Oracle8*i*™ *inter*Media Locator

User's Guide and Reference

Release 8.1.5

February 1999

Part No. A67298-01

Oracle8*i* *inter*Media Locator is a component of Oracle8*i* *inter*Media, a product designed to manage multimedia Web content within Oracle8*i*.

ORACLE®

Enabling the Information Age™

Oracle8*i* *inter*Media Locator User's Guide and Reference

Part No. A67298-01

Release 8.1.5

Primary Authors: Rod Ward

Contributors: Frank Wang, Jeff Hebert, Susan Shepard, Brenda Silva, Deborah Laderoute

Contents

List of Tables

Send Us Your Comments

Oracle8*i* *inter*Media Locator User's Guide and Reference, Release 8.1.5

Part No. A67298-01

Oracle Corporation 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 chapter, section, and page number (if available).

You can send comments to us in the following ways:

- e-mail: nedc_doc@us.oracle.com
- FAX - 603.897.3316. Attn: Oracle8*i* *inter*Media Locator Documentation
- postal service:
  Oracle Corporation
  Oracle8*i* *inter*Media Locator Documentation
  One Oracle Drive
  Nashua, NH 03062
  USA

If you would like a reply, please include your name, and a postal or e-mail address.

Preface

This guide describes how to use Oracle8*i* *inter*Media Locator.

Oracle8*i* *inter*Media Locator requires Oracle8*i* or Oracle8*i* Enterprise Edition.

For information about the differences between Oracle8*i* and Oracle8*i* Enterprise Edition and the features and options that are available to you, see *Getting to Know Oracle8i*.

Intended Audience

This guide is intended for anyone who is interested in storing, retrieving, and manipulating locator point data in an Oracle database, including developers of locator specialization services.

Structure

This guide contains the following chapters and appendixes:

Chapter 1	Introduces Oracle8*i* *inter*Media Locator; explains locator-related concepts.
Chapter 2	Describes the Oracle8*i* *inter*Media Locator functions, the geocoding service, and the locator operator, along with examples of their use.
Appendix A	Describes how to run the sample application and includes a source listing of that program.
Appendix B	Lists exceptions raised and potential errors, their causes, and user actions to correct them.

Related Documents

Note: For information added after the release of this guide, refer to the online README.TXT file in your *ORACLE_HOME* directory. Depending on your operating system, this file may be in:

```
ORACLE_HOME/md/doc/README.TXT
```

Please see your operating-system specific installation guide for more information.

For more information about using this product in a development environment, see the following documents in the Release 8.1.5 Oracle8*i* documentation set:

- *Getting to Know Oracle8i*
- *Oracle8i Application Developer's Guide - Fundamentals*
- *Oracle8i Administrator's Guide*
- *Oracle8i Error Messages*
- *Oracle8i Utilities*
- *Oracle8i Concepts*
- *Oracle8i Tuning*
- *SQL*Plus User's Guide and Reference*

Conventions

In this guide, Oracle8*i* *inter*Media Locator is sometimes referred to as *inter*Media Locator.

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The following conventions are also used in this guide:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.

Convention	Meaning
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
boldface text	Boldface text indicates a term defined in the text, the glossary, or in both locations.
italic text	Italic text is used for emphasis, for book titles, and variable names.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1 Introduction

Oracle8*i* *inter*Media Locator is a component of the Oracle8*i* *inter*Media product. Oracle8*i* *inter*Media Locator enables Oracle8*i* to support online internet-based geocoding facilities for locator applications and proximity queries.

Geocoding represents addresses and locations of interest (postal codes, demographic regions, and so forth) as geometric factors (points). These enable distances to be calculated and sites to be represented graphically in Web, data warehousing, customer information system, and enterprise resource planning applications. Geocoding services can be used to add the exact location (latitude and longitude) of points of interest to existing data files stored in Oracle8*i*.

Oracle8*i* *inter*Media Locator supports the leading online geocoding services including Centrus from QMSoft and MapXtreme from MapInfo. See the Oracle8*i* *inter*Media Locator Release Notes for additional information about the geocoding services provided by these Oracle Partners.

Oracle8*i* *inter*Media Locator also supports server-based geocoding and data scrubbing operations for data warehouse applications.

