specified in the data attributes. Doing this would cause the system to think that each UOM field has two UOM qualifiers.

```
SShipUnit
```

```
domainName,unitWidth,sShipUnitGid,isSplitable,unitNetVolume,unitNetWeight,shipUnitCount,unitWeight,unitVolume,unitHe
UOM:,,,,CUFT,LB,,LB,,,CUFT,LB,,
GUEST,,GUEST.001,false,0 CUFT,10 LB,1,10 LB,,,0 CUFT,0 LB,,001
```

Web Interface for Importing

You can import and export app server CSV files via a web interface.

Importing CSV Data with the Integration Manager

If you use the Integration Manager to upload a CSV file whose name ends in ".as.csv", Oracle Transportation Management will assume that the content of the file adheres to the rules of AppServer CSV files, and will process it as such. An example of a file name would be "location.as.csv", as opposed to "location.csv".

Each row in the file is processed via the application server instead of directly against the database. This has the benefit of keeping the application server data-cache synchronized with the database.

This page is accessed via **Business Process Automation > Integration > Integration Manager**. See the *Loading CSV Data via User Interface* section for details about this page.

Errors encountered when importing are reported back to the screen.

Loading CSV Files as Zip Files

Uploading a Zip File

In addition to the CSV files, your zip file must include a control file called csvutil.ctl to tell Oracle Transportation Management how to process the files. The control file specifies the sequence in which the CSV files should be processed, and specifies the parameters to use when processing each file.

For example, this zip file contains the csvutil.ctl (control) file, and two CSV files, activity.csv and activity2.csv. The csvutil.ctl file contains the following command lines:

```
-dataFileName activity.csv -command i
-dataFileName activity2.csv -command i
```

The above control file says to process the file activity.csv using the insert command, then process the file activity2.csv, also using the insert command.

If you are performing an insert or update on a table with a CLOB column, you may specify the ext_fname pseudo-column before the CLOB column's name. Any external files referenced in the column should be filenames relative to a directory in the ZIP file with the same name as the table (except for being in lowercase).

Uploading a zip file is the same as uploading any other file. Use the "Upload an XML/CSV" Transmission button accessed via **Business Process Automation > Integration > Integration Manager**.

After uploading your zip file, once processing is complete, you are prompted to download a "results" zip file.

Click **Save** to save the "results" zip file to your local workstation.

The csvutil.log file in the "result" zip file contains the log from processing all the CSV files in the zip file that you uploaded. Additionally, if any of the -command lines specify one of the exporting commands (xcsv, xcsvwpd, xcsvwcd, xcsvwpcd), the result file will contain files named by the -dataFileName parameter containing the exported data. If the exported tables have a column of type CLOB, subdirectories named after the table will be in the ZIP file, containing the

individual external files for the CLOB fields in each record; the external file names in the .csv files will be relative to those directories.

Command Parameters for the csvutil.ctl File

The parameters in the table below can be specified on a line in the csvutil.ctl control file to indicate what database operation each line is to perform. Note that for database operations, the -command option must be specified; for import, update, and delete operations, the -dataFileName option is required. All other parameters are optional. Note that all the parameters for one operation must be on one line in the csvutil.ctl file; this file does not support line-wrapping.

CSVUtil Command and Arguments

Commands	Arguments
clobEncoding	The encoding of separate CLOB files you import. Only applicable if importing separate CLOB files. Common settings are ISO-8859-1 and UTF-8 (default). You especially need to consider this when you import data containing characters outside the 7-bit ASCII set. Also, consider the encoding of your database. If not specified, the value of the -encoding parameter is used for CLOB file imports.
command	Mandatory parameter, must specify one of the following commands: i - insert CSV data into the database ii - insert data, while suppressing unique key constraint violations iu - attempts to insert data. If a primary key violation occurs, it updates the data. No delete statements are generated. u - update data in the database uu - update data, while suppressing “no data found” constraint violations d- delete data from the database dd- delete data, while suppressing “no data found” constraint violations xcsv - export a CSV file xcsvcd - export a multi-table CSV file with all subordinate child tables (e.g. shipment_stop, shipment_stop_d etc. for the shipment table). A table set called C.<table_name> controls which tables are considered to be children of a given table. For example, the C.SHIPMENT table set contains the following tables: shipment_stop, shipment_refnum, shipment_remark, etc. Similarly, the C.SHIPMENT_STOP table_set contains the shipment_stop_d table. If you log in as DBA.ADMIN in Oracle Transportation Management, you can use the Table Set Manager to modify the contents of the various C.* table sets. xcsvpcd - export a multi-table CSV file with both parent and child data. xcsvpd - export a multi-table CSV file with all referenced non-public foreign key records (parent data) required to load the record(s) in a foreign database. xsql - export data as a script of SQL insert statements rather than a CSV file
dataFileName	The dataFileName argument specifies the name of the file in the dataDir directory to either read or write. This field is required when importing a file, but is optional when exporting a file. If unspecified for an export, the output is written to System.out, and the glog.database.admin.csvUtilScript.csvOutput property will affect the output representation if CSV-embedded-in-XML format is chosen.

Commands	Arguments
domain_name	The domain_name parameter only applies to the export commands (xcsv and xsql). It specifies that only the data in that domain is to be exported, and only applies if the tableName parameter is used to specify which data will be exported; if the domain_name parameter is specified, the whereClause parameter will be ignored.
encoding	The encoding of the file you import. Common settings are ISO-8859-1 (default) and UTF-8. You especially need to consider this when you import data containing characters outside the 7-bit ASCII set. Also, consider the encoding of your database.
excludeClobs	Do not export clobs from the specified tables. The value is a comma-separated list of table names. Specifying a table name that does not appear in the export operation will be silently ignored.
excludeStageTables	Do not export "staging" records created by the overall Transportation and Global Trade Management Cloud administrator's account DBA.ADMIN. Only applies when the -tableName and -domain_name parameters are used.
excludeBeforeDate	Do not export records whose original insertion date is prior to the specified date. Only applies when the -tableName and -domain_name parameters are used. By default, all records are exported (subject to other constraints) regardless of their initial creation date.
excludePublic	Can be either Y (default) or N. If Y, records from the PUBLIC domain will not be exported.
includeChildren	If using the tableName and domain_name parameters, this specifies that records in "child" domains (with their domain name prefixed with the parent domain and a "/") will also be included along with the parent domain records matching the domain_name parameter. For example, -domain_name PARENT -includeChildren would specify that records in the "PARENT", "PARENT/CHILD", and "PARENT/CHILD/GRANDCHILD" domains would be included in the export.
removeUndefinedColumns	CSVUtil supports, by default, the ability to ignore columns that are not defined in the target table. This is especially useful when exporting from a migrated database with deprecated columns, into a newly created database that does not have the deprecated columns. There is some performance impact for this feature. To deactivate the feature, use the following command line option: -removeUndefinedColumns N This option is only available when running CSVUtil directly on the command line. It is not available using either the web or ClientUtil.
selectList	Only applicable when tableName is specified. Specifies choosing only rows from the table whose ordinal sequence numbers match the row numbers stored in the SELECT_LIST_D table for the GID specified as the value of this parameter.
sqlQuery	If specified, then xcsv command is required and tableName, domain_name, and whereClause are ignored.
sqlFileName	Operates the same as sqlQuery, but reads the query statement from the specified file instead of directly from the command line. Useful to avoid issues with the command shell and special characters.
tableName	The tableName argument is only specified for the xcsv and xsql commands. This specifies the name of the database table to export. Can be null if sqlQuery is specified. Must be upper case.
tableSet	If used, command can only be one of the "x" export commands, and this iteratively does CSVUtil operations with tableName for each of the tables listed in the specified table set. All other parameter rules are as for using the tableName parameter.
whereClause	Only used when tableName is specified and domain_name is omitted.

Commands	Arguments
xvalidate	Can be either Y (default) or N. When set to Y, CSVUtil gives you more user-friendly diagnostics messages. When you set XValidate to Y, it also prevents values in the database from being wiped out if there are null values in the CSV. Setting the XValidate to N may wipe out values in the database or create an error if the CSV file is missing the value.

CSV Files that Failed to Load

If any of the records in any of your CSV files fail to load, then your “results” zip file will contain a corresponding “.bad” file containing those records that failed to load. For example, activity.csv.bad and activity2.csv.bad. In this case, both of the CSV files contained one of more records that failed to load, so there is a corresponding .bad file for each CSV file.

Background Zip File Processing

If you are uploading a large zip file, you may want to process your zip file in the background and be notified via email when processing completes. You can then pull your "results" zip file using the "ZipFileDownloadServlet".

