

Oracle® Transportation Management

Data Management Guide

Release 6.5.3

Part No. F82872-01

September 2023

Copyright Notice

Oracle® Transportation Management Data Management Guide, Release 6.5.3

Part No. F82872-01

Copyright © 2005, 2023, Oracle and/or its affiliates.

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

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

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

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

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

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

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

Documentation Accessibility

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

Access to Oracle Support

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

Contents

COPYRIGHT NOTICE	III
ORACLE CUSTOMERS THAT HAVE PURCHASED SUPPORT HAVE ACCESS TO ELECTRONIC SUPPORT THROUGH MY ORACLE SUPPORT. FOR INFORMATION, VISIT HTTP://WWW.ORACLE.COM/PLS/TOPIC/LOOKUP?CTX=ACC&ID=INFO OR VISIT HTTP://WWW.ORACLE.COM/PLS/TOPIC/LOOKUP?CTX=ACC&ID=TRS IF YOU ARE HEARING IMPAIRED. CONTENTS	IV
FIGURES	VIII
TABLES	VIII
SEND US YOUR COMMENTS	X
PREFACE	XI
1. INTRODUCTION	1-1
DB.XML	1-1
WHY DO I WANT TO USE DB.XML?.....	1-1
HOW CAN I USE DB.XML?	1-1
CSV	1-2
A SAMPLE CSV FILE	1-2
MULTI-TABLE CSV FILES.....	1-2
INTERNATIONAL CHARACTERS	1-3
IMPORT	1-3
EXPORT	1-4
BEST PRACTICES	1-4
MIGRATION PROJECT	1-4
2. DB.XML	2-1
DB XML EXPORT	2-1
USING AN OBJECT NAME	2-1
USING AN OBJECT SET NAME.....	2-3
SPECIFYING A SQL QUERY	2-3
MIGRATION ENTITY	2-3
DB XML IMPORT	2-4
EXECUTION MODE.....	2-4
TRANSACTION CODE	2-4
REFRESH CACHE.....	2-6
LIFETIME EVENTS.....	2-6
COMMIT SCOPE	2-6
DB.XML USER INTERFACE	2-6

EXPORTING DB.XML	2-6
IMPORTING DB.XML	2-7
DB.XML COMMAND LINE EXECUTION	2-8
EXPORTING DB.XML	2-9
IMPORTING DB.XML	2-10
DB XML SERVLET.....	2-11
EXPORT MESSAGE FORMAT	2-11
IMPORT MESSAGE FORMAT	2-13
DB XML WEB SERVICE.....	2-14
EDITING DB.XML FILES	2-15
A SAMPLE DB.XML FILE.....	2-15
3. LOADING CSV DATA VIA THE COMMAND LINE.....	3-1
IMPORTING AND EXPORTING ON THE SERVER SIDE.....	3-1
CLOBS IN CSV FILES.....	3-7
EXPORTING WITH PARENT DATA	3-8
EXPORTING WITH CHILD DATA	3-8
EXPORTING WITH BOTH PARENT AND CHILD DATA	3-8
GL_USER TABLE	3-9
4. LOADING CSV DATA VIA WEB PAGES	4-1
IMPORTING	4-1
5. LOADING RATE DATA VIA CSV.....	5-1
IMPORTING LOCATION INFORMATION	5-1
IMPORTING SERVICE TIMES	5-3
IMPORTING X_LANE DATA FOR RATES	5-3
IMPORTING LTL RATES.....	5-4
SIMPLIFIED ERD FOR LTL RATES	5-5
SCENARIO-BASED ON SIMPLE UNIT BREAKS	5-6
SCENARIO-BASED ON COST PER POUND, SURCHARGE, AND DISCOUNT	5-7
SCENARIO-BASED ON COST PER POUND, CONDITIONAL SURCHARGE, GLOBAL SURCHARGE, AND DISCOUNT	5-8
IMPORTING TL RATES.....	5-9
SIMPLIFIED ERD FOR TL RATES	5-11
SCENARIO-BASED ON DISTANCE BANDS WITH FIXED CHARGES, AND STOP OFFS	5-12
SCENARIO-BASED ON COST PER MILE, STOP OFFS, AND SURCHARGES	5-13
SCENARIO-BASED ON COST PER HUNDREDWEIGHT, UNIT BREAKS, AND SURCHARGES	5-15
SCENARIO-BASED ON COST PER HUNDREDWEIGHT, UNIT BREAKS, AND SURCHARGES	5-16
6. LOADING CSV DATA VIA THE APPLICATION SERVER.....	6-1
WEB INTERFACE FOR IMPORTING AND EXPORTING APPSERVER CSV FILES	6-2
IMPORTING	6-2
EXPORTING.....	6-2
LOAD CSV FILES IN THE REPORT OWNER DIRECTORY.....	6-2
7. LOADING CSV DATA VIA INTEGRATION	7-1

GLOGXML DOCUMENT HIERARCHY	7-1
8. LOADING CSV FILES AS ZIP FILES	8-1
UPLOADING A ZIP FILE	8-1
CSV FILES THAT FAILED TO LOAD	8-1
BACKGROUND ZIP FILE PROCESSING.....	8-1
9. EXPORTING CSV FILES VIA THE INTERFACE.....	9-1
CSV EXPORT SCREENS.....	9-1
EXPORTING DATA AS A ZIP FILE	9-1
EXPORTING LARGE ZIP FILES IN THE BACKGROUND.....	9-1
RUNNING CSVUTIL IN THE BACKGROUND	9-2
10. EXPORTING REFERENCED PUBLIC DATA DURING MULTI-TABLE EXPORTS	10-1
11. PIPING CSV OUTPUT TO A REMOTE ORACLE TRANSPORTATION MANAGEMENT INSTANCE	11-1
SYNCHRONIZING DATA BETWEEN DIFFERENT ORACLE TRANSPORTATION MANAGEMENT VERSIONS	11-3
12. EXPORTING TABLE SETS AND PIPING TO A REMOTE INSTANCE	12-1
13. COPYING RATES BETWEEN DATABASES USING ZIP FILES.....	13-1
STEP 1 – CREATE A CSVUTIL.CTL FILE (CSVUTIL CONTROL FILE) FOR EXPORTING.....	13-1
STEP 2 – USE THE INTEGRATION UPLOAD SCREEN TO UPLOAD THE ZIP FILE CREATED IN STEP 1	13-1
STEP 3 – DOWNLOAD THE ZIP FILE CONTAINING THE RATE OFFERING	13-1
STEP 4 – CREATE A CSVUTIL.CTL FILE FOR IMPORTING.....	13-1
STEP 5 – CREATE ANOTHER BACKGROUND ZIP FILE.....	13-1
STEP 6 – UPLOAD THE ZIP FILE FROM STEP 5 TO THE TARGET INSTANCE.....	13-2
14. IMPORTING VOYAGE SCHEDULE DATA	14-1
DELETING SCHEDULES	14-2
15. COPYING DOMAINS.....	15-1
IN SCHEMA COPY.....	15-1
WHAT THE OBJECTS DO	15-2
SET-UP	15-2
COPY DOMAINS.....	15-2
RESULT OF IN SCHEMA COPY	15-3
DATABASE LINK COPY	15-3
CREATE LINK FROM TARGET TO SOURCE DATABASE.....	15-3
GENERATE SCRIPT	15-4
COPY DOMAINS.....	15-4
DIFFERENCE BETWEEN DOMAINS.....	15-5
RERUN DATABASE LINK COPY	15-5
16. DELETING DOMAINS	16-1

17. REFERENCE A: DB.XML TRANSACTION CODES	17-1
18. REFERENCE B: SPECIFYING COMPLEX QUERIES	18-1
EXAMPLE OF A COMPLEX QUERY	18-1
19. REFERENCE C: CSVUTIL RESPONSE MESSAGES.....	19-1
RESPONSE MESSAGES WITH NO ERRORS.....	19-1
ERROR MESSAGES.....	19-1
IMPORT	19-2
EXPORT	19-5
20. APPENDIX: GENERAL PROPERTIES.....	20-1

Figures

Figure 5-1	5-3
Figure 5-2	5-5
Figure 5-3	5-11
Figure 9-1	9-3
Figure 11-1	11-2
Figure 12-1	12-2

Tables

Table 2-1	2-2
Table 2-2	2-3
Table 2-3	2-5
Table 2-4	2-8
Table 2-5	2-9
Table 2-6	2-10
Table 3-1	3-2
Table 5-1	5-17
Table 14-1	14-1
Table 14-2	14-2
Table 15-1	15-1

Table 15-2	15-2
Table 15-3	15-2
Table 15-4	15-4
Table 19-1	19-2
Table 19-2	19-2
Table 19-3	19-2
Table 19-4	19-3
Table 19-5	19-3
Table 19-6	19-4
Table 19-7	19-4
Table 19-8	19-5
Table 19-9	19-6
Table 20-1: General Properties	20-1

Send Us Your Comments

Oracle® Transportation Management Data Management Guide, Release 6.5.3

Part No. F82872-01

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

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

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

- Electronic mail: otm-doc_us@oracle.com

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

Preface

This manual is for members of the Oracle Transportation Management implementation team, who are responsible for maintaining and updating data in Oracle Transportation Management at your site. This manual provides step-by-step instructions for importing and exporting data in CSV and db.xml format.

This manual does not cover the installation of any components required to import or export. See the Administration Guide for for installation and configuration instructions.

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.

1. Introduction

DB.XML

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

The DB XML tool facilitates the direct query/update of data directly from/to the OTM database tables. As such, the tool should only be used by those already familiar with the responsibilities and capabilities that come with using such tools and who may already be familiar with database tools like SQLDeveloper, TOAD etc.

NOTE: Updates made directly to the OTM database by DB XML 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.

In the DB XML file, there can be more than one element 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 OTM 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_CORPORATION, LOCATION_REFNUM, 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 section 2, DB.XML.

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.

Why do I want to use DB.XML?

Compared to CSV (Comma Separated Values), DB.XML supports manipulation of parent-child records as a unit. This gives DB.XML an advantage compared to CSV when updating, for example, rate information.