Using simple location queries, Oracle8*i* *inter*Media Locator allows Web and other applications to retrieve information based on distance. For example, using a set of geocoded address data and simple query-by-text or query-by-map operations, users can use a Web browser-based application, enter a distance, and identify the nearest location from a specific address or reference point on a map. For example, Oracle8*i* *inter*Media Locator applications can help you locate stores, offices, distribution points, and other points of interest based on their distance from a given postal (zip) code, address, or other reference point.

Oracle8*i* *inter*Media Locator supports geocoding, storage, and retrieval of geocoded, spatial-point data in Oracle8*i* databases. Oracle8*i* *inter*Media Locator is not designed to be an end-user application. It consists of:

- An *inter*Media Locator object type that describes and supports only the point-geometry object type
- A geocode result object type that describes the geocode result definition
- A call interface described by two geocode result functions used for geocoding spatial data that also contains the output geocode result object and the *inter*Media Locator geometry object
- A function to better estimate the index level for use with the spatial locator index for within-distance queries that use a radius distance greater than 100 miles
- A procedure to create a spatial locator index on the column where the spatial information is stored in the geocoded table that is used by the locator operator
- A locator operator that uses geometric intersection algorithms and the spatial locator index for performing within-distance queries

Based on this implementation, this Oracle8*i* *inter*Media Locator release supports:

- Geocoding spatial-point data by providing the means to add a geocoded address column or objects to existing tables and storing it locally in the Oracle8*i* universal database server
- Inserting and retrieving geocoded address data
- Performing simple within-distance text- or map-based queries on the geocoded data

Some example applications for this locator function are the following:

- Locate stores, offices, or distribution points based on their distances from a given reference point such as an address or postal code.
- Locate restaurants or hotels within a given point-to-point distance using a person's specific address or current location on a map, such as at a tourist information center.

These features enable database designers to extend existing application databases with geocoded, spatial-point data, or to build new geocoded spatial-point applications. Web application developers can build specialized web-enabled *inter*Media Locator applications.

For additional information, see the following:

- Chapter 2 describes *inter*Media Locator functions, the geocoding service, and the locator operator along with basic examples of using the Oracle8*i* *inter*Media Locator object types.

- Appendix A describes a number of sample scripts that are installed and that you can modify and run.
- Appendix B describes Oracle8*i inter*Media Locator exceptions and error messages.

2

*inter*Media Locator Functions

2.1 *inter*Media Locator Implementation

The implementation of Oracle8*i* *inter*Media Locator functions consists of a set of object types, an index method type, and an operator on these types. A geometry is stored in a single row in a column of type SDO_GEOMETRY. Spatial index creation and maintenance is done using data definition language (DDL) (CREATE, ALTER, DROP) and data manipulation language (DML) (INSERT, UPDATE, DELETE) statements.

2.1.1 *inter*Media Locator Structures

The geometric description of an *inter*Media Locator object is stored in a single row in a column of type SDO_GEOMETRY. This row is in a user-defined table that has one primary key column (or a set columns that constitute a primary key) and optionally one or more attribute columns.

The object type SDO_GEOMETRY is defined as:

```
Create Type SDO_GEOMETRY as object (
SDO_GTYPE NUMBER,
SDO_SRID NUMBER,
SDO_POINT SDO_POINT_TYPE,
SDO_ELEM_INFO MDSYS.SDO_ELEM_INFO_ARRAY,
SDO_ORDINATES MDSYS.SDO_ORDINATE_ARRAY);
```

The attributes have the following semantics:

- SDO_GTYPE - Indicates the type of the geometry. The valid geometry type is:

  `1 = POINT`
  The geometry type must always be 1.

- SDO_SRID - Spatial reference identifier. This is always NULL.
- SDO_POINT - Is an object type with attributes X, Y, and Z; all of type NUMBER represented as longitude, latitude, and NULL, respectively.
- SDO_ELEM_INFO - Is always NULL.
- SDO_ORDINATES - Is always NULL.

2.2 Results Definition and Geocode Functions

This section contains a description of the geocode result object type definition and the call interface described by two geocode functions as shown in Table 2–1.