Note: If you submit through the web UI a large request that takes more than a minute or so to process and do not background it, you will receive a timeout error from the load-balancing proxy in front of your Transportation and Global Trade Management Cloud application server. Your request may not have failed in this case; it may simply still be running but the proxy timed-out waiting for it to complete. However, if your control file does not contain the email notification command described below, you will never be informed as to where the results zip file is located when and if the request completes.

For example, this is a “request” zip file whose name specifies that background processing is desired: `test1.bg.zip`. Notice that the filename ends with “bg.zip” rather than just “.zip”. This naming convention indicates that background processing is desired. Here is a sample csvutil.ctl file that illustrates how to have an email sent out when processing completes:

```
-dataFileName activity.csv -command i
-dataFileName activity2.csv -command i
-mailTo customername@example.com -mailFrom customername@example.com -subject zipFileProcessDone -message
 Hello -smtpHost mail-server.example.com
```

Note: The last line does not contain the -command parameter, as it is not a database operation, but an email operation to notify you that the CSV operations are complete. It is strongly recommended to put the -mailTo parameter and its related parameters at the last line of the csvutil.ctl file, although they can also be placed earlier to report progress indications. The csvutil.ctl file is executed in the order of the lines within it, so a database operation listed after a -mailto command will not have yet been executed when the email is sent.

Note: Note that the -mailFrom parameter must match one of the registered Mail Sender addresses for your Oracle Transportation Management cloud server, and (if specified) the -smtpHost parameter value must match the value specified for the glog.mail.smtp.external.host property (if a 3third party email service is used instead of Oracle Cloud Infrastructure email); the -smtpHost parameter should not be specified if you are using Oracle Cloud Infrastructure email service.

Clicking on the link in the email takes you to a listing of the zip files on the web server.

You may click on the desired zip file to download it to your local workstation. The zip files ending in "result.zip" are the "results" or "output" zip files.

If things go wrong during background processing, your results zip file will contain error output, which you can read with a text editor rather than WinZip or other unzip tools.

Loading CSV Data via Integration

Introduction to Loading CSV Data via Integration

CSVDataLoad XML element provides the capability to embed the contents of a CSV file for insertion into the database as a Transmission XML document. Each CSVDataLoad element can contain only one CSV File. This element should only be used for small sets of data.

Transmission Document Hierarchy

Below you can see the XML document hierarchy. The elements have been indented to show the hierarchy and relationship.

```
<Transmission>
 <TransmissionHeader> . . .
 </TransmissionHeader>
 <TransmissionBody>
 <GLogXMLElement>
 <CSVDataLoad>
 ---CSV Data Load Contents---
 </CSVDataLoad>
 </GLogXMLElement>
 <GLogXMLElement>
 <CSVDataLoad>
 ---CSV Data Load Contents---
 </CSVDataLoad>
 </GLogXMLElement>
 </TransmissionBody>
 </Transmission>
```

Below is a sample document that would be used to insert some data into the rate tables:

```
<Transmission>
 <TransmissionHeader>
 <UserName>DBA.ADMIN</UserName>
 </TransmissionHeader>
 <TransmissionBody>
 <GLogXMLElement>
 <CSVDataLoad>
 <CsvCommand>iu</CsvCommand>
 <CsvTableName>X_LANE</CsvTableName>
 <CsvColumnList>X_LANE_GID,X_LANE_XID,SOURCE_POSTAL_CODE,SOURCE_COUNTRY_CODE3_GID,SOURCE_GEO_HIERARCHY_GID,DEST_POST
CsvColumnList>
 <CsvRow>"MYDOMAIN.194064","194064","194","USA","USZIP3","064","USA","USZIP3","MYDOMAIN"</CsvRow>
 <CsvRow>"MYDOMAIN.194065","194065","194","USA","USZIP3","065","USA","USZIP3","MYDOMAIN"</CsvRow>
 </CSVDataLoad>
 </GLogXMLElement>
 </TransmissionBody>
 </Transmission>
```

The CSV rows are also used to define any EXEC SQL commands as well as the actual rows of data. For example, if the rows contain data for columns which are dates, then an EXEC SQL statement must be present as the first <CsvRow> before any rows containing data. For example: `<CsvRow>EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT ='YYYYMMDDHH24MISS'</CsvRow>`

Note: the <CSVRow> line text bodies must not contain any reserved XML characters such as < or >. If they do, the characters must be escaped with `<` and `>`.

Loading Rate Data via CSV

Introduction to Loading Rate Data via CSV

This chapter gives you examples of:

- The tables you need to import to set up rates in Oracle Transportation Management.
- How to format the CSV files.
- The order in which you must import tables.

Refer to the Oracle Transportation Management Data Dictionary to learn what data you need and in what order you need to import it.

Note: Any blank columns are not included in the CSV files. See the Data Dictionary for a complete list of columns.

Importing Location Information

This section describes how to import location information in CSV format. A set of sample CSV files is presented. Tables must be loaded in the order presented in this section. Otherwise, foreign key violations occur.

1. Import the LOCATION Table.

   The following example illustrates how you specify LOCATION data in CSV format.

```
LOCATION
LOCATION_GID,LOCATION_XID,LOCATION_NAME,CITY,PROVINCE,PROVINCE_CODE,POSTAL_CODE,COUNTRY_CODE3_GID,TIME_ZONE_GID,
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.YELLOW,YELLOW,YELLOW LOCATION,PITTSBURGH,,PA,99999,USA,America/
New_York,,,N,N,,MYDOMAIN,N,U,N,,,,Y
MYDOMAIN.MYLOCATION,MYLOCATION,MYLOCATION,PHILADELPHIA,,PA,19001,USA,America/
New_York,40.12726,-75.12881,N,N,COMMERCIAL,MYDOMAIN,N,U,N,0,S,0,Y
MYDOMAIN.MYCORPORATION,MYCORPORATION,MYCORPORATION,PHILADELPHIA,,PA,19001,USA,America/
New_York,40.12726,-75.12881,N,N,COMMERCIAL,MYDOMAIN,N,U,N,0,S,0,Y
```

2. Import the LOCATION_ADDRESS table.

   The following example illustrates how you specify LOCATION_ADDRESS data in CSV format.

```
LOCATION_ADDRESS
LOCATION_GID,LINE_SEQUENCE,ADDRESS_LINE,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.YELLOW,1,432 YELLOW AVE,MYDOMAIN
MYDOMAIN.MYCORPORATION,1,11 EMPEROR AVE,MYDOMAIN
MYDOMAIN.MYLOCATION,1,123 MAPLE STREET,MYDOMAIN
MYDOMAIN.MYLOCATION,2,BUILDING H,MYDOMAIN
MYDOMAIN.MYLOCATION,3,ROOM 100,MYDOMAIN
```

3. Import the CORPORATION Table.

   The following example illustrates how you specify CORPORATION data in CSV format.

   > **Note:** Each CORPORATION_GID must correspond to a LOCATION_GID specified in the location table (See example).

```
CORPORATION
CORPORATION_GID,CORPORATION_XID,CORPORATION_NAME,DOMAIN_NAME,IS_DOMAIN_MASTER,IS_SHIPPING_AGENTS_ACTIVE,IS_ALLOW
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.MYCORPORATION,MYCORPORATION,MYCORP,MYDOMAIN,N,N,N
MYDOMAIN.YELLOW INC,YELLOW INC,YELLOW INCORPORATED,MYDOMAIN,N,N,N
```

4. Import the LOCATION_CORPORATION Table.

   The following example illustrates how you specify LOCATION_CORPORATION data in CSV format. This links a location to a corporation.

```
LOCATION_CORPORATION
LOCATION_GID,CORPORATION_GID,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.MYLOCATION,MYDOMAIN.MYCORPORATION,MYDOMAIN
MYDOMAIN.MYCORPORATION,MYDOMAIN.MYCORPORATION,MYDOMAIN
MYDOMAIN.YELLOW,MYDOMAIN.YELLOW INC,MYDOMAIN
```

5. Import the SERVPROV Table.

   The following example illustrates how you specify SERVPROV data in CSV format. Each SERVPROV_GID must correspond to a LOCATION_GID specified in the location table (See example).

```
SERVPROV
SERVPROV_GID,SERVPROV_XID,AUTO_PAYMENT_FLAG,DOMAIN_NAME,IS_DISPATCH_BY_REGION,ALLOW_TENDER,IS_ACCEPT_SPOT_BIDS,I
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.YELLOW,YELLOW,N,MYDOMAIN,N,Y,N,N,N,N,N,N,S
```