How can I use DB.XML?

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

- OTM User Interface

- Command Line: using the OTM application server remotely.
- HTTP POST to servlet on OTM application server (requires authentication)
- SOAP web service

See section 2, DB.XML for details.

CSV

CSVUtil is a utility for importing and exporting data in CSV format in and out of the Oracle Transportation Management database. CSVUtil also exports data as a script of insert statements. This document describes how to use CSVUtil and shows some sample CSV files.

CSV files are compact and enable you to import large amounts of data into Oracle Transportation Management. You typically want to use CSVUtil when importing rates into a fresh installation of Oracle Transportation Management.

There are three ways to use CSVUtil:

- On the DOS/UNIX command line
- Via the Oracle Transportation Management web interface
- Via integration transmissions

A Sample CSV File

Below is a sample CSV file:

```
ICON
ICON_GID,ICON_XID,DESCRIPTION,PATH,DOMAIN_NAME,INSERT_USER,INSERT_DATE,UPDATE_USE
R,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.

After line 3 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,,
```

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.

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.

Migration Project

The Migration Project feature, added in Oracle Transportation Management v6.3.2, introduces 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 set represents object groups and objects are that needs 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 refer to the online help How-To Topic: Promote To Production (P2P) Process.

2. 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:

- Using an Object Name
- Using an Object Set Name
- 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 it is now 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="6.3.3">
  <TRANSACTION_SET useLOBElement="true">
    <"parent table name" parent_coll="value_1" parent_col2="CLOBID#1"
parent_col3="BLOBID#1" ..etc..>
      <"child table name" child_coll="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 Name

The **Object Name** is intended to refer to one of the OTM primary objects e.g. LOCATION, AGENT, RATE_GEO etc., though this is not enforced. The name actually corresponds to a pre-configured

property which contains the name of file containing the SQL query to execute to retrieve the required data.

For example, the following property specifies the file to be used to retrieve data for the LOCATION Object Name:

```
glog.integration.dbxml.query.LOCATION=sql/Location.sql
```

Note: the file location above is relative to the <OTM_HOME>/glog/glog_resources directory but it could be any file that is available on the CLASSPATH.

The following is an excerpt from the contents of the file provided with the standard installation:-

```
select location.*,
       cursor (select location_accessorial.* from location_accessorial where
location_accessorial.location_gid = location.location_gid) as
location_accessorial,
...etc...
from location
```

The use of a "cursor" is to produce the child element, in this case for the LOCATION_ACCESSORIAL records for this location, and there is a cursor for every required child table (and in "grandchild" tables and so on).

The base install will provide the following pre-configured queries:

Table 2-1

Object Name	SQL File
LOCATION	sql/Location.sql
RATE_GEO	sql/RateGeo.sql
RATE_OFFERING	sql/RateOffering.sql
AGENT_ACTION	sql/AgentAction.sql
AGENT_EVENT	sql/AgentEvent.sql
AGENT	sql/Agent.sql
CORPORATION	sql/Corporation.sql
SAVED_QUERY	sql/SavedQuery.sql
SAVED_CONDITION	sql/SavedCondition.sql
USER_MENU_LAYOUT	sql/UserMenuLayout.sql
MONITOR_PROFILE	sql/MonitorProfile.sql
SHIPMENT	sql/Shipment.sql
STATUS_TYPE	sql/StatusType.sql

Object Name	SQL File
WORKFLOW_TOPIC_INFO	sql/WorkflowTopicInfo.sql
NOTIFY_SUBJECT_CONTACT	sql/NotifySubjectContact.sql
PLANNING_PARAMETER	sql/PlanningParameter.sql
BN_RULE	sql/BNRule.sql
NOTIFY_SUBJECT_STYLESHEET	sql/NotifySubjectStylesheet.sql
OB_ORDER_BASE	sql/ObOrderBase.sql

New customer-defined object names and SQL files can be added by setting the associated properties.

The use of a 'where clause' is optional for the export by Object Name as it is feasible that the SQL file can contain the complete statement.

Note: In order to be somewhat generic, the provided files all require a 'where clause' to retrieve a specific record. Without it, ALL records for the object name will be retrieved.

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:

Table 2-2

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.

Specifying a SQL Query

The complete SQL query (similar to the contents of the provided Object Name files) 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 OTM. 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 OTM tables. It can also completely replace a current set of child records with a new set.

Note: The maximum size of the import file is limited by the `glog.integration.dbxml.importMaxSize` property.

Execution Mode

The execution of the import is done via the application business logic (formerly called APP mode). This makes available the capabilities to refresh associated application cache and to raise lifetime events for the objects create, modified or removed as part of the import.

The valid parent/child relationships are determined by the persistence logic internal to the OTM 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.

Transaction Code

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
- **U:** Update only 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.

Table 2-3

Object Name	Child Tables
LOCATION	LOCATION_ACCESSORIAL, LOCATION_ADDRESS, LOCATION_REFNUM, LOCATION_REMARK, LOCATION_ROLE_PROFILE, LOCATION_SPECIAL_SERVICE, LOCATION_STATUS, LOCATION_ACTIVITY_TIME_DEF
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
USER_MENU_LAYOUT	USER_MENU_LAYOUT
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
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

Refresh Cache

The OTM 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.

DB.XML User Interface

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. The following fields are available:-
 - i. DB Object: Object name to be exported. See Using an Object Name.
 - ii. Where Clause: For example you can enter DOMAIN_NAME='GUEST' or rownum<3. You can also combine the two like this DOMAIN_NAME='GUEST' and rownum<3.
 - b. DB Object Set. The following fields are available:-
 - i. DB Object Set: Object Set name to exported. See Using an Object Set Name.
 - ii. Where Clause: For example you can enter DOMAIN_NAME='GUEST' or rownum<3. You can also combine the two like this DOMAIN_NAME='GUEST' and rownum<3.
 - 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 DB Object as **LOCATION** and the 'where clause' as **LOCATION_GID = 'NYC'**

```

-<xml2sql>
-<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="N"
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">
<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>
</TRANSACTION_SET>
</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: If your output is too large for Notepad, you need to the command line to execute the command.

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 transactionCode from I to either II, U, 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 **successCount** and an **errorCount**. The count is the number of transactions that were successful or in error.

DB.XML Command Line Execution

A command line script exists to provide the same capability that is available via the OTM User Interface¹. The script is called `dbxml.sh` or `dbxml.bat` (depending on platform) and is present in the `<otm_home>/utils/integration/scripts` directory.

The DB XML command line operates as a remote web client (using the same API as the web UI).

The user's environment must be correct. In particular, the `GLOG_HOME` environment variable must be defined to point at the `<otm_home>` directory, and the `CLASSPATH` environment variable must be set as it is for running WebLogic.

Some required parameters depend on the mode selected. The following are the required parameters for each mode:

Mode = "remote web client"

Table 2-4

Parameter	Usage
<code>web</code> [The <code>server</code> parameter still functions but is deprecated]	The hostname of the OTM application server where the request will be sent via "http" (default). If non-standard ports are used, the format given should be <i>hostname:port</i> . Note: The default scheme of "http" can be modified by setting the property :- <code>glog.util.command.scheme=scheme</code> For example, <code>glog.util.command.scheme=https</code>
<code>username</code>	OTM application user name to used for execution of database commands. This is required for the correct VPD security.
<code>password</code>	OTM application password to use for execution of database commands. If not specified and executing in an interactive user session, DBXML will prompt at the interactive session standard input for a password. If executing in a non-interactive environment (no login terminal), this parameter is required; however, it is considered unsafe to use when prompting can be done because such a password can be seen in the Unix/Linux <code>ps</code> command's output.

Database direct access and backend internal API modes have been removed, so the legacy `-app`, `-dbURL`, `-dbConn`, `-dbUsername`, and `-dbPassword` parameters have no effect.

¹ See OTM Administration Guide for complete instructions on configuring a Java command environment.

Exporting DB.XML

Following are the additional command line parameters for the DBXML export command "xmlExport":

Table 2-5

Parameter	Usage
-dbObjectName Or -dbObjectSetName Or -sqlQuery Or -sqlFile Or -entityName	Specifies the Object Name. See Using an Object Name Specifies the Object Set Name. See Using an Object Set Name. Specifies the SQL Query. See Specifying a SQL Query. Same as -sqlQuery except that the statement is contained in a file (useful to avoid issues with the command shell and special characters) Specifies the Migration Entity. See Migration Entity
-whereClause	Used in conjunction with -dbObjectName and -dbObjectSetName to narrow the result set of retrieved objects.
-rootName	Used in conjunction with -sqlQuery and -sqlFile to provide the root XML element of each retrieved object.
-pkList	Used in conjunction with -entityName to provide the list of object PKs to be retrieved.
-footPrint	Flag available for all commands and indicates if INSERT and UPDATE date/time and user columns are retrieved.
-useLOBElement	Flag available for all commands and indicates if LOB column data is to be exported as separate elements or as base 64 encoded attribute values.
-localDir & -localFileName	Used to specify that results should be saved to a file rather than displayed to the console.

Using Pre-defined Primary Data Objects

The following is an example of exporting (using remote web server access by default) the first RATE_GEO database object found in the database:

```
dbxml.sh xmlExport -web localhost -username GUEST.ADMIN -password -dbObjectName  
RATE_GEO  
-whereClause "rownum < 2" -localDir ./ -localFileName rate_geo1.db.xml
```

This example creates the file "rate_geo1.db.xml" in the current working directory.

You need to modify the following arguments specific to your situation:

- Web: hostname of remote OTM application server.
- Username: User name used to login to the remote Oracle Transportation Management instance.
- Password: password corresponding to the username.
- whereClause: SQL 'where' clause used to limit size of export.
- localDir: directory on your PC where output file is written.
- localFileName: name of local output file, relative to localDir. If not specified, defaults to "command.out". Note that `-localFileName` has no effect if `-localDir` is not also specified.

Using a SqlQuery

The following is an example of exporting (using internal API client access) all the activity records in the database:

```
dbxml.sh xmlExport -app localhost
-username GUEST.ADMIN -sqlQuery "select * from activity"
-rootName ACTIVITY -localDir ./ -localFileName activity.db.xml
```

The above command creates the `activity.db.xml` file in the current working directory.

Importing DB.XML

Following are the additional command line parameters for the DBXML export command "xmlImport":

Table 2-6

Parameter	Usage
<code>-inputXMLFile</code>	Full Path Name of the file containing the DBXML transaction set to be imported.
<code>-transactionCode</code>	The transaction code for the import. See the Reference A: DB.XML Transaction Codes section for possible transaction codes.
<code>-schema</code>	If present, must be 'GLOGOWNER' (the default) or 'REPORTOWNER'. If absent, the default is assumed.
<code>-managedTables</code>	If the transaction code is RC, this specifies the managed tables that are to be considered for child deletion.
<code>-updateCache</code>	Indicator to specify that in memory cache should be refreshed.
<code>-events</code>	Indicator to specify that lifetime events should be published for modified data.
<code>-commitScope</code>	One of 'PK' (default) or 'SET'. If not present, default is assumed.

You can use `dbxml.sh` or `dbxml.bat` (depending on platform) to import a client-side `db.xml` file into a remote Oracle Transportation Management instance.

Here is a sample command line:

```
dbxml.sh xmlImport -web localhost -username DBA.ADMIN -password -
transactionCode IU -inputXMLFile rate.db.xml
```

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

DB XML Servlet

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

<http://localhost/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"
http://localhost/GC3/glog.integration.servlet.DBXMLServlet?command=xmlExport
```

Export Message Format

Object Name

The following is the format for the XML message to export XML based on a DB object name:

```
<?xml version="1.0" encoding="UTF-8"?>
<sql2xml>
  <DBObject>
    <Name>{db object name}</Name>
    <Predicate>{where clause}</Predicate>
  </DBObject>
  <FootPrint>{Y|N}</FootPrint>
  <UseLOBElement>{Y|N}</UseLOBElement>
</sql2xml>
```

For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<sql2xml>
```

```

    <DBObject>
      <Name>LOCATION</Name>
      <Predicate>location_gid = 'GUEST.MY_LOC'</Predicate>
    </DBObject>
    <FootPrint>N</FootPrint>
    <UseLOBElement>Y</UseLOBElement>
  </sql2xml>

```

Object Set

The following is the format for the XML message to export XML based on a DB object set name:

```

<?xml version="1.0" encoding="UTF-8"?>
<sql2xml>
  <ObjectSet>
    <Name>{db object set name}</Name>
    <Predicate>{where clause}</Predicate>
  </ObjectSet>
  <FootPrint>{Y|N}</FootPrint>
  <UseLOBElement>{Y|N}</UseLOBElement>
</sql2xml>

```

For example,

```

<?xml version="1.0" encoding="UTF-8"?>
<sql2xml>
  <ObjectSet>
    <Name>DomainReferenceData</Name>
    <Predicate>domain_name = 'GUEST'</Predicate>
  </ObjectSet>
</sql2xml>

```

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>{where 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: It is necessary to encode the SQL comparison operators (<, >) with < and >

For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<sql2xml>
  <Query>
    <RootName>Location</RootName>
    <Statement>SELECT * FROM LOCATION WHERE rownum &lt; 10</Statement>
  </Query>
</sql2xml>
```

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="6.3.3">
  <TransactionCode>{I|II|U|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>
```

```
</TRANSACTION_SET>
</xml2sql>
```

For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<xml2sql Version="6.3.3">
  <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 located under:

```
http://<server:port>/GC3Services/glog.integration.webservice.command.CommandService?WSDL
```

Where `server` and `port` are specific to the host and port configured for the WebLogic server running OTM (aka the OTM application server).

The service is secured via Web Service Security in common with all other OTM 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, i.e.:

```
<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="6.3.3">
  <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.

3. Loading CSV Data via the Command Line

This chapter describes how to import and export CSV from the command line.

Importing and Exporting on the Server Side

This section describes how to use CSVUtil to export and import data from a local Oracle Transportation Management database. Note you must execute this command within the context of the OTM application server's operating system environment (especially including the CLASSPATH environment variable specification).

CSVUtil has the following syntax and arguments.

```
java glog.database.admin.CSVUtil -command
<i|ii|iu|u|uu|d|dd|xcsv|xcsvcd|xcsvpcd|xcsvpd|xsql> -connectionId
<connectionId> -tableName <tableName> -dataDir <dataDirectory> -dataFileName
<dataFileName> -appendFile -runsqlloader -domain_name <domainName> -useT2 <Y|N>
-debug -XMLCSVOutput -sqlQuery <queryString> -whereClause <whereClause> -
clobDir <clobDirectory> -xvalidate <Y|N> -encoding <encoding>
```

CSVUtil supports the following commands and arguments (note that some of these options conflict and should not be used in combination, as noted below):

Table 3-1

Commands	Arguments
command	<p>Mandatory parameter, must specify one of the following commands:</p> <ul style="list-style-type: none">i - insert CSV data into the databaseii - insert data, while suppressing unique key constraint violationsiu - attempts to insert data. If a primary key violation occurs, it updates the data. No delete statements are generated.u - update data in the databaseuu - update data, while suppressing "no data found" constraint violationsd- delete data from the databasedd- delete data, while suppressing "no data found" constraint violationsxcsv - export a CSV filexcsvcd - 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

Commands	Arguments
connectionId	<p>The connectionId is a shorthand method for providing an Oracle username, password, and server for direct database access.</p> <p>For example, if you specify your connectionId as codegen, you need to add the following properties to your glog.properties file:</p> <pre>glog.database.codegen.schema=glogowner glog.database.codegen.t2client.driverClassName=oracle.jdbc.driver.OracleDriver glog.database.codegen.t2client.databaseURL=jdbc:oracle:thin:@localhost:1521 glog.database.codegen.user=glogload glog.database.codegen.password=glogload glog.database.codegen.server=dbserver glog.database.codegen.t2client.pool=</pre> <p>This parameter is required in all usages of CSVUtil; only the database schema attribute of the connectionId will be used except for CSVUtil runs with the -command i and -runsqlloader parameters (in which case all attributes will be forwarded to the runsqlloader program).</p>
dbUrl	<p>A somewhat more verbose means of specifying the oracle database instance to access. As for the glog.database.codegen.t2client.databaseURL property, the URL must be of the format:</p> <pre>jdbc:oracle:thin:@localhost:1521</pre> <p>Note that the -dbService, -dbUser, and -dbPassword parameters must also be specified if this parameter is specified.</p> <p>This option is only applicable for CSVUtil runs with the -command i and -runsqlloader parameters.</p>
dbUser	<p>The Oracle database username used to access the Oracle Transportation Management data in the database. Usually glogdba.</p> <p>This option is only applicable for CSVUtil runs with the -command i and -runsqlloader parameters.</p>
dbPassword	<p>The Oracle database password for the username specified by the dbUser parameter.</p> <p>This option is only applicable for CSVUtil runs with the -command i and -runsqlloader parameters.</p>

Commands	Arguments
dbService	<p>The Oracle database instance (SQL catalog) to be accessed on the specified Oracle database server host.</p> <p>This option is only applicable for CSVUtil runs with the -command i and -runsqlloader.</p>
tableName	<p>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.</p>
tableSet	<p>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.</p>
dataDir	<p>The dataDir argument specifies the location to either read or write the file specified in the -dataFileName argument. The following glog.property file setting controls the default value of the dataDir argument:</p> <pre>glog.database.load.dir=d:\\upload</pre> <p>In this case, the default directory has been set to d:\upload. Note that two backslashes are required in glog.properties.</p>
dataFileName	<p>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.</p>
appendFile	<p>The appendFile argument only applies to the export commands (xcsv and xsql). If specified, CSVUtil will append to the file specified by the dataFileName argument instead of overwriting it.</p>
removeUndefinedColumns	<p>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:</p> <pre>-removeUndefinedColumns N</pre> <p>This option is only available when running CSVUtil directly on the command line. It is not available using either the web or ClientUtil.</p>
runsqlloader	<p>The runsqlloader argument only applies to import commands (-command i). If specified, the Oracle sqlloader program will be used to load the CSV file instead of a java procedure. If you have sqlloader installed on your system the sqlloader is faster than the java procedure.</p>

Commands	Arguments
-maxError	<p>By default in CSVUtil, after 50 errors occur, processing stops. You can change this default value to make it higher or lower using the -maxError command line argument.</p> <p>For example:</p> <p>-maxError 20</p> <p>This parameter is currently only available when running CSVUtil as a java application on the command line.</p>
domain_name	<p>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.</p>
includeChildren	<p>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,</p> <p>-domain_name PARENT -includeChildren</p> <p>would specify that records in the "PARENT", "PARENT/CHILD", and "PARENT/CHILD/GRANDCHILD" domains would be included in the export.</p>
useT2	<p>Used to avoid using the T2Connection class, which depends on VPD being already setup correctly. When loading certain Oracle Transportation Management "system" tables, it is necessary to avoid the use of the T2 connection class (it's a chicken or the egg type situation). For normal data loading, using the T2Connection class is correct and desirable.</p>
debug	<p>Used for debugging. Causes extra logging of the operations performed by CSVUtil.</p>
XMLCSVOutput	<p>If true, then output looks like this:</p> <pre data-bbox="521 1398 1256 1671"><TableName></TableName> or <SqlQuery>...</SqlQuery> <ColumnList></ColumnList> <ExecSQL></ExecSQL> <Row>...</Row> <Row>...</Row></pre>
sqlQuery	<p>If specified, then xcsv command is required and tableName, domain_name, and whereClause are ignored.</p>

Commands	Arguments
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.
whereClause	Only used when tableName is specified and domain_name is omitted.
clobDir	Directory where external Clob files are read. Only used when importing external Clob files and not using sqlloader.
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 OTM 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.
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.
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.
clobEncoding	The encoding of separate CLOB files you import. Only applicable if importing separate CLOB files. 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. If not specified, the value of the -encoding parameter is used for CLOB file imports.
migrate	If specified, SQL errors during CSVUtil execution will cause CSVUtil to exit with a completion status of 1. Normally, CSVUtil always exits with status 0 (success code), regardless of the success or failure of the database operation. This option allows scripts using command-line CSVUtil to detect when a problem has occurred.