Table 2–1 interMedia Locator Functions and Procedures

Type/Function	Description
GEOCODE_RESULT object	Geocode result object definition
GEOCODE1 function	Geocode function that contains a lastline field; but no city, state, or postal code (zip) fields
GEOCODE1 function	Geocode function that contains city, state, and postal code (zip) fields, but no lastline field

GEOCODE_RESULT Object

Purpose

This object describes the geocode result definition.

Syntax

```
create type GEOCODE_RESULT AS OBJECT(
   matchcode varchar2(16),
   firmname  varchar2(512),
   addrline  varchar2(512),
   addrline2 varchar2(512),
   city      varchar2(512),
   state     varchar2(512),
   zip       varchar2(5),
   zip4      varchar2(4),
   lastline  varchar2(512),
   county    varchar2(32),
   block     varchar2(32),
   loccode   varchar2(16),
   cart      varchar2(16),
   dpbc      varchar2(16),
   lotcode   varchar2(16),
   lotnum    varchar2(16)
);
/
```

Parameters

matchcode	Match result, indicating the quality of a match
firmname	Firm name
addrline	Address line 1
addrline2	Address line 2
city	City
state	State
zip	Postal (zip) code
zip4	Plus 4 digit zip code
lastline	City, state, zip code
county	Federal information processing (FIPS) county code
block	Census block identifier
loccode	Location code
cart	Carrier route (postal service)
dpbc	Delivery point bar code
lotcode	Line of travel code
lotnum	Line of travel number

Usage Notes

In their implementation of *inter*Media Locator, geocode vendors may make use of all or most fields in the GEOCODE_RESULT table. See the vendor's documentation for a complete description of this object and the fields used.

Exceptions

Application-specific exceptions:

http_error, -20000

geocoder_error, -20001

unit_error, -20003

GEOCODE1 Function (with lastline field)

Purpose

This function is used for geocoding and includes a lastline field that contains city, state, and zip code information.

Syntax

```
function GEOCODE1(url       in varchar2,
                  proxy     in varchar2,
                  name      in varchar2,
                  pwd       in varchar2,
                  firmname  in varchar2,
                  addrline  in varchar2,
                  addrline2 in varchar2,
                  lastline  in varchar2,
                  mm        in varchar2,
                  stdaddr   out MDSYS.GEOCODE_RESULT,
                  location  out MDSYS.SDO_GEOMETRY) return varchar2;
pragma restrict_references(GEOCODE1, WNDS, WNPS);
```

Parameters

url	Vendor Web site for geocoding: for example, www.centrus-software.com/oracle/geoservice.dll
proxy	Security protection mechanisms (firewall) address, NULL or '' if none
name	Customer name, (for accounting)
pwd	Password (for accounting)
firmname	Firm name
addrline	Address line 1
addrline2	Address line 2
lastline	Contains city, state, postal (zip) code, and zip4 information
mm	Matchmode; a string telling the vendor which match mode to use, such as STANDARD, NORMAL, and so forth. See vendor sites for more information.
stdaddr	Standard address object or output geocode result object (defined previously)
location	Locator geometry object, SDO_GEOMETRY, containing latitude and longitude information

Return Value

This return value is the error code returned as a string by the geocode vendor; typically, the string contains an error code and a message, such as 0:SUCCESS. See the specific vendor documentation for more information.

Usage Notes

The lastline field contains the city, state, and postal (zip) code information.

Exceptions

None.

Examples

Example 1: Geocode a single record interactively.

```
-- Geocode a single record interactively.
set serveroutput on
set timing on
set pagesize 50000

declare
  geo_result MDSYS.GEOCODE_RESULT;
  geom MDSYS.SDO_GEOMETRY;
  result varchar2(255);
begin
  result := geocoder_http.GEOCODE1(
              'http://www.centrus-software.com/oracle/geoservice.dll',
              'www-proxy.us.acme.com',
              'user', 'password',
              'oracle','1 oracle dr','', 'nashua NH 03062',
              'tight',
              geo_result, geom);
  dbms_output.put_line(result);
exception
when geocoder_http.http_error then
   dbms_output.put_line('Internet problem - cannot connect');
when geocoder_http.geocoder_error then
   dbms_output.put_line('Geocoder problem - contact vendor');
when others then
   dbms_output.put_line('Oracle Error - check your PL/SQL');
end;
/
```