6. Import the LOCATION_ROLE_PROFILE Table.

   The following example illustrates how you specify LOCATION_ROLE_PROFILE data in CSV format. Each location should have at least one row in this table.

```
LOCATION_ROLE_PROFILE
LOCATION_GID,LOCATION_ROLE_GID,CALENDAR_GID,FIXED_HANDLING_TIME,FIXED_HANDLING_TIME_UOM_CODE,FIXED_HANDLING_TIME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.YELLOW,CARRIER,,0,S,0,N,N,MYDOMAIN
MYDOMAIN.MYLOCATION,SHIPFROM/SHIPTO,,0,S,0,N,N,MYDOMAIN
MYDOMAIN.MYCORPORATION,BILL TO,,0,S,0,N,N,MYDOMAIN
MYDOMAIN.MYCORPORATION,REMIT TO,,0,S,0,N,N,MYDOMAIN
```

7. Import the LOCATION_REMARK Table.

   The following example illustrates how you specify LOCATION_REMARK data in CSV format.

```
LOCATION_REMARK
LOCATION_GID,REMARK_SEQUENCE,REMARK_QUAL_GID,REMARK_TEXT,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.MYLOCATION,1,REM,DRIVER CANNOT HAVE A BEARD,MYDOMAIN
MYDOMAIN.MYLOCATION,2,REM,DRIVER MUST HAVE SAFETY GLASSES,MYDOMAIN
```

Importing Service Times

The following example illustrates how you specify SERVICE_TIME data in CSV format.

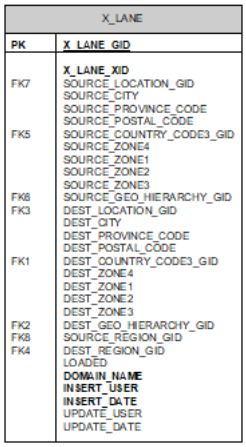
```
SERVICE_TIME
X_LANE_GID,RATE_SERVICE_GID,SERVICE_TIME_VALUE,SERVICE_DAYS,DOMAIN_NAME,SERVICE_TIME_VALUE_UOM_CODE,SERVICE_TIME_VAL
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.194-064,MYDOMAIN.VOYAGE-DEFAULT,172800,,MYDOMAIN,S,172800
MYDOMAIN.194-065,MYDOMAIN.VOYAGE-DEFAULT,86400,,MYDOMAIN,S,86400
```

In the above example, note that you must specify SERVICE_DAYS, and leave the SERVICE_TIME_VALUE unspecified. As an alternative, you must specify SERVICE_TIME_VALUE in seconds, and leave the SERVICE_DAYS unspecified. You must never specify both a SERVICE_TIME_VALUE and a SERVICE_DAYS value on the same record for the SERVICE_TIME table.

Importing X_LANE Data for Rates

This section provides an example for loading X_LANE data in CSV format. Typically, the X_LANE tables are loaded prior to the loading of the RATE_GEO and RATE_GEO_COST tables.

Simplified Entity Relationship Diagram (ERD) for X_LANE Data



The following example illustrates how you specify GEO_HIERARCHY and X_LANE data in CSV format.

```
GEO_HIERARCHY
GEO_HIERARCHY_GID,GEO_HIERARCHY_XID,RANK,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.USZIP4,USZIP4,4,MYDOMAIN

X_LANE
X_LANE_GID,X_LANE_XID,SOURCE_POSTAL_CODE,SOURCE_COUNTRY_CODE3_GID,SOURCE_GEO_HIERARCHY_GID,DEST_POSTAL_CODE,DEST_COU
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.194-064,194-064,194,USA,MYDOMAIN.USZIP4,64,USA,MYDOMAIN.USZIP4,MYDOMAIN
MYDOMAIN.194-065,194-065,194,USA,MYDOMAIN.USZIP4,65,USA,MYDOMAIN.USZIP4,MYDOMAIN
MYDOMAIN.MY LANE,MY LANE,194,,POSTAL_CODE,64,,POSTAL_CODE,MYDOMAIN
```

Importing LTL Rates

This section describes how to specify LTL rates and gives sample CSV files for several scenarios.

The following tables must be loaded (in order):

- RATE_OFFERING (setup manually on Oracle Transportation Management web pages)
- X_LANE (see the *Importing X_LANE Data for Rates* section.)
- RATE_GEO
- ACCESSORIAL_CODE

- ACCESSORIAL_COST
- RATE_GEO_ACCESSORIAL (*)
- RATE_GEO_COST_GROUP
- RATE_GEO_COST
- RATE_UNIT_BREAK_PROFILE
- RATE_UNIT_BREAK
- RATE_GEO_COST_UNIT_BREAK

> **Note:** (*) RATE_GEO_ ACCESSORIAL must come after RATE_GEO, but is not required before the remaining tables.

Assumptions:

- You have loaded the rate offering table using Oracle Transportation Management web pages
- You have loaded the X_Lane table (see the *Importing X_LANE Data for Rates* section.)

Simplified ERD for LTL Rates

Simplified Entity Relationship Diagram (ERD) for LTL Rate Data

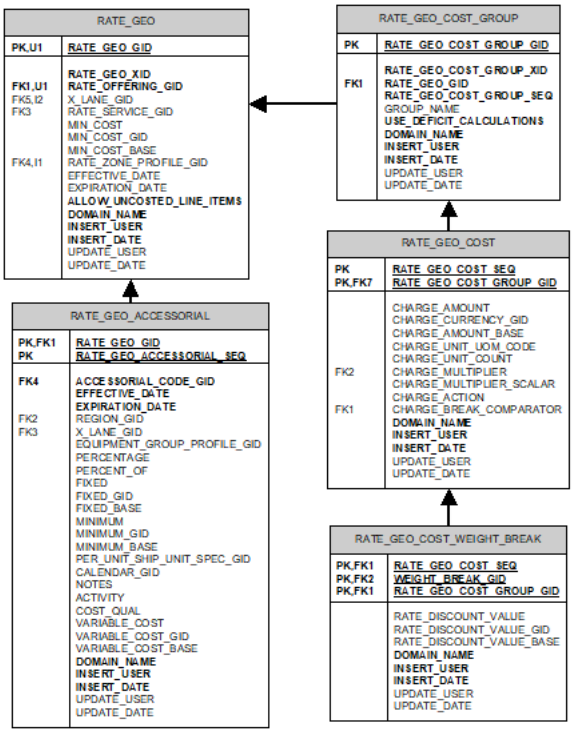


Table Notes:

- RATE_GEO Table
  Allow_uncosted_line_items in Y/N (defaults to “N”)
- RATE_GEO_ACCESSORIAL
  - Left_Operand1 – Basis options define what variable you want to base your conditional charge on.
  - Oper1_gid – The operand you compare with.

- Low_value1 – Depending on the operand you use, you might need only the low_value1 or additionally the high_value1.

- RATE_GEO_COST_GROUP Table

  Use_deficit_calculations in Y/N (defaults to “N”)

- RATE_GEO_COST Table
  - charge_unit_uom_code - unit of measure (e.g. “LB” for pounds, or “MI” for miles)
  - charge_unit_count - hundredweight, etc.
  - charge_action – add (A), setmin (M), setmax (X), multiply/discount (D)
  - charge_break_comparator -identifies data element used to access the break

Scenario 1: Based on Simple Unit Breaks

This first scenario assumes that rates are defined as simple unit breaks.

1. Import RATE_GEO table.

```
RATE_GEO
RATE_GEO_GID,RATE_GEO_XID,RATE_OFFERING_GID,MIN_COST,MIN_COST_GID,MIN_COST_BASE,X_LANE_GID,DOMAIN_NAME
"MYDOMAIN.194-064","194-064","MYDOMAIN.YELLOW",1.0,"USD",1.0,"MYDOMAIN.194-064","MYDOMAIN"
"MYDOMAIN.194-065","194-065","MYDOMAIN.YELLOW",1.0,"USD",1.0,"MYDOMAIN.194-065","MYDOMAIN"
```

2. Import RATE_GEO_COST_GROUP table.

```
RATE_GEO_COST_GROUP
RATE_GEO_COST_GROUP_GID,RATE_GEO_COST_GROUP_XID,RATE_GEO_GID,RATE_GEO_COST_GROUP_SEQ,GROUP_NAME,DOMAIN_NAME
"MYDOMAIN.194-064","194-064","MYDOMAIN.194-064",1,"MY_GROUP_NAME","MYDOMAIN"
"MYDOMAIN.194-065","194-065","MYDOMAIN.194-065",1,"MY_GROUP_NAME","MYDOMAIN"
```