Commands	Arguments
badFileDir	Specifies the directory path (relative to the current working directory if a relative path) where "bad" records from input files should be logged. If errors occur for an input file, an output file will be written to this directory with the name of the input file (without directory prefix) and an additional suffix of ".bad". This file will contain all records that failed to be processed. By default, bad files are not created.
remoteHost	Only meaningful for the "x" commands, specifies that the exported data, rather than being delivered to a local file, should be pushed to a remote Oracle Transportation Management instance, where this parameter's value is the host name of the remote OTM application server. If this parameter is specified, the remoteUser, remotePassword, and remoteCommand parameters are also needed.
remoteUser	Only meaningful if remoteHost is specified. Specifies the OTM username for logging into the remote Oracle Transportation Management system for performing the batch transfer.
remotePassword	Only meaningful if remoteHost is specified. Specifies the OTM password for logging into the remote Oracle Transportation Management system for performing the batch transfer.
remoteCommand	Only meaningful if remoteHost is specified. Specifies the CSVUtil command (as defined for the command parameter) to execute on the remote system for receiving the transfer. Note that this cannot be one of the "x" (export) commands, but only one of the insert, update, or delete commands ("I", "ii", "iu", "u", "uu", "d", or "dd"), since this remote command is receiving the data exported locally.
fromDomain	Specifies the from-domain for VPD domain copying. If specified, the toDomain parameter must be specified as well.
toDomain	Specifies the to-domain for VPD domain copying. If specified, the fromDomain parameter must be specified as well.
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.

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)
- In the CSV file, refer to an external file holding the Clob. (no size restrictions on the CLobs, newline characters allowed)

Note: CSVUtil can only handle one Clob per record.

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

```
CLOB_TEST
```

```
SEQ,DESCR,XML
9,"THIS IS SO COOL",<asdf>blahblah</asdf>
10,"LINE2",<qwerty>yaya</qwerty>
```

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,"THIS IS SO COOL",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 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.

Exporting With Parent Data

To export a data record with its parent data, you can do the following:

```
java glog.database.admin.CSVUtil -command xcswupd -tableName SHIPMENT -
whereClause "shipment_gid = 'MDIETL.184'" -connectionId angel37
```

The above command exports the record for shipment MDIETL.184, along with all the referenced non-public foreign key records required to successfully load the SHIPMENT record in a foreign database. The generated CSV file is in multi-table format.

Note: All the xcsw* commands are far more expensive in terms of CPU usage than the plain xcsw 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 do the following:

```
java glog.database.admin.CSVUtil -command xcswcd -tableName SHIPMENT -
whereClause "shipment_gid = 'MDIETL.184'" -connectionId angel37
```

The above command exports the record for shipment MDIETL.184, along with all the subordinate child tables such as shipment_stop, shipment_stop_d etc.

Exporting With Both Parent and Child Data

To export a data record with both its parent and child data, you can do the following:

```
java glog.database.admin.CSVUtil -command xcswpcd -tableName SHIPMENT -
whereClause "shipment_gid = 'MDIETL.184'" -connectionId angel37
```

This command should be used with care since it can take a while to run.

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 an authorized GL_USER, 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.

4. Loading CSV Data via Web Pages

Running CSVUtil via the command line is only possible if your client environment is configured correctly. If your client environment is not configured, you can still run CSVUtil via the web.

Importing

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

1. Log 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.
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. If you are loading a large file, you may specify the **runsqlloader** option. This will only work if sqlloader is installed on the Oracle Transportation Management application server. The following line must be added to the jserv.properties file to make sqlloader run from the web:

```
wrapper.path = d:/product/oracle/ora81/bin
```

This entry would be different depending on the location of the Oracle bin directory.

10. The **xvalidate** drop-down list allows you to turn off verbose diagnostic messaging. To leave messaging on, the value in the drop down list should be Y, which is the default.
11. In the encoding drop down list, 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.
12. Click **Run** and Oracle Transportation Management displays a results page.

To read more about interpreting error messages, see the **Reference C: CSVUtil Response Messages** section.

5. 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_C
ODE, COUNTRY_CODE3_GID, TIME_ZONE_GID, LAT, LON, IS_TEMPORARY, IS_MAKE_APPT_BEFORE
_PLAN, RATE_CLASSIFICATION_GID, DOMAIN_NAME, IS_SHIPPER_KNOWN, IS_ADDRESS_VALID,
IS_MIXED_FREIGHT_THU_ALLOWED, SLOT_TIME_INTERVAL, SLOT_TIME_INTERVAL_UOM_CODE,
SLOT_TIME_INTERVAL_BASE, IS_LTL_SPLITABLE
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, US
A, 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_MASTE
R,IS_SHIPPING_AGENTS_ACTIVE,IS_ALLOW_HOUSE_COLLECT
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_REGIO
N,ALLOW_TENDER,IS_ACCEPT_SPOT_BIDS,IS_ACCEPT_BROADCAST_TENDERS,IS_LOCALIZE_B
ROADCAST_CONTACT,DO_CONDITIONAL_ACCEPTS,IS_INTERNAL_NVOC,IS_ACCEPT_BY_SSU_A
LLOWED,IS_COPY_INV_DELTA_BACK_TO_SHIP,INVOICING_PROCESS
EXEC SQL ALTER SESSION SET NLS_DATE_FORMAT = 'YYYYMMDDHH24MISS'
MYDOMAIN.YELLOW,YELLOW,N,MYDOMAIN,N,Y,N,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_HANDLI
NG_TIME_UOM_CODE,FIXED_HANDLING_TIME_BASE,CREATE_XDOCK_HANDLING_SHIPMENT,CRE
ATE_POOL_HANDLING_SHIPMENT,DOMAIN_NAME
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_VALUE_BASE
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.

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.

X_LANE	
PK	X_LANE_GID
	X_LANE_XID
FK7	SOURCE_LOCATION_GID
	SOURCE_CITY
	SOURCE_PROVINCE_CODE
	SOURCE_POSTAL_CODE
FK5	SOURCE_COUNTRY_CODE3_GID
	SOURCE_ZONE4
	SOURCE_ZONE1
	SOURCE_ZONE2
	SOURCE_ZONE3
FK6	SOURCE_GEO_HIERARCHY_GID
FK3	DEST_LOCATION_GID
	DEST_CITY
	DEST_PROVINCE_CODE
	DEST_POSTAL_CODE
FK1	DEST_COUNTRY_CODE3_GID
	DEST_ZONE4
	DEST_ZONE1
	DEST_ZONE2
	DEST_ZONE3
FK2	DEST_GEO_HIERARCHY_GID
FK8	SOURCE_REGION_GID
FK4	DEST_REGION_GID
	LOADED
	DOMAIN_NAME
	INSERT_USER
	INSERT_DATE
	UPDATE_USER
	UPDATE_DATE

Figure 5-1

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_HI
ERARCHY_GID,DEST_POSTAL_CODE,DEST_COUNTRY_CODE3_GID,DEST_GEO_HIERARCHY_GID,DOMA
IN_NAME
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

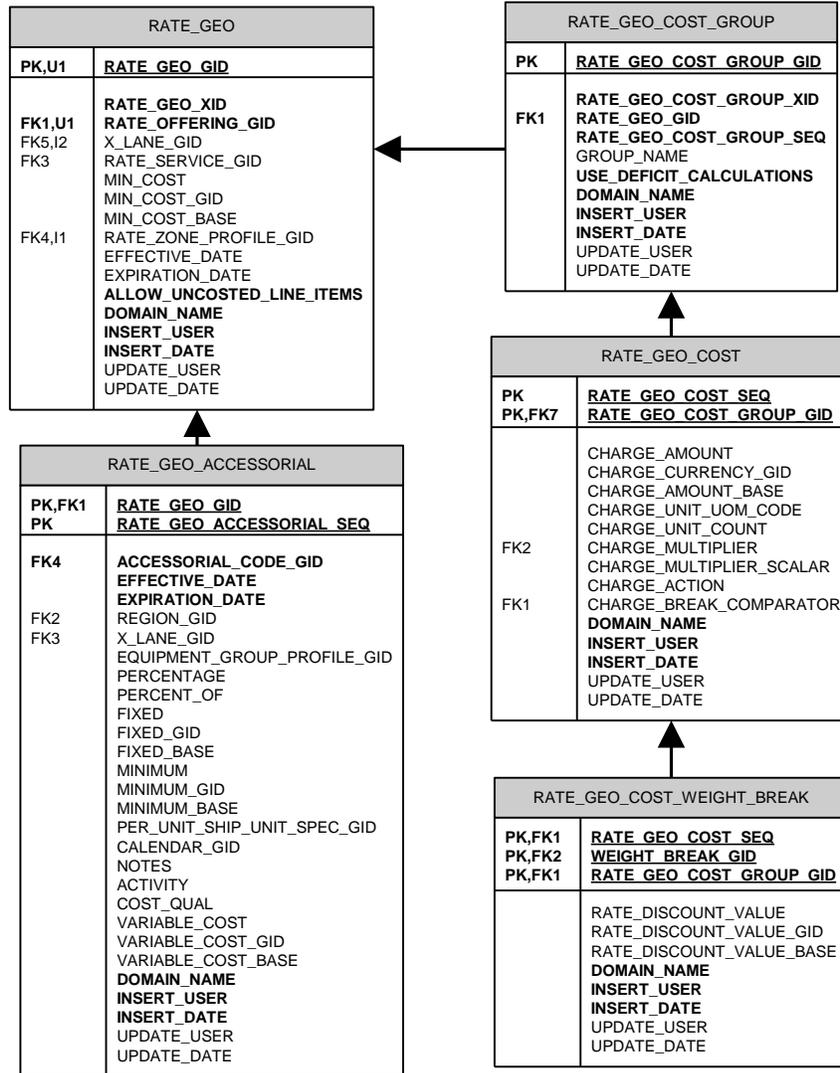


Figure 5-2

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—Based on Simple Unit Breaks