Example 2: Geocode a table in batch mode using the entire object.

```
-- See how to create this sample table using the file nh_cs.sql
-- Geocode a table in batch mode using the entire object.

-- HOW TO CUSTOMIZE IT FOR YOUR USE:
-- 1. Change the select statement in declaration section to match
--    your input table;
--    If you are placing the geocode result into the same table, make sure
--     rowid is selected; if you are geocoding into a different table, make sure
--    the primary keys are selected.
--
-- 2. In the update call at the end, if you are placing all your results
--    back to the same table, use update ... where rowid = r.rowid;
--    otherwise, use insert into ... where pk = r.pk;
--
-- 3. Exception handling:
```

```
--    The routine generates http_error and geocoder_error.
--    HTTP_ERROR corresponds to transmission problem.
--    GEOCODER_ERROR is when an address record cannot be matched by the
--     geocoder from the vendor Web site, and the result you get back is likely
--    to be null.
--    You should decide how to handle these errors according to your
--    own needs.
--    The GEOCODER_ERROR exception can be examined in the result variable.
--
declare
  CURSOR crs is
     select company, address, city, state, zipcode, rowid from
nh_computer_stores;
  standard_address MDSYS.GEOCODE_RESULT;
  geom_location MDSYS.SDO_GEOMETRY;
  result varchar2(255);
begin
  for r in crs loop
   begin
    result := geocoder_http.GEOCODE1(
      'http://www.centrus-software.com/oracle/geoservice.dll',
      'www-proxy.us.acme.com',
      'user','password',
      r.company,
      r.address, '',
      r.city, r.state, r.zipcode,
      'normal',
      standard_address,
      geom_location);
   exception
   when geocoder_http.geocoder_error then
      dbms_output.put_line('Geocoder error, continuing');
   when others then
      dbms_output.put_line('HTTP or server error, quit');
      exit;
   end;
   update nh_computer_stores
      set std_addr = standard_address, location = geom_location
      where rowid = r.rowid;
<<end_loop>>
    null;
  end loop;
end;
/
```

Example 3: Geocode a table in batch mode using fields in the object.

```
-- Geocode a table in batch mode using fields in the object.

-- HOW TO CUSTOMIZE IT FOR YOUR USE:
-- 1. Change the select statement in declaration section to match
--    your input table;
--    If you are placing the geocode result into the same table, make sure
--     rowid is selected; if you are geocoding into a different table, make sure
--    the primary keys are selected.
--
-- 2. In the update call at the end, if you are placing all your results
--    back to the same table, use update ... where rowid = r.rowid;
--    otherwise, use insert into ... where pk = r.pk;
--
-- 3. Exception handling:
--    The routine generates http_error and geocoder_error.
--    HTTP_ERROR corresponds to transmission problem.
--    GEOCODER_ERROR is when an address record cannot be matched by the
--     geocoder from the vendor Web site, and the result you get back is likely
--    to be null.
--    You should decide how to handle these errors according to your
--    own needs.
--    The GEOCODER_ERROR exception can be examined in the result variable.
--
declare
  CURSOR crs is
     select company, address, city, state, zipcode, rowid from
nh_computer_stores;
  standard_address MDSYS.GEOCODE_RESULT;
  geom_location MDSYS.SDO_GEOMETRY;
  result varchar2(255);
begin
  for r in crs loop
   begin
    result := geocoder_http.GEOCODE1(
      'http://www.centrus-software.com/oracle/geoservice.dll',
      'www-proxy.us.acme.com',
      'user','password',
      r.company,
      r.address, '',
      r.city, r.state, r.zipcode,
      'normal',
      standard_address,
      geom_location);
   exception
   when geocoder_http.geocoder_error then
      dbms_output.put_line('Geocoder error, continuing');
   when others then
      dbms_output.put_line('HTTP or server error, quit');
      exit;
   end;
   update nh_computer_stores
      set std_street = standard_address.address,
      std_city = standard_address.city,
      std_state = standard_address.state,
      std_zip = standard_address.zip,
      std_zip4 = standard_address.zip4,
      location = geom_location
      where rowid = r.rowid;
<<end_loop>>
    null;
  end loop;
end;
/
```

GEOCODE1 Function (with city, state, and postal code (zip) fields)

Purpose

This function is used for geocoding and includes city, state, and postal (zip) code fields.