3. Import RATE_GEO_COST table.

```
RATE_GEO_COST
RATE_GEO_COST_SEQ,RATE_GEO_COST_GROUP_GID,CHARGE_UNIT_UOM_CODE,CHARGE_UNIT_COUNT,CHARGE_BREAK_COMPARATOR,DOMAIN_
1,"MYDOMAIN.194-064","LB",100,"SHIPMENT.WEIGHT","MYDOMAIN"
1,"MYDOMAIN.194-065","LB",100,"SHIPMENT.WEIGHT","MYDOMAIN"
```

4. Import RATE_UNIT_BREAK_PROFILE table.

```
RATE_UNIT_BREAK_PROFILE
RATE_UNIT_BREAK_PROFILE_GID,RATE_UNIT_BREAK_PROFILE_XID,DATA_TYPE,LOOKUP_TYPE,UOM_TYPE,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.LT 1000,LT 1000,U,M,WEIGHT,MYDOMAIN
```

5. Import RATE_UNIT_BREAK table.

```
RATE_UNIT_BREAK
RATE_UNIT_BREAK_GID,RATE_UNIT_BREAK_XID,RATE_UNIT_BREAK_PROFILE_GID,RATE_UNIT_BREAK_MAX,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.1000,0-1000,MYDOMAIN.LT 1000,1000 LB,MYDOMAIN
MYDOMAIN.1000-3000,1000-3000,MYDOMAIN.LT 1000,3000 LB,MYDOMAIN
```

6. Import RATE_GEO_COST_UNIT_BREAK table.

```
RATE_GEO_COST_UNIT_BREAK
RATE_GEO_COST_GROUP_GID,RATE_GEO_COST_SEQ,RATE_UNIT_BREAK_GID,CHARGE_AMOUNT,CHARGE_AMOUNT_GID,CHARGE_AMOUNT_BASE
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.194-064,1,MYDOMAIN.1000,48.53,USD,48.53,MYDOMAIN
MYDOMAIN.194-064,1,MYDOMAIN.1000-3000,37.56,USD,37.56,MYDOMAIN
```

Scenario 2: Based on Cost per Pound, Surcharge, and Discount

This scenario assumes that:

- Freight cost is $0.07 per lb

- Fuel Surcharge is 3% of Total Cost (Accessorial)
- Discount is 65% of Total Cost
- There is a $50 allowance for loading
- The minimum charge is based on 10,000 lb
- Total Cost = (weight * 0.07 – 50.00) * (65% Discount) * (Accessorial Surcharge of 3%)
- Min Cost = (10,000 * 0.07 – 50.00) * (1 - 0.65) * (1.03) = 234.325

1. Import RATE_GEO table.

```
RATE_GEO
RATE_GEO_GID,RATE_GEO_XID,RATE_OFFERING_GID,MIN_COST,MIN_COST_GID,MIN_COST_BASE,X_LANE_GID,DOMAIN_NAME
"MYDOMAIN.194-064-2","194-064-2","MYDOMAIN.YELLOW",234.325,"USD",234.325,"MYDOMAIN.194-064","MYDOMAIN"
```

2. Import ACCESSORIAL_COST table.

```
ACCESSORIAL_COST
ACCESSORIAL_COST_GID,ACCESSORIAL_COST_XID,CHARGE_MULTIPLIER,CHARGE_MULTIPLIER_SCALAR,CHARGE_ACTION,CHARGE_TYPE,U
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FS,FS,SHIPMENT.COSTS.AMOUNT,1.03,A,B,N,A,N,MYDOMAIN,N
```

3. Import ACCESSORIAL_CODE table.

```
ACCESSORIAL_CODE
ACCESSORIAL_CODE_GID,ACCESSORIAL_CODE_XID,ACCESSORIAL_DESC,APPLY_GLOBALLY,DOMAIN_NAME,IS_FLOW_THRU,IS_VAT_EXEMPT
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FUEL_SURCHARGE,FUEL_SURCHARGE,FUEL SURCHARGE,Y,MYDOMAIN,N,N
```

4. Import RATE_GEO_ACCESSORIAL table.

```
RATE_GEO_ACCESSORIAL
ACCESSORIAL_COST_GID,RATE_GEO_GID,ACCESSORIAL_CODE_GID,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FS,MYDOMAIN.194-064-2,MYDOMAIN.FUEL_SURCHARGE,MYDOMAIN
```

5. Import RATE_GEO_COST_GROUP table.

```
RATE_GEO_COST_GROUP
RATE_GEO_COST_GROUP_GID,RATE_GEO_COST_GROUP_XID,RATE_GEO_GID,RATE_GEO_COST_GROUP_SEQ,GROUP_NAME,DOMAIN_NAME
MYDOMAIN.194-064-2,194-064-2,MYDOMAIN.194-064-2,1,MY_GROUP_NAME_2,MYDOMAIN
```

6. Import RATE_GEO_COST table.

```
RATE_GEO_COST
RATE_GEO_COST_SEQ,RATE_GEO_COST_GROUP_GID,CHARGE_AMOUNT,CHARGE_CURRENCY_GID,CHARGE_AMOUNT_BASE,CHARGE_UNIT_UOM_C
1,"MYDOMAIN.194-064-2",0.07,"USD",0.07,"LB",1,"SHIPMENT.WEIGHT",,"A","MYDOMAIN"
2,"MYDOMAIN.194-064-2",-50.0,"USD",-50.0,,1,,,"A","MYDOMAIN"
3,"MYDOMAIN.194-064-2",,,,,1,,0.35,"D","MYDOMAIN"
```

> **Note:** An alternative to using the data specified for the RATE_GEO_ACCESSORIAL table above would be to add another Sequence to this table with the following (representing a 3% surcharge of the total value):

```
4,"MYDOMAIN.194-064-2",,,,,1,,1.03,"D","MYDOMAIN"
```

Scenario 3: Based on Cost per Pound, Conditional Surcharge, Global Surcharge, and Discount

This scenario assumes that:

- Freight cost is $0.07 per lb
- Unload fee is $10 if the weight > 20000lb (Accessorial)
- Fuel Surcharge is 3% of Total Cost (Accessorial)
- Discount is 65% of Total Cost
- There is a $50 allowance for loading
- The minimum charge is based on 10,000 lb

Summary

- Total Cost = ((weight * 0.07 – 50.00) * (65% Discount) + (if weight>20000lb then Accessorial Surcharge of 10)) * (1.03)
- Min Cost = (10,000 * 0.07 – 50.00) * (1 - 0.65) * (1.03) = 234.325

1. Import RATE_GEO table.

```
RATE_GEO
RATE_GEO_GID,RATE_GEO_XID,RATE_OFFERING_GID,MIN_COST,MIN_COST_GID,MIN_COST_BASE,X_LANE_GID,DOMAIN_NAME
MYDOMAIN.194-064-3,194-064-3,MYDOMAIN.YELLOW,234.325,USD,234.325,MYDOMAIN.194-064,MYDOMAIN
```

2. Import ACCESSORIAL_COST table.

```
ACCESSORIAL_COST
ACCESSORIAL_COST_GID,ACCESSORIAL_COST_XID,LEFT_OPERAND1,OPER1_GID,LOW_VALUE1,AND_OR1,LEFT_OPERAND2,OPER2_GID,LOW
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FS,FS,,,,,,,,SHIPMENT.COSTS.AMOUNT,,,,,1.03,A,B,N,A,N,MYDOMAIN,N,0,A,N
MYDOMAIN.FS-2,FS-2,SHIPMENT.STOPS.SHIPUNITS.ACTIVITY,EQ,D,S,SHIPMENT.STOPS.WEIGHT,GT,20000
 LB,SHIPMENT,10,USD,10,1,,A,B,N,A,N,MYDOMAIN,,,,N
```

3. Import ACCESSORIAL_CODE table.

```
ACCESSORIAL_CODE
ACCESSORIAL_CODE_GID,ACCESSORIAL_CODE_XID,ACCESSORIAL_DESC,APPLY_GLOBALLY,DOMAIN_NAME,IS_FLOW_THRU,IS_VAT_EXEMPT
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FUEL_SURCHARGE,FUEL_SURCHARGE,FUEL SURCHARGE,Y,MYDOMAIN,N,N
```

4. Import RATE_GEO_ACCESSORIAL table.

```
RATE_GEO_ACCESSORIAL
ACCESSORIAL_COST_GID,RATE_GEO_GID,ACCESSORIAL_CODE_GID,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FS-2,MYDOMAIN.194-064-3,MYDOMAIN.FUEL_SURCHARGE,MYDOMAIN
MYDOMAIN.FS,MYDOMAIN.194-064-3,MYDOMAIN.FUEL_SURCHARGE,MYDOMAIN
```

5. Import RATE_GEO_COST_GROUP table.

```
RATE_GEO_COST_GROUP
RATE_GEO_COST_GROUP_GID,RATE_GEO_COST_GROUP_XID,RATE_GEO_GID,RATE_GEO_COST_GROUP_SEQ,GROUP_NAME,DOMAIN_NAME
MYDOMAIN.194-064-3,194-064-3,MYDOMAIN.194-064-3,1,MY_GROUP_NAME_3,MYDOMAIN
```

6. Import RATE_GEO_COST table.

```
RATE_GEO_COST
RATE_GEO_COST_SEQ,RATE_GEO_COST_GROUP_GID,CHARGE_AMOUNT,CHARGE_CURRENCY_GID,CHARGE_AMOUNT_BASE,CHARGE_UNIT_UOM_C
1,MYDOMAIN.194-064-3,0.07,USD,0.07,LB,1,SHIPMENT.WEIGHT,,A,MYDOMAIN
2,MYDOMAIN.194-064-3,-50,USD,-50,,1,,,A,MYDOMAIN
3,MYDOMAIN.194-064-3,,,,,1,,65,D,MYDOMAIN
```

Importing TL Rates

This section describes how to specify TL rates and gives sample CSV files for several scenarios.