This 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_B
ASE,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_G
ROUP_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_C
OUNT,CHARGE_BREAK_COMPARATOR,DOMAIN_NAME
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_TYP
E,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_UNI
T_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,DOMAIN_NAME
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—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
-

Summary

1. Import RATE_GEO table.

```

RATE_GEO
RATE_GEO_GID,RATE_GEO_XID,RATE_OFFERING_GID,MIN_COST,MIN_COST_GID,MIN_COST_B
ASE,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_MULTIPLE
R_SCALAR,CHARGE_ACTION,CHARGE_TYPE,USE_DEFAULTS,CHARGE_MULTIPLIER_OPTION,USE
S_UNIT_BREAKS,DOMAIN_NAME,IS_FILED_AS_TARIFF
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,DO
MAIN_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_G
ROUP_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_CODE,CHARGE_UNIT_COUNT,CHARGE_MULTIPLIER,
CHARGE_MULTIPLIER_SCALAR,CHARGE_ACTION,DOMAIN_NAME
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–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_B
ASE,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_VALUE2,CHARGE_MULTIPLIER,CHARGE_AMOUNT,
CHARGE_AMOUNT_GID,CHARGE_AMOUNT_BASE,CHARGE_UNIT_COUNT,CHARGE_MULTIPLIER_SCA
LAR,CHARGE_ACTION,CHARGE_TYPE,USE_DEFAULTS,CHARGE_MULTIPLIER_OPTION,USES_UNI
T_BREAKS,DOMAIN_NAME,ROUNDING_TYPE,ROUNDING_FIELDS_LEVEL,ROUNDING_APPLICATIO
N,IS_FILED_AS_TARIFF
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,DO
MAIN_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_G
ROUP_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_CODE,CHARGE_UNIT_COUNT,CHARGE_MULTIPLIER,
CHARGE_MULTIPLIER_SCALAR,CHARGE_ACTION,DOMAIN_NAME
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 TL Rates

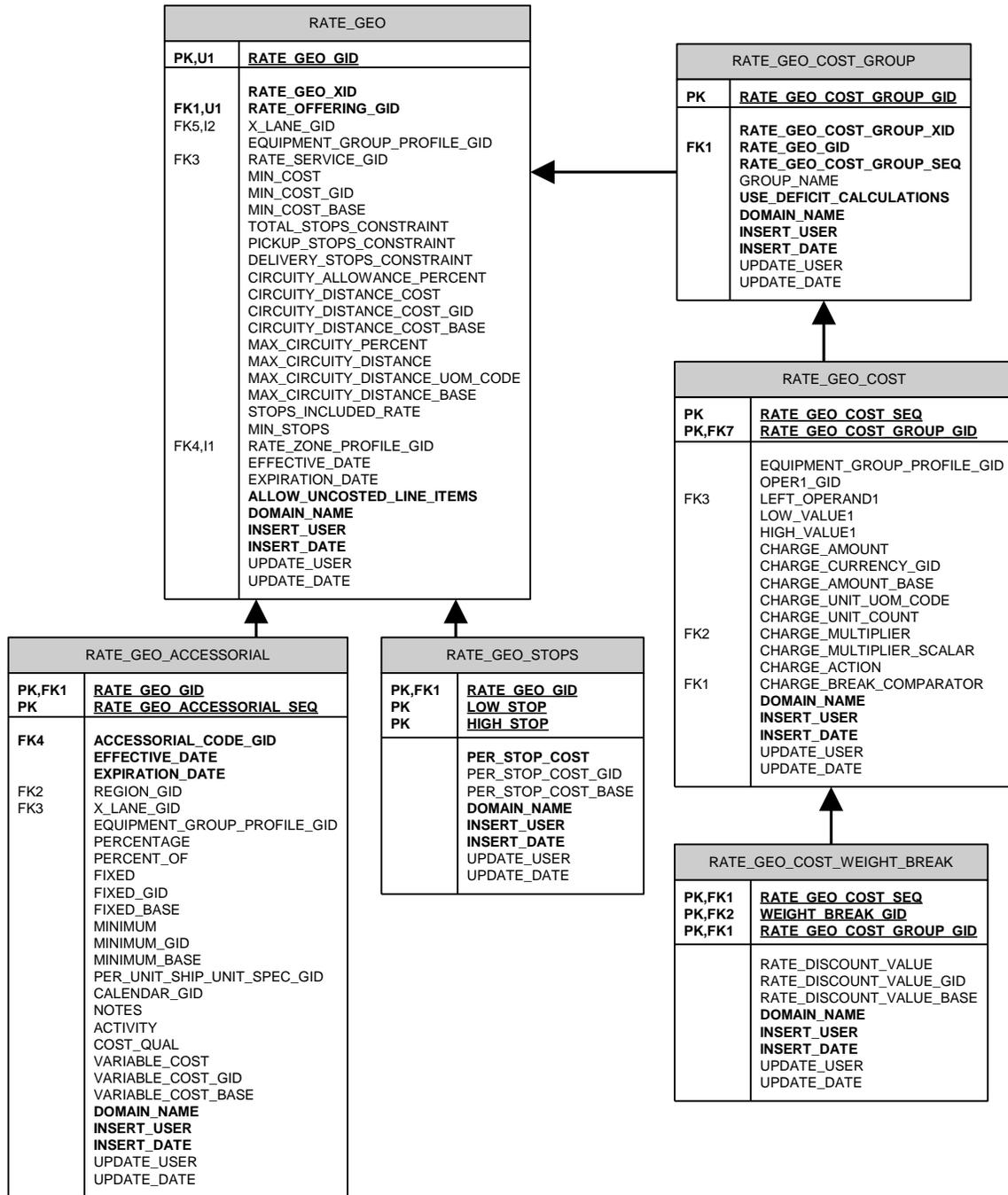


Figure 5-3

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

- Oper1_gid – field value "BETWEEN" is a shortcut for $X > \text{low}$ and $X \leq \text{high}$. 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–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_B
ASE,X_LANE_GID,TOTAL_STOPS_CONSTRAINT,STOPS_INCLUDED_RATE,DOMAIN_NAME
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_COS
T_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_G
ROUP_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_CURRENCY_GID,CHARGE_AMOUNT_BASE,DOMAIN_NAM
E
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–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_B
ASE,X_LANE_GID,TOTAL_STOPS_CONSTRAINT,STOPS_INCLUDED_RATE,DOMAIN_NAME
"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,CH
ARGE_AMOUNT_GID,CHARGE_AMOUNT_BASE,CHARGE_UNIT_UOM_CODE,CHARGE_UNIT_COUNT,CH
ARGE_ACTION,CHARGE_TYPE,USE_DEFAULTS,CHARGE_MULTIPLIER_OPTION,USES_UNIT_BREA
KS,DOMAIN_NAME,IS_FILED_AS_TARIFF
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,DO  
MAIN_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_COS  
T_BASE,DOMAIN_NAME  
MYDOMAIN.194-064-TL2,1,1,50,USD,50,MYDOMAIN  
MYDOMAIN.194-064-TL2,2,2,65,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_G  
ROUP_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_CODE,CHARGE_UNIT_COUNT,CHARGE_MULTIPLIER,  
CHARGE_MULTIPLIER_SCALAR,CHARGE_ACTION,DOMAIN_NAME  
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,CHARGE_AMOUNT_BASE,CHARGE_UNIT_UOM_CODE,C
HARGE_UNIT_COUNT,CHARGE_MULTIPLIER,CHARGE_MULTIPLIER_SCALAR,CHARGE_ACTION,DO
MAIN_NAME
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–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_B
ASE,X_LANE_GID,TOTAL_STOPS_CONSTRAINT,STOPS_INCLUDED_RATE,DOMAIN_NAME
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,CH
ARGE_AMOUNT_GID,CHARGE_AMOUNT_BASE,CHARGE_UNIT_UOM_CODE,CHARGE_UNIT_COUNT,CH
ARGE_ACTION,CHARGE_TYPE,USE_DEFAULTS,CHARGE_MULTIPLIER_OPTION,USES_UNIT_BREA
KS,DOMAIN_NAME,IS_FILED_AS_TARIFF
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,DO
MAIN_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_G
ROUP_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_CODE,CHARGE_UNIT_COUNT,CHARGE_MULTIPLIER,
CHARGE_MULTIPLIER_SCALAR,CHARGE_ACTION,CHARGE_BREAK_COMPARATOR,DOMAIN_NAME
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_PROF
ILE_NAME,DATA_TYPE,LOOKUP_TYPE,UOM_TYPE,DOMAIN_NAME,INSERT_USER,INSERT_DATE,
UPDATE_USER,UPDATE_DATE
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_UNI
T_BREAK_MAX,DOMAIN_NAME,INSERT_USER,INSERT_DATE,UPDATE_USER,UPDATE_DATE
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,DOMAIN_NAME
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–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.

Table 5-1

	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_B
ASE,X_LANE_GID,TOTAL_STOPS_CONSTRAINT,STOPS_INCLUDED_RATE,DOMAIN_NAME
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_G
ROUP_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_OPERAND
1,LOW_VALUE1,CHARGE_AMOUNT,CHARGE_CURRENCY_GID,CHARGE_UNIT_UOM_CODE,CHARGE_U
NIT_COUNT,CHARGE_MULTIPLIER,CHARGE_MULTIPLIER_SCALAR,CHARGE_ACTION,CHARGE_BR
EAK_COMPARATOR,CHARGE_TYPE,CHARGE_MULTIPLIER_OPTION,USES_UNIT_BREAKS,ROUNDIN
G_TYPE,ROUNDING_FIELDS_LEVEL,ROUNDING_APPLICATION,IS_FILED_AS_TARIFF
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_TYP
E,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_UNI
T_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,CHARGE_DISCOUNT,DOMAIN_NAME
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

```

6. 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, as opposed to 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 is small sample file. This example omits the optional UOM line.