Syntax

```
function GEOCODE1(url       in varchar2,
                  proxy     in varchar2,
                  name      in varchar2,
                  pwd       in varchar2,
                  firmname  in varchar2,
                  addrline  in varchar2,
                  addrline2 in varchar2,
                  city      in varchar2,
                  state     in varchar2,
                  zip       in varchar2,
                  mm        in varchar2,
                  stdaddr   out MDSYS.GEOCODE_RESULT,
                  location  out MDSYS.SDO_GEOMETRY) return varchar2;
pragma restrict_references(GEOCODE1, WNDS, WNPS);
```

Parameters

url	Vendor Web site for geocoding: for example, www.centrus-software.com/oracle/geoservice.dll
proxy	Security protection mechanisms (firewall) address, NULL or '' if none
name	Customer name, (for accounting)
pwd	Password (for accounting)
firmname	Firm name
addrline	Address line 1
addrline2	Address line 2
city	City name
state	State name
zip	Postal (zip) code
mm	Matchmode; a string telling the vendor which match mode to use, such as STANDARD, NORMAL, and so forth See vendor sites for more information.
stdaddr	Standard address object or output geocode result object (defined previously)
location	Locator geometry object, SDO_GEOMETRY, containing latitude and longitude information

Return Value

The return value is the error code returned as a string by the geocode vendor; typically, the string contains an error code and a message, such as 0:SUCCESS. See the specific vendor documentation for more information.

Usage Notes

The city, state, and postal (zip) fields replace the lastline field described in the previous function.

Exceptions

None.

Examples

See the examples in the previous GEOCODE1 function description.

2.3 Estimate Level and Spatial Locator Index

This section describes the ESTIMATE_LEVEL function and the spatial locator index. If you must use the ESTIMATE_LEVEL function, call this function prior to creating the spatial locator index. The spatial locator index must be created before you can use the locator operator described in Section 2.4.

Table 2–2 interMedia Locator ESTIMATE_LEVEL Function and Spatial Locator Index

Function/Procedure	Description
ESTIMATE_LEVEL	Estimates an appropriate index_level parameter value when most of your LOCATOR_WITHIN_DISTANCE queries use a radius distance value that exceeds 100 miles.
SETUP_LOCATOR_INDEX	Creates the spatial locator index.

ESTIMATE_LEVEL

Purpose

This function calculates an index_level parameter value for use in the SETUP_LOCATOR_INDEX procedure.

Note: Only call this function if most of your LOCATOR_WITHIN_DISTANCE queries use a radius distance value greater than 100 miles; otherwise, the default value of 13 is appropriate as the index_level parameter value.

Syntax

```
function ESTIMATE_LEVEL(radius1 in number,
                        radius2 in number) return integer;
```

Parameters

radius1	Small radius in miles.
radius2	Large radius in miles.

Return Value

The return value is the appropriate index_level parameter value to use in the SETUP_LOCATOR_INDEX procedure.

Usage Notes

If you expect to use a large radius distance for queries that is greater than 100 miles, you should call this function to determine the most appropriate index_level parameter value for your data.

A LOCATOR_WITHIN_DISTANCE query with a circular radius distance greater than 100 miles actually degenerates into an ellipse with two semiaxes (radii). Therefore, this function has two parameters, radius1 to represent the small semiaxis and radius2 to represent the large semiaxis of the ellipse. For Oracle8*i* Release 8.1.5, you should provide the same value for both radius1and radius2 parameters.

If you must call this function, call this function after you geocode your data and before you create your spatial locator index. A more appropriate index_level parameter value is expected to give you better performance on your data.

Exceptions

Application-specific exceptions:

unit_error, -20004

Examples

Create a setup spatial locator index.

```
select geocoder_http.estimate_level(200,200) from dual;
9
```

SETUP_LOCATOR_INDEX

Purpose

This procedure creates the spatial locator index.