The following tables must be loaded (in order):

- RATE_OFFERING (setup manually on Oracle Transportation Management web pages)
- X_LANE (see the *Importing X_LANE Data for Rates* section.)
- RATE_GEO
- ACCESSORIAL_CODE
- ACCESSORIAL_COST
- RATE_GEO_ACCESSORIAL (*)
- RATE_GEO_STOPS (*)

- RATE_GEO_COST_GROUP
- RATE_GEO_COST

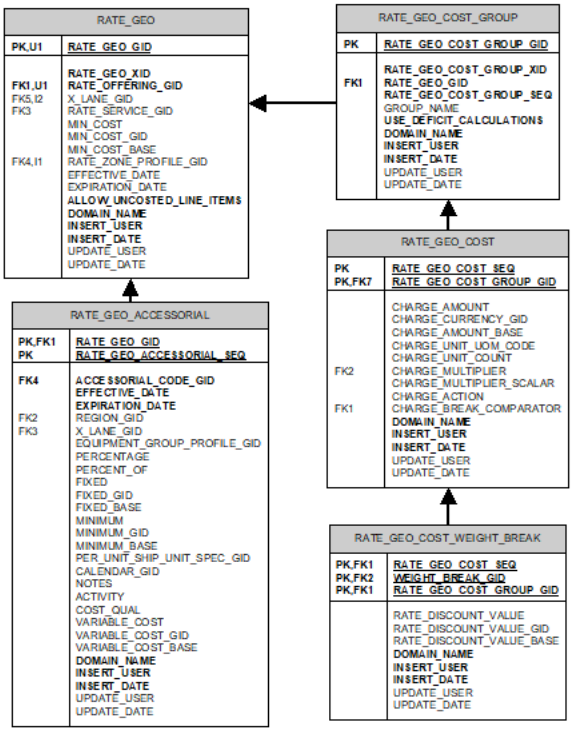
> **Note:** (*) RATE_GEO_ ACCESSORIAL and RATE_GEO_STOPS must come after RATE_GEO, but are not required before the remaining tables.

Assumptions:

- You have loaded the rate offering table using Oracle Transportation Management web pages
- You have loaded the X_Lane table (see the *Importing X_LANE Data for Rates* section).

Simplified ERD for Truckload Rates

Simplified Entity Relationship Diagram (ERD) for Truckload Rate Data



RATE_GEO Table

- Allow_uncosted_line_items in Y/N (defaults to “N”)

RATE_GEO_ACCESSORIAL

- Left_Operand1 – Basis options define what variable you want to base your conditional charge on.
- Oper1_gid – The operand you compare with.
- Low_value1 – Depending on the operand you use, you might need only the low_value1 or additionally the high_value1.

RATE_GEO_COST_GROUP Table

- Use_deficit_calculations in Y/N (defaults to “N”)

RATE_GEO_COST Table

- Oper1_gid – field value “BETWEEN” is a shortcut for X > low and X <= high. Other possible values include “<”, “<=”, “>”, “>=”, “=”, and “<>”.
- charge_unit_uom_code - unit of measure (e.g. “LB” for pounds, or “MI” for miles)
- charge_unit_count - hundredweight, etc.
- charge_action – add (A), setmin (M), setmax (X), multiply (D)
- charge_break_comparator -identifies data element used to access the break

Scenario 1: Based on Distance Bands with Fixed Charges and Stop Offs

This scenario assumes that:

- TL rates are defined using distance bands, with a flat charge within each band
- For Rate Geo A:
  - If distance between 10 and 100 miles, charge $50
  - If distance is between 100 and 200 miles, charge $75
- For Rate Geo B:
  - If distance between 10 and 100 miles, charge $80

1. Import RATE_GEO table.

```
RATE_GEO
RATE_GEO_GID,RATE_GEO_XID,RATE_OFFERING_GID,MIN_COST,MIN_COST_GID,MIN_COST_BASE,X_LANE_GID,TOTAL_STOPS_CONSTRAIN
MYDOMAIN.194-064-TL1,194-064-TL1,MYDOMAIN.YELLOW,1,USD,1,MYDOMAIN.194-064,6,2,MYDOMAIN
MYDOMAIN.194-065-TL1,194-065-TL1,MYDOMAIN.YELLOW,1,USD,1,MYDOMAIN.194-065,6,2,MYDOMAIN
```

2. Import RATE_GEO_STOPS table.

```
RATE_GEO_STOPS
RATE_GEO_GID,LOW_STOP,HIGH_STOP,PER_STOP_COST,PER_STOP_COST_GID,PER_STOP_COST_BASE,DOMAIN_NAME
"MYDOMAIN.194-064-TL1",1,2,50.00,"USD",50.00,"MYDOMAIN"
"MYDOMAIN.194-064-TL1",3,4,100.00,"USD",100.00,"MYDOMAIN"
"MYDOMAIN.194-065-TL1",1,2,25.50,"USD",25.50,"MYDOMAIN"
"MYDOMAIN.194-065-TL1",3,4,85.00,"USD",85.00,"MYDOMAIN"
```

3. Import RATE_GEO_COST_GROUP table.

```
RATE_GEO_COST_GROUP
RATE_GEO_COST_GROUP_GID,RATE_GEO_COST_GROUP_XID,RATE_GEO_GID,RATE_GEO_COST_GROUP_SEQ,GROUP_NAME,DOMAIN_NAME
"MYDOMAIN.194-064-TL1","194-064-TL1","MYDOMAIN.194-064-TL1",1,"MY_GROUP_NAME_TL","MYDOMAIN"
"MYDOMAIN.194-065-TL1","194-065-TL1","MYDOMAIN.194-065-TL1",1,"MY_GROUP_NAME_TL","MYDOMAIN"
```

4. Import RATE_GEO_COST table.

```
RATE_GEO_COST
RATE_GEO_COST_SEQ,RATE_GEO_COST_GROUP_GID,OPER1_GID,LEFT_OPERAND1,LOW_VALUE1,HIGH_VALUE1,CHARGE_AMOUNT,CHARGE_CU
1,"MYDOMAIN.194-064-TL1","BETWEEN","SHIPMENT.DISTANCE","10 MI","100 MI",50.00,"USD", 50.00,"MYDOMAIN"
2,"MYDOMAIN.194-064-TL1","BETWEEN","SHIPMENT.DISTANCE","100 MI","200 MI",75.00,"USD", 75.00,"MYDOMAIN"
1,"MYDOMAIN.194-065-TL1","BETWEEN","SHIPMENT.DISTANCE","10 MI","100 MI",80.00,"USD", 80.00,"MYDOMAIN"
```

Scenario 2: Based on Cost per Mile, Stop Offs, and Surcharges

This scenario assumes that:

- The freight cost is $1.75 per mile
- Stop Off Charges
  - Allowed 6 stops total, with 2 stops included in rate
  - Charge of $50 for 3rd stop, and $65 for subsequent stops

- Fuel Surcharge is $0.02 per mile (Accessorial)
- Minimum charge on transport is $450

Summary

- Total Cost = (distance * 1.75) + stop off charges + (Accessorial of $0.02 per mile)
- Min Transport = (450.00) + stop off charges + (Accessorial of $0.02 per mile)

1. Import RATE_GEO table.

```
RATE_GEO
RATE_GEO_GID,RATE_GEO_XID,RATE_OFFERING_GID,MIN_COST,MIN_COST_GID,MIN_COST_BASE,X_LANE_GID,TOTAL_STOPS_CONSTRAIN
"MYDOMAIN.194-064-TL2","194-064-TL2","MYDOMAIN.YELLOW",1.0,"USD",1.0,"MYDOMAIN.194-064",6, 2,"MYDOMAIN"
```