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

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

```
SShipUnit
domainName,unitWidth,sShipUnitGid,isSplittable,unitNetVolume,unitNetWeight,shipU
nitCount,unitWeight,unitVolume,unitHeight,receivedNetVolume,receivedNetWeight,u
nitLength,sShipUnitXid
UOM:,,,CUFT,LB,,LB,,,CUFT,LB,,
GUEST,,GUEST.001,false,0,10,1,10,,,0,0,,001
```

Here is 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 isSplittable field).

```
SShipUnit
domainName,unitWidth,sShipUnitGid,isSplittable,unitNetVolume,unitNetWeight,shipU
nitCount,unitWeight,unitVolume,unitHeight,receivedNetVolume,receivedNetWeight,u
nitLength,sShipUnitXid
GUEST,,GUEST.001,false,0 CUFT,10 LB,1,10 LB,,,0 CUFT,0 LB,,001
```

Here is an example that will result in errors. You cannot specify a UOM line if 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,isSplittable,unitNetVolume,unitNetWeight,shipU
nitCount,unitWeight,unitVolume,unitHeight,receivedNetVolume,receivedNetWeight,u
nitLength,sShipUnitXid
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 and Exporting AppServer CSV Files

Importing

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 Web Pages** chapter for details about this page.

Errors encountered when importing are reported back to the screen.

Exporting

Care must be taken when exporting an AppServer CSV file due to the lack of support for where-clauses. You should be logged in as a user whose vpd_profile limits the number of rows selected from the entity you plan on exporting. Where-clauses will be supported in future releases. In the example below, the user is logged in as “GUEST.FEWROWS”. This user has a vpd_profile which limits the number of rows in the s_ship_unit table.

You can use the following URL to export (if it is not on your user menu):

```
http://hostname/servlets/glog.integration.servlet.IntegrationMenuServlet?integr
ation_stylesheet=integration/csvexport.xsl
```

1. In the **command** field, select the “as.xcsv” command.
2. In the **tableName** field, specify an “EntityName” instead of a table name. In this case, the entity name is “SShipUnit” which differs from the database table name, which would be “S_SHIP_UNIT”.
3. Click the **Run** button. Your output will then appear as follows:

You can then do a “View->Full Screen” in your browser, and select “View Source” (by right-clicking on your mouse). This will place the output in notepad so you can save it to a local file.

Load CSV Files in the Report Owner Directory

Below is the command for loading CSV files in the reportowner directory.

From the application server script8 directory, run the following command.

```
./update_onecsv_rpt.sh REPORT_CONTROL /opt/otm-55-w1/glog/config/dbareportowner
```


7. 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.

Refer to the CSV Data Loading Guide document for details on the CSV file format.

GlogXML 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_GI
D,SOURCE_GEO_HIERARCHY_GID,DEST_POSTAL_CODE,DEST_COUNTRY_CODE3_GID,DEST_GEO_HIE
RARCHY_GID,DOMAIN_NAME</CsvColumnList>
<CsvRow>"MYDOMAIN.194064","194064","194","USA","USZIP3","064","USA","USZIP3","M
YDOMAIN"</CsvRow>
<CsvRow>"MYDOMAIN.194065","194065","194","USA","USZIP3","065","USA","USZIP3","M
YDOMAIN"</CsvRow>
</CSVDataLoad>
</GLogXMLElement>
</TransmissionBody>
</Transmission>
```


8. 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 1
-dataFileName activity2.csv -command 1
```

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.

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, you are prompted to download a "results" zip file.

Click the **Save** button 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.

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 or 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".

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 1
-dataFileName activity2.csv -command 1
-mailTo youremail@yourcompany.com -mailFrom youremail@yourcompany.com -subject
zipFileProcessDone -message Hello -smtpHost mail-server.com
```

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 a stack trace, which you can read with a text editor rather than WinZip.

9. 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.

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 command "Upload an XML/CSV Transmission":
4. Press the Save button 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.
6. The zip file also contains a log file containing information regarding the execution(s) 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. The following screen shot shows you how:

Export Object Type
 Export Table

Output Destination
 Remote Instance

Run Job in Background
 Y

Use Select List
 N

Run

Command
 XCSV

Table Name
 ACTIVITY

Exclude Public
 Y

Where Clause

Remote Host

Remote User

Remote Password

From Domain

To Domain

Remote Command
 i

Remote Host Version
 6.0

Email Address
 youremail@email.com

SMTP Host
 mail.email.com

Run

Figure 9-1

The remote fields in the above screenshot have the same meanings as defined for the command-line CSVUtil command in section 3 of this Guide.

As shown above, specify your email address and a SMTP Host to run in the background. The results will be emailed to you when the background job completes (instead of returning the results to the screen).

In this example, the following content was emailed:

```
<CSVUtil>
<Command>xcsv</Command>
<DataDir>/</DataDir>
<DataFileName>>null</DataFileName>
<ExcludePublic>>true</ExcludePublic>
<Write>
<DatabaseGlobalName>QGC317.HARMONY.GLOGTECH.COM</DatabaseGlobalName>
<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,INS
ERT_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.GLOGLO
AD"
"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, such as piping a large table set to a RemoteHost.

10. Exporting Referenced PUBLIC Data during Multi-Table Exports

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

Here is a sample CSVUtil command line for exporting referenced public data:

```
java glog.database.admin.CSVUtil -excludePublic N -command xcsvwpcd -  
connectionId localdb4 -dataDir . -dataFileName whatever.csv -tableName RATE_GEO  
-whereClause "domain_name = 'DGANO'"
```

Notice the `-excludePublic` option is set to `N`, meaning that public data should not be excluded (it should be exported, in other words).

11. Piping CSV Output to a Remote Oracle Transportation Management Instance

CSVUtil supports piping CSV Output to a remote Oracle Transportation Management instance. Refer to the screenshots below:

Export Object Type
 Export Table

Output Destination
 Remote Instance

Run Job in Background
 Y

Use Select List
 N

Run

Command
 XCSV

Table Name
 ACTIVITY

Exclude Public
 Y

Where Clause

Remote Host
 localhost

Remote User
 DBA.ADMIN

Remote Password
 ●●●●●●

From Domain

To Domain

Remote Command
 ii

Remote Host Version
 6.0

Email Address

SMTP Host

Run

Figure 11-1

In the above example, the ACTIVITY table is first exported. The results are then immediately sent to the given Remote Host (in this case back to localhost). You must also specify the Remote User, Remote Password and Remote Command (CSVUtil command) to use on the Remote host.

When you click the Run button you will get XML output showing all the processing that occurred – i.e. export the activity table, send the file over to the remote host, then run CSVUtil on the remote host,

and get feedback from the remote host. If you are moving a large amount of data, you should background this operation so your web browser does not time out; the small size of the return document reporting the final completion status does not reduce the amount of time for the actual local export and remote import operations.

Synchronizing Data between Different Oracle Transportation Management Versions

CSVUtil supports the ability to extract and push data to a remote Oracle Transportation Management instance whose version is earlier (or later).

When pushing data to the remote instance, CSVUtil queries the data dictionary to determine which columns in the given table exist on the remote system. Columns which do not exist on the remote system are omitted from the CSV file.

When pushing data to a remote system, you must indicate the version of the remote system. This is required because the format of the URL is different between version 5.x and version 6.x of Oracle Transportation Management.

12. Exporting Table Sets and Piping to a Remote Instance

CSVUtil supports exporting ordered table sets. An example of an ordered table set is the EXPORT table set, which lists several hundred tables sorted in foreign key sequence (top-down). Tables in the table_set_detail table may be prefixed by NNNNNNN for the purpose of sequencing the tables. For example:

```
SQL> select table_name from table_set_detail where table_set = 'EXPORT' order
by table_name;
```

```
TABLE_NAME
```

```
-----
0000000.RATE_OPERAND
0001000.ACCESSORIAL_BASIS_PRECEDENCE
0002000.ACCESSORIAL_CODE
0003000.RATABLE_OPERATOR
0004000.RATE_GEO_COST_OPERAND
0005000.COUNTRY_ZONE
0006000.COUNTRY_CODE
0007000.CURRENCY
0008000.DIM_RATE_FACTOR
0009000.ACCESSORIAL_COST
0010000.RATE_UNIT_BREAK_PROFILE
```

As you can see, the tables are prefixed by NNNNNNN in order to ensure they are sequenced within the table set. When you export a table set, you normally pipe it to a remote system. If you do not pipe it to a remote system, it will generate a bunch of temporary files on the source system and leave them there.

Here is a sample screen shot showing how you would normally export the EXPORT table set and pipe it to a remote system.

Export Object Type
 Export Table

Output Destination
 Remote Instance

Run Job in Background
 Y

Use Select List
 N

Run

Command
 XCSV

Table Name
 ACTION

Exclude Public
 Y

Where Clause

Remote Host
 localhost

Remote User
 DBA.ADMIN

Remote Password
 ●●●●●●

From Domain

To Domain

Remote Command
 ii

Remote Host Version
 5.5

Email Address
 youremail@email.com

SMTP Host
 mail.email.com

Run

Figure 12-1

Notice that the above screen requests background processing by specifying an email address and SMTP Host.

13. Copying Rates between Databases 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.

Step 1 – Create a `csvutil.ctl` file (CSVUtil Control 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.

Step 2 – Use the Integration Upload Screen to 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.

Step 3 – Download the Zip File Containing the Rate Offering

When you receive the email, download the zip file containing the rate offering, and extract the `rate_geo_out.csv` file.

Step 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
```

Step 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"`.

Step 6 – Upload the zip file from Step 5 to the target instance

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.

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

This will tell CSVUtil to ignore deprecated columns.

14. Importing Voyage Schedule Data

You can import ocean schedules from a variety of portals, like ESG, CargoSmart, INTTRA, and GTNexus.

Assuming you want to load data from multiple providers into separate partitions, load the data from the first provider in the staging tables. Once complete, the data should be moved to the database in the first partition. After the first data set is complete, the data from the second provider should be loaded in the staging tables. After that, the data should be moved to the database in the second partition. This would continue until all the data is loaded.

1. Acquire voyage schedule data. While the data from some providers is available in the correct Oracle Transportation Management format, you need to ensure that the format is correct prior to loading it in the staging tables.
2. Setup Mapping of Data Sources and Partition Keys
This step is optional, but it makes it easier for you to see what partition key goes with what data provider.