Syntax

```
procedure SETUP_LOCATOR_INDEX(tabname in varchar2,
                              colname in varchar2,
                              index_level in number := 13);
```

Parameters

tabname	Table name where the spatial information is stored
colname	Column name where the spatial information is stored within ’tabname’
index_level	Value determined by calling the ESTIMATE_LEVEL function when the radius distance exceeds 100 miles and a better index level is required to improve performance on your data The default value is 13.

Return Value

None.

Usage Notes

This procedure creates a metadata table called SDO_GEOM_METADATA under the invoker’s or current user’s schema. It creates a special domain index of type spatial_locator_index. The name of the index is:

```
substr((tabname,1,5)||’_’substr(colname,1,5)||’_idx’||_HL6N1$
```

Do not delete these extra tables after creating the index.

This procedure must be executed to create the spatial locator index for the geocoded table before you can use the LOCATOR_WITHIN_DISTANCE operator; otherwise, an error message is returned indicating no spatial locator index is created. For example:

```
ERROR at line 1:
ORA-20000: Interface Not Supported without a Spatial Index
ORA-06512: at "MDSYS.SDO_3GL", line 184
ORA-06512: at line 1
```

Usually, you do not need to modify the value of the index_level parameter if most of your LOCATOR_WITHIN_DISTANCE queries are using a radius distance value of 100 miles or less. However, to achieve better performance on your data, you can change this value depending on the most popular radius distance for most of your LOCATOR_WITHIN_DISTANCE queries. To estimate a better value for the index_level parameter, call the ESTIMATE_LEVEL function. In this case, you must call the ESTIMATE_LEVEL function before you create your spatial locator index.

Exceptions

None.

Examples

Create a setup spatial locator index.

```
procedure SETUP_LOCATOR_INDEX('cust_table', 'location', 13);
```

2.4 Locator Operator

This section describes the function used when working with the *inter*Media Locator object type.

Table 2–3 interMedia Locator Operator

Function	Description
LOCATOR_WITHIN_DISTANCE	Determines if two points are within a specified geometric distance from one another.

LOCATOR_WITHIN_DISTANCE

Purpose

This operator uses geometric intersection algorithms and a spatial index to identify the set of spatial points that are within some specified geometric distance (radius distance) of a given point of interest (center of a circle).

Syntax

```
LOCATOR_WITHIN_DISTANCE(T.Column MDSYS.SDO_GEOMETRY, aGeom MDSYS.SDO_GEOMETRY,
params VARCHAR2) ;
```

Parameters

params	Determines the behavior of the operator
	Valid keywords and their semantics are described as follows:
distance	Required; the radius distance value
units	Required; the unit value; can be mile, ft (feet), or meter

Return Value

The expression LOCATOR_WITHIN_DISTANCE(arg1, arg2, arg3) = 'TRUE' evaluates to TRUE for point pairs that are within the specified distance apart, and FALSE otherwise.

Usage Notes

- The distance around a point of interest describes a circle and this distance is defined as the minimum radius distance between these two points.
- The operator must always be used in a WHERE clause and the condition that includes the operator should be an expression of the form:

```
LOCATOR_WITHIN_DISTANCE(arg1, arg2,
'distance = <some_dist_val>, units=mile') = 'TRUE'.
```

- It is required that T.Column have a spatial locator index built on it. See Section 2.3 for more information.
- LOCATOR_WITHIN_DISTANCE( ) is not supported for spatial joins.
- The default unit is latitude and longitude. Therefore, you should always specify a unit such as: mile, ft, or meter.

Exceptions

None.

Examples

Example 1: Simple point query.

```
SELECT A.GID FROM POINTS A WHERE LOCATOR_WITHIN_DISTANCE
(A.Geometry, :aGeom, ‘distance = 10 units=mile’) = ‘TRUE’ ;
```