2. Import ACCESSORIAL_COST table.

```
ACCESSORIAL_COST
ACCESSORIAL_COST_GID,ACCESSORIAL_COST_XID,CHARGE_MULTIPLIER,CHARGE_AMOUNT,CHARGE_AMOUNT_GID,CHARGE_AMOUNT_BASE,C
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FS-TL2,FS-TL2,SHIPMENT.DISTANCE,0.02,USD,0.02,MI,1,A,B,N,A,N,MYDOMAIN,N
```

3. Import ACCESSORIAL_CODE table.

```
ACCESSORIAL_CODE
ACCESSORIAL_CODE_GID,ACCESSORIAL_CODE_XID,ACCESSORIAL_DESC,APPLY_GLOBALLY,DOMAIN_NAME,IS_FLOW_THRU,IS_VAT_EXEMPT
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FUEL_SURCHARGE,FUEL_SURCHARGE,FUEL SURCHARGE,Y,MYDOMAIN,N,N
```

4. Import RATE_GEO_ACCESSORIAL table.

```
RATE_GEO_ACCESSORIAL
ACCESSORIAL_COST_GID,RATE_GEO_GID,ACCESSORIAL_CODE_GID,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FS-TL2,MYDOMAIN.194-064-TL2,MYDOMAIN.FUEL_SURCHARGE,MYDOMAIN
```

5. Import RATE_GEO_STOPS table.

```
RATE_GEO_STOPS
RATE_GEO_GID,LOW_STOP,HIGH_STOP,PER_STOP_COST,PER_STOP_COST_GID,PER_STOP_COST_BASE,DOMAIN_NAME
MYDOMAIN.194-064-TL2,1,1,50,USD,50,MYDOMAIN
MYDOMAIN.194-064-TL2,2,,65,USD,65,MYDOMAIN
```

> **Note:** Leaving the HIGH_STOP value empty indicates that the last charge will be applied to all the stops greater than the LOW_STOP value. (i.e. for stops >= 2, charge $65 per stop).

6. Import RATE_GEO_COST_GROUP table.

```
RATE_GEO_COST_GROUP
RATE_GEO_COST_GROUP_GID,RATE_GEO_COST_GROUP_XID,RATE_GEO_GID,RATE_GEO_COST_GROUP_SEQ,GROUP_NAME,DOMAIN_NAME
MYDOMAIN.194-064-TL2,194-064-TL2,MYDOMAIN.194-064-TL2,1,MY_GROUP_NAME_TL2,MYDOMAIN
```

7. Import RATE_GEO_COST table.

```
RATE_GEO_COST
RATE_GEO_COST_SEQ,RATE_GEO_COST_GROUP_GID,CHARGE_AMOUNT,CHARGE_CURRENCY_GID,CHARGE_AMOUNT_BASE,CHARGE_UNIT_UOM_C
1,MYDOMAIN.194-064-TL2,1.75,USD,1.75,MI,1,SHIPMENT.DISTANCE,,A,MYDOMAIN
2,MYDOMAIN.194-064-TL2,450,USD,450,,1,,,M,MYDOMAIN
```

> **Note:** Seq#2, with a charge action of “M”, indicates that the minimum of the running calculated cost has to be $450 (i.e. if the calculation from Seq#1 is less than $450, then the new value to be used going forward is $450).

An alternative method of specifying this rate would be to recognize that a minimum of $450 equates to distance of 257.143 miles. A comparison for this distance could be used. This would be the corresponding result.

```
RATE_GEO_COST
```

```
RATE_GEO_COST_SEQ,RATE_GEO_COST_GROUP_GID,OPER1_GID,LEFT_OPERAND1,LOW_VALUE1,CHARGE_AMOUNT,CHARGE_CURRENCY_GID,CHARG
1,"MYDOMAIN.194-064-TL2",">","SHIPMENT.DISTANCE","237.143 MI",1.750,"USD",1.750,"MI",1,
 "SHIPMENT.DISTANCE",,"A","MYDOMAIN"
2,"MYDOMAIN.194-064-TL2","<=","SHIPMENT.DISTANCE","257.143 MI",450.0,"USD",450.0,,1, ,,"A","MYDOMAIN"
```

Note: An alternative to using the data specified for the RATE_GEO_ACCESSORIAL table above would be to add another Sequence to the RATE_GEO_COST table with the following (representing a surcharge of $0.02 per mile):

```
3,"MYDOMAIN.194-064-TL2",0.020,"USD",0.020,"MI",1,"SHIPMENT.DISTANCE",,"A","MYDOMAIN"
```

Scenario 3: Based on Cost per Hundredweight, Unit Breaks, and Surcharges

This scenario assumes that:

- The freight cost is per hundredweight based on unit breaks
- Fuel Surcharge is $0.02 per mile (Accessorial)

Summary

Total Cost = ((weight/100) * (weight break charge)) + (Accessorial of $0.02 per mile)

1. Import RATE_GEO table.

```
RATE_GEO
RATE_GEO_GID,RATE_GEO_XID,RATE_OFFERING_GID,MIN_COST,MIN_COST_GID,MIN_COST_BASE,X_LANE_GID,TOTAL_STOPS_CONSTRAIN
MYDOMAIN.194-064-TL3,194-064-TL3,MYDOMAIN.YELLOW,1,USD,1,MYDOMAIN.194-064,6,2,MYDOMAIN
```

2. Import ACCESSORIAL_COST table.

```
ACCESSORIAL_COST
ACCESSORIAL_COST_GID,ACCESSORIAL_COST_XID,CHARGE_MULTIPLIER,CHARGE_AMOUNT,CHARGE_AMOUNT_GID,CHARGE_AMOUNT_BASE,C
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FS-TL3,FS-TL3,SHIPMENT.DISTANCE,0.02,USD,0.02,MI,1,A,B,N,A,N,MYDOMAIN,N
```

3. Import ACCESSORIAL_CODE table.

```
ACCESSORIAL_CODE
ACCESSORIAL_CODE_GID,ACCESSORIAL_CODE_XID,ACCESSORIAL_DESC,APPLY_GLOBALLY,DOMAIN_NAME,IS_FLOW_THRU,IS_VAT_EXEMPT
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FUEL_SURCHARGE,FUEL_SURCHARGE,FUEL SURCHARGE,Y,MYDOMAIN,N,N
```

4. Import RATE_GEO_ACCESSORIAL table.

```
RATE_GEO_ACCESSORIAL
ACCESSORIAL_COST_GID,RATE_GEO_GID,ACCESSORIAL_CODE_GID,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.FS-TL3,MYDOMAIN.194-064-TL3,MYDOMAIN.FUEL_SURCHARGE,MYDOMAIN
```

5. Import RATE_GEO_COST_GROUP table.

```
RATE_GEO_COST_GROUP
RATE_GEO_COST_GROUP_GID,RATE_GEO_COST_GROUP_XID,RATE_GEO_GID,RATE_GEO_COST_GROUP_SEQ,GROUP_NAME,DOMAIN_NAME
MYDOMAIN.194-064-TL3,194-064-TL3,MYDOMAIN.194-064-TL3,1,MY_GROUP_NAME_TL3,MYDOMAIN
```

6. Import RATE_GEO_COST table.

```
RATE_GEO_COST
RATE_GEO_COST_SEQ,RATE_GEO_COST_GROUP_GID,CHARGE_AMOUNT,CHARGE_CURRENCY_GID,CHARGE_AMOUNT_BASE,CHARGE_UNIT_UOM_C
1,MYDOMAIN.194-064-TL3,,,,LB,100,SHIPMENT.WEIGHT,,A,SHIPMENT.WEIGHT,MYDOMAIN
```

   Note: An alternative to using the data specified for the RATE_GEO_ACCESSORIAL table above would be to add another Sequence to this table with the following (representing a surcharge of $0.02 per mile):

```
2,"MYDOMAIN.194-064-TL3",0.020,"USD",0.020,"MI",1,"SHIPMENT.DISTANCE",,"A","MYDOMAIN"
```

7. Import RATE_UNIT_BREAK_PROFILE table.

```
RATE_UNIT_BREAK_PROFILE
RATE_UNIT_BREAK_PROFILE_GID,RATE_UNIT_BREAK_PROFILE_XID,RATE_UNIT_BREAK_PROFILE_NAME,DATA_TYPE,LOOKUP_TYPE,UOM_T
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
"MYDOMAIN.TL 40 TO 45 THOU","TL 40 TO 45 THOU","TL 40 TO 45
 THOU","U","M","WEIGHT","MYDOMAIN","MYDOMAIN.ADMIN","20060821190229",,
```