```
pkg_voyage.setup_data_source ('PROVIDER1',1)
```

- PROVIDER1 is the name of the data provider
- 1 is the partition number.

Repeat this step to assign data from a second data provider to partition 2 and so on.

There is a maximum of seven partitions available. It is the VOYAGE and VOYLOC tables that are partitioned, not the staging tables. It is possible to combine multiple data source providers in a single partition. This requires you to load data from all data providers in that partition prior to initiating the loading process.

3. Load the Mapping Tables using normal CSV functionality

Table 14-1

Table	Description
X_VOY_LOC_MAP	mapping of data source location IDs to Oracle Transportation Management locations GIDs
X_VOY_CAR_MAP	mapping of data source service provider IDs to Oracle Transportation Management Service Provider GIDs

4. Load the Staging Tables using normal CSV functionality
Load data into the X_VOYAGE and X_VOYLOC tables. The DATA_SOURCE column of the tables should be set to the appropriate data source ID. The data must contain the complete set of voyage data.

5. Delete the current voyage schedules and load the new data set from the staging tables.

```
pkg_voyage.load_schedule (null,200,'Y')
```

- The first parameter is null because the procedure will look up the partition key using the mapping previously setup. If you did not map data sources to partition keys, you need to make sure to load each data provider's data set in a separate partition.
- 200 defines the batch size in terms of the number of records the database should hold in its buffer before it writes them to the database

- 'Y' states that errors will be logged to the log file

The Load Schedule procedure takes the new data from the X_VOYAGE and X_VOYLOC tables by cross-referencing service provider IDs with the X_VOY_CAR_MAP table and the location IDs with the X_VOY_LOC_MAP table. Note that if a mapping is missing, the procedure creates a new location using the location ID as the GID, and adds a mapping record to the X_VOY_CAR_MAP or X_VOY_LOC_MAP as needed.

6. View Error Log

```
select * from error_log
```

If logging was enabled, and there were any problems during the above steps, a message will be posted to the error log.

7. View Data Mappings

If logging was enabled, the current mapping of data source and partition keys can be viewed by executing the following command using a SQL editor:

```
select * from data_source_partition_view
```

Deleting Schedules

pkg_voyage also contains the following:

Table 14-2

Purpose	Procedure	Parameters
Delete all the data in a specified partition.	delete_schedule	p_partkey (PLS_INTEGER)
Delete all the data from a specified data provider. Note that the name of this procedure is the same as the preceding one. The parameter, however, is different.	delete_schedule	p_dataSource (VARCHAR2)

15. Copying Domains

Note: While copying domains, make sure no user accesses the database. You can do this by shutting down the application server.

Note: If you want to copy domain1 that needs data from domain2 and you want domain1 to have access to all data in domain2 in the new target database too, you need to make sure you copy domain2 before domain1.

This chapter describes a set of tools to copy domains. Each of them has its limitations and advantages.

Table 15-1

Tool	Advantages	Limitations	Usage
In Schema Copy	Data in Clob and long columns can be copied.	You must rename the copied domain. For tables that have a domain_name column and a numeric primary key, the primary key will increment utilizing sequence numbers. However, its copied child fk column data still points to the old from_domain_name parent.	Only within one database.
Database Link Copy	Preferred tool to build a clean database out of an existing database. Tables in your target and source domain can contain different columns.* You can rename or keep the copied domain name. You can run this script multiple times to insert rows that have been added in the source database since the last database link copy.	Requires that a public database link can be created from the target database to the source database. Allows you to copy Clob and long columns. However, the data types in the local and remote domains must be the same.	Within or between databases.

* Tables might contain different columns if you migrated your source database from an earlier database version and you create your target database with the create_all script. In this case, your migrated database contains obsolete columns since the migration scripts do not generally drop obsolete columns.

In Schema Copy

This tool allows you to copy domains within one database. You can copy domains with or without their child domains. Child domains keep their original child domain names; only the parent domain name part is replaced.

What the Objects do

This tool uses these stored procedures in pkg_novpd_domain_copy:

Table 15-2

Procedure	Does This
set_copy_parameters	Stores what "from_domain" to copy into what "to_domain". You can enter multiple pairs of domains before executing the actual copying.
print_copy_parameters	Displays the list of "from_domain"s and "to_domain"s you have created with set_copy_parameters.
reset_parameters	Clears the list of domains to copy. You might want to do this if you notice a spelling error.

Set-up

Compile the packages.

1. Log in as glogowner.
2. sqlplus>@create_pkg_novpd_inschema_copy.sql to compile the package.
3. sqlplus>@novpd_domain_copy_script_builder.sql to generate the domain copy script, novpd_load.sql. In novpd_load.sql, there is a procedure for every table. Each procedure is enclosed by "declare" and a "/". You can remove a procedure from the script if you do not want to copy a certain table.

Copy Domains

1. sqlplus>execute pkg_novpd_domain_copy.set_copy_parameters('from_domain', 'to_domain', copy_child_domains, domain_info) to set your copy parameters.

Note: You need to execute this command for every domain to be copied.

Note: You can copy multiple domains with or without renaming them in a single run.

Note: If domains depend on each other for data and you want to rename at least one of these domains, you must copy all these domains in a single run. This will allow novpd_load.sql to keep all dependencies correct. If you do not, some of the data will be rejected due to foreign key violations.

Table 15-3

Parameter	Description
from_domain	Name of the domain to copy.
to_domain	The new name of the domain.
copy_child_domains	true - child domains will be copied false - child domains will not be copied.

Parameter	Description
domain_info	You must enter "false". This retrieves domain information from the local source database instead of a remote target database.

2. sqlplus>set serverout on size 1000000.
3. sqlplus>execute pkg_novpd_domain_copy.print_copy_parameters to display the parameter values entered.
4. If you notice that any of your parameters are wrong, you can reset all parameters with execute pkg_novpd_domain_copy.reset_parameters. If you do execute this statement, you must re-enter all your parameters.
5. sqlplus>@novpd_load.sql to copy all domains you have entered parameters for. There is a log file: inschema_domain_copy.log.
6. sqlplus>execute domainman.reset_sequence to reset the Oracle sequence numbers.
7. You need to restart Oracle Transportation Management running against the target database to be able to log in to your newly copied domain. The restart allows Oracle Transportation Management to refresh its caches.

Result of In Schema Copy

After novpd_load.sql has finished it displays the number of data rows that were copied and rejected.

Database Link Copy

This is the preferred tool to build a clean database out of an existing database.

Database Link Copy requires the packages pkg_novpd_domain_copy that depends on pkg_domain_export. It copies tables with the domain_name column as well as the security tables.

The total number of rows copied or rejected is written in the database_link_domain_copy.log file. If an exception happens, the exception code as well as the primary key is also written to the log file. Furthermore, if the exception is a foreign key violation, the log will include the foreign key.

Create Link from Target to Source Database

1. Log in to the **target** database with a DBA level account. You must then navigate to the /glog/oracle/script8/ directory.
2. sqlplus>alter system set global_names=false
3. sqlplus>create public database link "loader.oracle.com" connect to "username_in_source_database" identified by "password_in_source_database" using 'source_database'

Example: create public database link "loader.oracle.com" connect to "glogowner" identified by "glogowner" using 'hera35'

Use exact double or single quotes as shown above.

Later, if you need to change the database link to point to a different database, you must first drop the database link (drop public database link loader.oracle.com) and then recreate it.

4. Sqlplus>select count(1) from shipment@loader.oracle.com to confirm that the database link is active.

Generate Script

1. `Sqlplus>connect glogowner/password@targetdb`
2. `Sqlplus>@create_pkg_novpd_inschema_copy.sql`
3. `Sqlplus>@database_link_domain_copy_script_builder.sql` to generate the `link_load.sql` script. `link_load.sql` contains a stored procedure for every table it will copy. Each procedure is enclosed by "declare" and a "/".

Note: You can remove a procedure from the `link_load.sql` script if you do not want to copy a certain table. Note that once you remove a procedure for a table, its child tables are rejected.

Note: Like `novpd_load.sql`, `link_load.sql` only contains a stored procedure for tables that the two databases have in common. Furthermore, only data in columns that appear in both the target and source database will be copied. This allows you to copy domains between databases of different releases.

Note: You may encounter some problems

1. When uncopied columns are required and have no default values or triggers.
2. When the same column in both target and source database has different data types such as CLOB and LONG.
3. When data records in your domain point to records in a domain that do not exist in the target domain. You will see error 2291 in your log file (foreign key violation).
4. When the sequence number of your source database is higher than your target database. If any of the records in your copied domain refers to a table with only a sequence number as primary key, the referring record will be rejected.

Copy Domains

Note: During the domain copy, only one commit per table and domain is executed. If you want to copy a large amount of data, be sure to allocate enough rollback tablespace and segments.

1. `sqlplus>execute pkg_novpd_domain_copy.set_copy_parameters('from_domain', 'to_domain', copy_child_domains, domain_info)` to set your copy parameters.

Note: You need to execute this command for every domain to be copied.

Note: You can copy multiple domains with or without renaming them in a single run.

Note: If domains depend on each other for data and you want to rename at least one of these domains, you must copy all these domains in a single run. This will allow `link_load.sql` to keep all dependencies correct. If you do not, some of the data will be rejected due to foreign key violations.

Table 15-4

Parameter	Description
<code>from_domain</code>	Domain name in source database.
<code>to_domain</code>	Domain name in target database. If it is the same as <code>from_domain</code> , then no renaming is performed. Otherwise, <code>from_domain</code> would be renamed to <code>to_domain</code> during the copying.

Parameter	Description
copy_child_domains	true - then child domains will be copied false - child domains will not be copied.
domain_info	You must enter "true". This retrieves domain information from the remote source database instead of the local target database.

2. sqlplus>set serverout on size 1000000.
3. sqlplus>execute pkg_novpd_domain_copy.print_copy_parameters to display the parameter values entered.
4. If you notice that any of your parameters are wrong, you can reset all parameters with execute pkg_novpd_domain_copy.reset_parameters. If you do execute this statement, you must re-enter all your parameters.
5. sqlplus>@link_load.sql to copy all domains you have entered parameters for. There is a log file: database_link_domain_copy.log
6. sqlplus>execute domainman.reset_sequence to reset the Oracle sequence numbers.
7. You need to restart Oracle Transportation Management running against the target database to be able to log in to your newly copied domain. The restart allows Oracle Transportation Management to refresh its caches.

Difference Between Domains

You can find the difference between two domains and list the primary keys.

1. Sqlplus>set serverout on size 1000000
2. sqlplus>execute pkg_domain_export.diff_remote(*remote_domain*, *local_domain*)
Note: Differences here, most likely depends on static data missing, in your target database, in a domain like PUBLIC. Also, you might have missed to copy dependant domains in one session.
3. sqlplus>execute pkg_domain_export.diff_table_remote(*remote_domain*, *local_domain*)

Rerun database link copy

As long as the target database schema has not changed, you can run the link_load.sql script again and again to insert rows that have been added to the source database since the last database link copy. This is also useful to keep PUBLIC domains in two databases synchronized. Note that this does not update existing rows in the target database

Note: If the target database schema has changed, you need to run the database_link_domain_copy_script_builder.sql script again to create an updated link_load.sql script.

16. Deleting Domains

This chapter describes the steps to delete domains in Oracle Transportation Management.

1. Shut down the Oracle Transportation Management application. This includes WebLogic, Tomcat, Apache, etc.
2. Log in directly to the database using a database management utility such as SQLPLUS. Log into the database as `glogowner`.
3. Delete a single domain. Enter the following command at the SQLPLUS prompt:

```
Exec domainman.delete_domain('DOMAIN');
```

Note: Substitute the domain name that you want to delete for DOMAIN. Since this does a cascade delete, this may take a significant amount of time. If there is any data cross-referenced between domains, the data referenced will not be deleted. For example, if Shipments in DomainA reference rates in DomainB, and you delete DomainB, rates in DomainB referenced by shipments in DomainA can not be deleted.

4. Delete mutiple domains. Enter the following commands at the SQLPLUS prompt:

```
Exec domainman.mark_domain_for_delete ('DOMAIN', including_sub_domains);
```

Note: Substitute the domain name that you want to delete for DOMAIN. Including_sub_domains equals "true" or "false". If it is "true", then the child domains are also marked for deleted. Otherwise, the child domains are not included for deletion. This procedure should be called for each domain to be deleted. Every time the procedure is called, the domain and its child domains are cached in memory. If you make a mistake, you have to log out the session and re-log in.

```
Exec domainman.delete_marked_domains;
```

Note: This procedure iterates through all the domains and child domains marked in the previous step. It deletes one table at a time for the domains and their children. It yields better performance. Futhermore, it can delete cross-referenced data within domains in this transaction.

5. Delete non-existent domain data. Enter the following commands at the SQLPLUS prompt:

```
Set serverout on size 1000000  
Exec domainman.report_unreferenced_domains;
```

Note: This procedure reports all the non-existent domains table by table. The non-existent domains are the ones which are not in domain table. They could be from a bug from previous delete domain procedure or the result from loading a CSV. After reviewing the report generated from the previous step, you can call the next procedure to delete the data in all the tables for the non-existent domains.

```
Exec domainman.delete_unreferenced_domains;
```


17. Reference A: DB.XML Transaction Codes

When importing db.xml with any of the methods described in this document, there are six transaction codes currently available:

- I - Insert Mode: Only inserts are performed. If the data already exists in the database, you will get primary key errors.
- IU - Insert/Update Mode: Attempts to insert data. If a primary key violation occurs, it updates the data. No delete statements are generated.
- RC - Replace Children Mode: 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.
- II: Insert/Ignore Dupes: Attempts to insert a new record but only if it does not exist already.
- U: Update only existing records.
- 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_DATE
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_DATE
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,,
```

Sample command line:

```
java glog.database.admin.CSVUtil -command rc -connectionId localdb -dataDir .
-dataFileName geo_hierarchy.csv
```

In version 5.5 and later, the "rc" command is available after you upload a CSV file via the integration manager.

18. Reference B: Specifying Complex Queries

This section shows the SQL query corresponding to the predefined rate_geo database object.

Example of a Complex Query

Use this example to build your own complex queries when no predefined database object exists for the data you want to export.

```
select rate_geo.*, \
  cursor (select rate_geo_stops.* from rate_geo_stops where
    rate_geo_stops.rate_geo_gid = rate_geo.rate_geo_gid) as rate_geo_stops, \
  cursor (select rate_geo_accessorial.* from rate_geo_accessorial where
    rate_geo_accessorial.rate_geo_gid = rate_geo.rate_geo_gid) as
    rate_geo_accessorial, \
  cursor (select rg_special_service.* from rg_special_service where
    rg_special_service.rate_geo_gid = rate_geo.rate_geo_gid) as
    rg_special_service, \
  cursor (select rg_special_service_accessorial.* from
    rg_special_service_accessorial where
    rg_special_service_accessorial.rate_geo_gid = rate_geo.rate_geo_gid) as
    rg_special_service_accessorial, \
  cursor (select rate_geo_cost_group.*, \
    cursor (select rate_geo_cost.*, \
      cursor (select rate_geo_cost_weight_break.* \
        from rate_geo_cost_weight_break \
        where rate_geo_cost_weight_break.rate_geo_cost_seq =
          rate_geo_cost.rate_geo_cost_seq and
          rate_geo_cost_weight_break.rate_geo_cost_group_gid =
          rate_geo_cost.rate_geo_cost_group_gid) as rate_geo_cost_weight_break
        \
      from rate_geo_cost \
      where rate_geo_cost.rate_geo_cost_group_gid =
        rate_geo_cost_group.rate_geo_cost_group_gid ) as rate_geo_cost \
    from rate_geo_cost_group \
    where rate_geo.rate_geo_gid = rate_geo_cost_group.rate_geo_gid) as
    rate_geo_cost_group \
  from rate_geo "
```

The main thing to notice is the use of nested cursors to specify a hierarchical query.

19. Reference C: 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:
 - a. ProcessCount: The number of rows that were successfully processed
 - o ErrorCount: The number of rows where an error was detected
 - o 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>
<DataDir>.\</DataDir>
<DataFileName>weight_break.csv</DataFileName>
<ProcessCSV>
<TableName>WEIGHT_BREAK</TableName>
<ColumnList>WEIGHT_BREAK_GID,WEIGHT_BREAK_XID,WEIGHT_BREAK_PROFILE_GID,WEIGHT_B
REAK_MAX,WEIGHT_BREAK_MAX_UOM_CODE,WEIGHT_BREAK_MAX_BASE,DOMAIN_NAME</ColumnLis
t>
<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>
<DataDir>.\</DataDir>
<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 **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 **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>JUNK</TableName>
<Exception>ORA-00942: table or view does not exist
</Exception>
<Data>"Data1", "Data2", "Data3"</Data>
</Error>
```

Import

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.

Table 19-1

Heading	Data
Message:	<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.

Table 19-2

Heading	Data
Message:	<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: <ul style="list-style-type: none"> • 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.

Table 19-3

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

Heading	Data
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.

Table 19-4

Heading	Data
Message:	<pre><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></pre>
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.

Table 19-5

Heading	Data
Message:	<pre><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></pre>
Occurs When:	Trying to insert a string in a numeric field.

Heading	Data
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.

Table 19-6

Heading	Data
Message:	<pre><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></pre>
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.

Table 19-7

Heading	Data
Message:	<pre><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, "MYDOMAIN1234567890123456789012345678901234567890123456789012345678901234567890" </Data> </Error></pre>

Heading	Data
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.

Table 19-8

Heading	Data
Message:	<pre><Error> <TableName>WEIGHT_BREAK</TableName> <RowsProcsted>0</RowsProcsted> <Data>"MYDOMAIN.LT 4500","LT 4500","MYDOMAIN.DEFAULT",4500,"LB",4500,"MYDOMAIN"</Data> </Error></pre>
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

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.

Table 19-9

Heading	Data
Message:	<pre><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 ...</pre>
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.

20. Appendix: General Properties

See the “Advanced Configuration: Customer-defined Properties” chapter in the Administration Guide for instructions on how to manage reserved properties.

Table 20-1: General Properties

Property	Reserved	Description
glog.database.admin.csvUtilScript.csvOutput	Yes	<p>Oracle Transportation Management can export CSV data into XML comment section if the property value is set to COMMENTS. However, the string "--" must not occur within XML comments. In order to use "--" in data, set this property to CDATA. Oracle Transportation Management can either export CSV data in XML comment section or in XML CDATA section. By default, Oracle Transportation Management 6.4.3 exports CSV data into CDATA section; this is a change from previous releases, which defaulted to COMMENTS.</p> <p>Default: CDATA</p>
<p>glog.database.archive.dbserver = host (default: \$dbserver\$)</p> <p>glog.database.archive.sid = service (default: \$glog.database.sid\$)</p> <p>glog.database.archive.port = port (default: \$glog.database.port\$)</p> <p>glog.database.archive.user = user (default: archive_c)</p> <p>glog.database.archive.password = password (default: {warchive.archive_c})</p>	Yes	<p>Use these properties to build a connection string to the data source, "ARCHIVE.ARCHIVE" that points to the ARCHIVE_C schema. Assuming the ARCHIVE_C schemas is located in the same database as GLOGOWNER, the installer needs only to add the archive_c user password to the shared wallet with key archive.archive_c, or override the glog.database.archive.password password in glog.properties.</p>
glog.migrationProject.dir	Yes	<p>Path where the migration project packages should reside for import and export.</p> <p>Default: <otm_home>/glog/integration/projects</p>