Example 2: Computer store query.

```
Rem
Rem $Header: geolocate.sql 14-sep-98.11:51:16 pfwang Exp $
Rem
Rem geolocate.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem

-- This routine dynamically creates a geometry of interest,
-- for example, Oracle office location. Then it queries against the
-- NH_COMPUTER_STORES table to find out how many computer stores are
-- within a certain distance radius of the office. In this case, 10 miles.

set serveroutput on
set pagesize 50000

declare
  standard_address MDSYS.GEOCODE_RESULT;
  geom_location    MDSYS.SDO_GEOMETRY;
  result           varchar2(255);
  type cur_type is ref cursor ;
  crs cur_type;
begin
  result := geocoder_http.geocode1(
              'http://www.centrus-software.com/oracle/geoservice.dll',
              ’www-proxy.us.acme.com’,
              ’user’, ’password’,
              ’Oracle’,’1 Oracle Drive’,’’, ’03062’,
              ’tight’, standard_address, geom_location);
  if (instr(upper(result),’SUCCESS’) = 0) then
     raise geocoder_http.geocoder_error;
  end if;
  open crs for
    ’select company from nh_computer_stores where ’||
       ’MDSYS.LOCATOR_WITHIN_DISTANCE(location, :1, ’’distance=10
units=Mile’’)=’’TRUE’’’
  using geom_location;
  loop
    fetch crs into result;
    exit when crs%NOTFOUND;
    dbms_output.put_line(result);
  end loop;
  close crs;
exception
when geocoder_http.http_error then
   dbms_output.put_line(’Internet problem - cannot connect’);
when geocoder_http.geocoder_error then
   dbms_output.put_line(’Geocoder problem - contact vendor’);
when others then
   dbms_output.put_line(’Oracle Error - check your PL/SQL’);
end;
/
```

A Sample Programs

Oracle8*i inter*Media Locator includes a number of scripts that you can modify and run.

A.1 Sample Scripts

Sample Oracle8*i inter*Media Locator scripts are available in the following directory after you install this product:

```
$ORACLE_HOME/md/demo/geocoder
```

These scripts consist of the following files:

- geohttp.sql

  This file contains two parts. One part is for running a geocode function in interactive mode and the other is for running the geocode function in batch mode.

  – Interactive mode.

    See Example 1 in “GEOCODE1 Function (with lastline field)” on page 2-6 for a listing of this part of the file.

  – Batch mode.

    You must update the setup tables in the nh_cs.sql file before you run the geohttp.sql in batch mode. See Example 2 in “GEOCODE1 Function (with lastline field)” on page 2-7 or Example 3 in “GEOCODE1 Function (with lastline field)” on page 2-9 for a listing of this part of the file.

- geoindex.sql

  This file contains:

  - A function named ESTIMATE_LEVEL to better estimate the index level for use with the spatial locator index for within-distance queries that use a radius distance greater than 100 miles. See the example in "ESTIMATE_LEVEL" on page 2-15 for a listing of this file.
  - A procedure statement named SETUP_LOCATOR_INDEX that builds a setup spatial locator index on the location column that contains the spatial information within the cust_table table where the spatial information is stored. See the example in "SETUP_LOCATOR_INDEX" on page 2-17 for a listing of this file.

- geolocate.sql

  This file contains a routine that dynamically creates a geometry of interest and then queries against the NH_COMPUTER_STORES table to find out how many stores are within a 10-mile radius of the office. See Example 2 in "LOCATOR_WITHIN_DISTANCE" on page 2-19 for a listing of this file.

B

Exceptions and Error Messages

B.1 Exceptions

This appendix describes the geocode HTTP package exceptions.

B.1.1 Geocode HTTP Package Exceptions

The following exceptions are associated with the geocode HTTP package.

http_error EXCEPTION
PRAGMA EXCEPTION_INIT(http_error, -20000)

Cause: This exception is raised when an HTTP transmission error occurs.

Action: The HTTP server may be down or the communications link may be down. Try again several times until successful or try again later.

geocoder_error EXCEPTION
PRAGMA EXCEPTION_INIT(geocoder_error, -20001)

Cause: This exception is raised when a geocode vendor error occurs. This error is raised when a row cannot be matched by the geocode vendor and the result returned is likely to be null.

Action: Check with the specific vendor returning this exception to help diagnose the underlying problem and determine an alternative solution.

unit_error EXCEPTION
PRAGMA EXCEPTION_INIT(unit_error, -20003)

Cause: This exception is raised when a unit conversion error occurs.

Action: A unit value is not recognized. Check your unit value for compliance.

radius_error EXCEPTION
PRAGMA EXCEPTION_INIT(unit_error, -20004)

Cause: This exception is raised when a negative radius value is used.

Action: Change the radius value to a positive value.

Index