8. Import RATE_UNIT_BREAK table.

```
RATE_UNIT_BREAK
RATE_UNIT_BREAK_GID,RATE_UNIT_BREAK_XID,RATE_UNIT_BREAK_PROFILE_GID,RATE_UNIT_BREAK_MAX,DOMAIN_NAME,INSERT_USER,
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
"MYDOMAIN.40000","40000","MYDOMAIN.TL 40 TO 45 THOU","40000
 LB","MYDOMAIN","MYDOMAIN.ADMIN","20060821190229",,
"MYDOMAIN.45000","45000","MYDOMAIN.TL 40 TO 45 THOU","45000
 LB","MYDOMAIN","MYDOMAIN.ADMIN","20060821190229",,
```

9. Import RATE_GEO_COST_UNIT_BREAK table.

```
RATE_GEO_COST_UNIT_BREAK
RATE_GEO_COST_GROUP_GID,RATE_GEO_COST_SEQ,RATE_UNIT_BREAK_GID,CHARGE_AMOUNT,CHARGE_AMOUNT_GID,CHARGE_AMOUNT_BASE
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.194-064-TL3,1,MYDOMAIN.40000,1.14,USD,1.14,MYDOMAIN
MYDOMAIN.194-064-TL3,1,MYDOMAIN.45000,1.07,USD,1.07,MYDOMAIN
```

Scenario 4: Based on Cost per Hundredweight, Unit Breaks, and Surcharges

This scenario assumes that:

- The freight cost is per hundredweight based on unit breaks which are based on mileage bands.

Cost per Weight

	Cost per Weight	
Mileage Band	40000 lbs	45000 lbs
0 – 50	0.85	0.50
51 – 55	0.87	0.82
56 - 60	0.88	0.83

- Weighing charge is $20
- Vacuuming fee is $0.25 per CWT with a $115 minimum

Summary

Total Cost = ((weight/100) * (unit break charge)) + $20 + (Vacuuming Fee of 0.25 per CWT)

Note: Min $115 for vacuuming is reached when the weight is at 46,000 lbs

1. Import RATE_GEO table.

```
RATE_GEO
RATE_GEO_GID,RATE_GEO_XID,RATE_OFFERING_GID,MIN_COST,MIN_COST_GID,MIN_COST_BASE,X_LANE_GID,TOTAL_STOPS_CONSTRAIN
MYDOMAIN.194-064-TL4,194-064-TL4,MYDOMAIN.YELLOW,1,USD,1,MYDOMAIN.194-064,6,2,MYDOMAIN
```

2. Import RATE_GEO_COST_GROUP table.

```
RATE_GEO_COST_GROUP
RATE_GEO_COST_GROUP_GID,RATE_GEO_COST_GROUP_XID,RATE_GEO_GID,RATE_GEO_COST_GROUP_SEQ,GROUP_NAME,DOMAIN_NAME
```

```
MYDOMAIN.194-064-TL4,194-064-TL4,MYDOMAIN.194-064-TL4,1,MY_GROUP_NAME_TL4,MYDOMAIN
```

3. Import RATE_GEO_COST table.

```
RATE_GEO_COST
RATE_GEO_COST_SEQ,DOMAIN_NAME,RATE_GEO_COST_GROUP_GID,OPER1_GID,LEFT_OPERAND1,LOW_VALUE1,CHARGE_AMOUNT,CHARGE_CU
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
1,MYDOMAIN,MYDOMAIN.194-064-TL4,LT,SHIPMENT.WEIGHT,45000
 LB,,,LB,1,SHIPMENT.WEIGHT,,A,SHIPMENT.WEIGHT,B,A,Y,,,,N
2,MYDOMAIN,MYDOMAIN.194-064-TL4,GE,SHIPMENT.WEIGHT,45000
 LB,,,LB,1,SHIPMENT.WEIGHT,,A,SHIPMENT.WEIGHT,B,A,Y,,,,N
3,MYDOMAIN,MYDOMAIN.194-064-TL4,,,,20,USD,,1,SHIPMENT,,A,,B,A,N,N,0,A,Y
```

4. Import RATE_UNIT_BREAK_PROFILE table.

```
RATE_UNIT_BREAK_PROFILE
RATE_UNIT_BREAK_PROFILE_GID,RATE_UNIT_BREAK_PROFILE_XID,DATA_TYPE,LOOKUP_TYPE,UOM_TYPE,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.LESS THAN 40 PDS,LESS THAN 40 PDS,U,M,WEIGHT,MYDOMAIN
MYDOMAIN.GREATER THAN 45000 PDS,GREATER THAN 45000 PDS,U,M,WEIGHT,MYDOMAIN
```

5. Import RATE_UNIT_BREAK table.

```
RATE_UNIT_BREAK
RATE_UNIT_BREAK_GID,RATE_UNIT_BREAK_XID,RATE_UNIT_BREAK_PROFILE_GID,RATE_UNIT_BREAK_MAX,DOMAIN_NAME
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.0-50 MILES,0-50 MILES,MYDOMAIN.GREATER THAN 45000 PDS,45000 LB,MYDOMAIN
MYDOMAIN.51-55 MILES,51-55 MILES,MYDOMAIN.GREATER THAN 45000 PDS,45000 LB,MYDOMAIN
MYDOMAIN.56-60 MILES,56-60 MILES,MYDOMAIN.GREATER THAN 45000 PDS,45000 LB,MYDOMAIN
MYDOMAIN.0-50,0-50,MYDOMAIN.LESS THAN 40 PDS,44999 LB,MYDOMAIN
MYDOMAIN.51-55,51-55,MYDOMAIN.LESS THAN 40 PDS,44999 LB,MYDOMAIN
MYDOMAIN.56-60,56-60,MYDOMAIN.LESS THAN 40 PDS,44999 LB,MYDOMAIN
```

6. Import RATE_GEO_COST_UNIT_BREAK table.

```
RATE_GEO_COST_UNIT_BREAK
RATE_GEO_COST_GROUP_GID,RATE_GEO_COST_SEQ,RATE_UNIT_BREAK_GID,CHARGE_AMOUNT,CHARGE_AMOUNT_GID,CHARGE_AMOUNT_BASE
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.194-064-TL4,1,MYDOMAIN.0-50,0.85,USD,0.85,,MYDOMAIN
MYDOMAIN.194-064-TL4,2,MYDOMAIN.51-55,0.87,USD,0.87,,MYDOMAIN
MYDOMAIN.194-064-TL4,3,MYDOMAIN.56-60,0.88,USD,0.88,,MYDOMAIN
MYDOMAIN.194-064-TL4,1,MYDOMAIN.0-50 MILES,0.5,USD,0.5,,MYDOMAIN
MYDOMAIN.194-064-TL4,2,MYDOMAIN.51-55 MILES,0.82,USD,0.82,,MYDOMAIN
MYDOMAIN.194-064-TL4,3,MYDOMAIN.56-60 MILES,0.83,USD,0.83,,MYDOMAIN
```

Exporting/Importing Rates between Instances using Zip Files

CSVUtil can be used to copy a rate offering, along with all of its prerequisite parent and child data from one database to another.

1. **Create a csvutil.ctl File for Exporting**:

   You create a CSVUtil control file containing commands, and then place it in a zip file whose name ends with .bg.zip; for example: exp_rate_offering.bg.zip. When the zip file name ends with “bg.zip”, it knows to run the export job in the background. Here are the contents of the csvutil.ctl file to export an entire rate offering:

```
-dataFileName rate_geo_out.csv -command xcsvwpcd -tableName RATE_GEO -whereClause "rate_offering_gid =
 'MDIETL.ASDF'" -excludePublic N
-mailTo customername@example.com -mailFrom customername@example.com -subject zipFileProcessDone -message
 hello -smtpHost mail.example.com
```

> **Note:** There may only be two lines of text in the above example.

- Place the csvutil.ctl file in a zip file called *name*.bg.zip, where *name* can be anything.
- The xcsvwpcd (export CSV with parent and child data) command will export the rate_geo records, and will recursively export all parent and child records. This can take a while (up to 8 hours).
- The -excludePublic N option means that referenced PUBLIC data will also be exported. If you are sure that your target database has all the required public data, then you can change this to Y, which will save some time on the export.

2. **Upload the Zip File Created in Step 1**:
   Use the Integration Upload Screen to upload the exp_rate_offering.bg.zip file. In response to your upload, you immediately receive a message indicating that your export job has been submitted to run in the background. You receive an email when the job completes. The email includes an HTML link to allow you to download the resultant zip file containing your multi-table export.
3. **Download the Rate Offering Zip File**:
   When you receive the email, download the zip file containing the rate offering and extract the rate_geo_out.csv file.
4. **Create a csvutil.ctl File for Importing**:
   Similar to step 1, you create another csvutil.ctl file for importing in the background. For example:

```
-dataFileName rate_geo_out.csv -command ii
-mailTo customername@example.com -mailFrom customername@example.com -subject zipFileProcessDone -message
 hello -smtpHost mail.example.com
```

5. **Create Another Background Zip File**:
   Now create another zip file which will contain the csvutil.ctl file from the previous step, as well as the rate_geo_out.csv file which was exported during step 2. The zip file should again end with “bg.zip”.
6. **Upload the Zip File from Step 5 to the Target**:
   To import to the target instance, again use the integration upload screen to upload the background zip file to target instance. You again receive a response indicating that you will get an email when the job completes. The email will again contain a link to allow you to download a results zip file which contains a log file. You will need to examine the log file to see how the import did.

   > **Note:** If you are exporting from a migrated database to a fresh database, use the -removeUndefinedColumns option.

   This will tell CSVUtil to ignore deprecated columns.

CSVUtil Response Messages

Introduction to CSVUtil Response Messages

At the completion of processing the command, CSVUtil responds in the form of an XML message. The XML message may contain the following elements:

- Information passed in as input parameters such as the Command, DataDir, and DataFileName.

- Information about the contents of the input file such as the TableName and ColumnList.
- An Error element identifying the error that was detected.
- Statistics on the success of the message as follows:
  - **ProcessCount**: The number of rows that were successfully processed
  - **ErrorCount**: The number of rows where an error was detected
  - **Skipcount**: The number of rows that were skipped because of duplicate or missing keys. This is only valid when using the ii command which suppresses unique key constraint violations when inserting data, or the uu and dd commands which suppress "no data found" constraint violations when updating/deleting data.

Response Messages with No Errors

Here is an example of a response indicating no errors. In this case, three data rows (based on the ProcessCount element) of the weight_break.csv file were successfully inserted.

```
<CSVUtil>
<Command>i</Command>
<DataFileName>weight_break.csv</DataFileName>
<ProcessCSV>
<TableName>WEIGHT_BREAK</TableName>
<ColumnList>WEIGHT_BREAK_GID,WEIGHT_BREAK_XID,WEIGHT_BREAK_PROFILE_GID,WEIGHT_BREAK_MAX,WEIGHT_BREAK_MAX_UOM_CODE,WE
ColumnList>
<ProcessCount>3</ProcessCount>
<ErrorCount>0</ErrorCount>
<SkipCount>0</SkipCount>
</ProcessCSV>
</CSVUtil>
```

The following is an example of the response message typically received when exporting data using the xcsv command.

```
<CSVUtil>
<Command>xcsv</Command>
<DataFileName>weight_break.csv</DataFileName>
<Write>
<TableName>WEIGHT_BREAK</TableName>
</Write>
</CSVUtil>
```

Error Messages

After processing a command, CSVUtil displays a response in the form of an XML message (see the *Introduction to Loading CSV Data via the Application Server* section). When an error is detected in the processing, the XML message will contain an Error element with the details. The Error XML element indicates the table name, indicates the type of error detected, and lists the data (or row in file) that was being processed when the error occurred.

Below is the error message that Oracle Transportation Management displayed in the procedure (see the *Introduction to Loading CSV Data via Integration* section). The TableName element indicates the table being processed, the Exception element provides the error message, and the Data element indicates the row being processed. In this case, it indicates that the JUNK table does not exist in the database.

```
<Error>
<TableName>Sample</TableName>
<Exception>ORA-00942: table or view does not exist
```

```
</Exception>
<Data>"Data1","Data2","Data3"</Data>
</Error>
```

Import Messages

This topic describes some common error messages while importing. For each error, there is an explanation of when the message occurs and the action needed to correct the error.

Message 1

```
<Exception> ORA-00942: table or view does not exist
```

- **Occurs When**: Table name improperly specified (misspelled or invalid table) on the first line of the CSV file.
- **Corrective Action**: Verify that the table exists and that the CSV file contains the correct table name.

Message 2

```
<Exception> ORA-00001: unique constraint (GLOGOWNER.PK_WEIGHT_BREAK) violated
```

- **Occurs When**: Inserting data with primary keys that are already in the database.
- **Corrective Action**: Depending on the action desired, one of the following can be used:
  - If the data should be skipped or ignored, use the ii command to suppress the message.
  - If the data is intended to be new, change the keys.
  - If the data is intended to be an update, use the u or uu command.

Message 3

```
<Exception>ORA-02292: integrity constraint (GLOGOWNER.FK_RGCWB_WEIGHT_BREAK_GID) violated - child record
 found
```

- **Occurs When**: During a delete when child records in other tables depend on the key being removed.
- **Corrective Action**: Delete child records in associated tables before deleting from this table, or use the RC command code in CSVUtil if deleting all associated child records is your intent.

Message 4

```
<Error>There are supposed to be 7 columns of data, but I found 6 columns in this line: ["MYDOMAIN.LT
 4500","LT 4500","MYDOMAIN.DEFAULT",4500,"LB","MYDOMAIN"]</Error>
<Error>
<TableName>WEIGHT_BREAK</TableName>
<Exception>ORA-01722: invalid number</Exception>
<Data>"MYDOMAIN.LT 4500","LT 4500","MYDOMAIN.DEFAULT",4500,"LB","MYDOMAIN"</Data>
</Error>
```

- **Occurs When**: Missing a column of data in one of the rows.
- **Corrective Action**: Verify that the data contains the number of fields as indicated in the ColumnList, and that the field formats (string, numeric, date, etc.) are valid.

Message 5

```
<Error>
<TableName>WEIGHT_BREAK</TableName>
<Exception>ORA-01722: invalid number</Exception>
<Data>"MYDOMAIN.LT 4500","LT 4500","MYDOMAIN.DEFAULT",4500,"LB","gung ho","MYDOMAIN"</Data>
```

```
</Error>
```

- **Occurs When**: Trying to insert a string in a numeric field.
- **Corrective Action**: Verify that the data contains the number of fields as indicated in the ColumnList, and that the field formats (string, numeric, date, etc.) are valid.

Message 6

```
<Error>There are supposed to be 7 columns of data, but I found 1 columns in this line: []</Error>
<Error>
<TableName>WEIGHT_BREAK</TableName>
<Exception>ORA-01400: cannot insert NULL into ("GLOGOWNER"."WEIGHT_BREAK"."WEIGHT_BREAK_GID")
</Exception>
<Data></Data>
</Error>
```

- **Occurs When**: The CSV file contains extra blank lines at the end. Oracle Transportation Management considers each line to represent a row of data, regardless of whether or not it is blank.
- **Corrective Action**: Remove any extra blank lines at the end of the file.

Message 7

```
<Error>
<TableName>WEIGHT_BREAK</TableName>
<Exception>ORA-01401: inserted value too large for column</Exception>
<Data>"MYDOMAIN.LT 4500","LT
 4500","MYDOMAIN.DEFAULT",4500,"LB",4500,"MYDOMAIN12345678901234567890123456789012345678901234567890123456789012345
Data>
</Error>
```

- **Occurs When**: Field length for one of the columns has been exceeded.
- **Corrective Action**: Limit the length of the input data field value to the appropriate size.

Message 8

```
<Error>
<TableName>WEIGHT_BREAK</TableName>
<RowsProcssed>0</RowsProcssed>
<Data>"MYDOMAIN.LT 4500","LT 4500","MYDOMAIN.DEFAULT",4500,"LB",4500,"MYDOMAIN"</Data>
</Error>
```

- **Occurs When**: Attempted to delete data where the data does not exist in the table.
- **Corrective Action**: Validate that the keys being used to delete the data are correct. Could use the dd command to suppress the error message.

Export Messages

This topic describes some common error messages while exporting. For each error there is an explanation of when the message occurs and the action needed to correct the error.

Message

```
<CSVUtil>
<Command>xcsv</Command>
<DataDir>.\</DataDir>
<DataFileName>weight_break.csv</DataFileName>
<Write>
<TableName>WEIGHT_BREAK2</TableName>
```

```
</Write>
</CSVUtil>
Caught exception: CSVUtil.SQLException: /CSVUtil.SQLException:
(null)/java.sql.SQLException: ORA-00936: missing expression
…
```

- **Occurs When**: Attempting to export data from a table that does not exist.
- **Corrective Action**: Verify table exists and that the CSV file contains the correct table name.