

Oracle Reader/Writer

FORMAT NOTES:

- This format is not supported by FME Base Edition.

ORACLE® VERSION:

- Any references to Oracle 8i throughout this chapter are also applicable to Oracle 9i and Oracle 10g.

ORACLE INSTANT CLIENT:

- Instant Client can be used to run your OCI, OCCI, JDBC, and ODBC applications without installing a full Oracle Client. Instant Client supports SQL*Plus. For more information on how it works with FME, see http://www.fmepedia.com/index.php/Oracle_Instant_Client.

Overview

The Oracle Reader/Writer allows the Feature Manipulation Engine (FME) to read and write attribute data stored using Oracle. This module communicates directly with Oracle for maximum throughput.

Oracle Spatial is also supported by FME:

- The object-relational model is documented in the *Oracle Spatial Object Reader/Writer*.
- The relational model is documented in *Oracle Spatial Relational Reader/Writer*.

If only attributes are to be read or written, then this Oracle Database reader and writer module of FME should be used. In addition, an `OracleQueryFactory` is available to extract data from an Oracle database within the FME factory pipeline.

Tip:

See the `QueryFactory` in the FME Functions and Factories manual. This factory also exploits the powerful query capabilities of Oracle Spatial.

See the `@SQL` function, also in the FME Functions and Factories manual. This function allows arbitrary Structured Query Language (SQL) statements to be executed against any Oracle database.

Oracle Quick Facts

Format Type Identifier	ORACLE8I_DB
Reader/Writer	Both
Licensing Level	Professional
Dependencies	None
Dataset Type	Database
Feature Type	Table name
Typical File Extensions	N/A
Automated Translation Support	Yes
User-Defined Attributes	Yes
Coordinate System Support	No
Generic Color Support	No
Spatial Index	Never
Schema Required	Yes
Transaction Support	Yes
Geometry Type	oracle_type

Geometry Support			
Geometry	Supported?	Geometry	Supported?
aggregate	no	point	no
circles	no	polygon	no
circular arc	no	raster	no
donut polygon	no	solid	yes
elliptical arc	no	surface	yes
ellipses	no	text	no
line	no	z values	n/a
none	yes		

Reader Overview

FME considers an Oracle dataset to be a database containing a collection of relational tables. The tables must be defined in the mapping file before they can be read. Arbitrary *WHERE* clauses and joins are fully supported. When using the object-relational model, an entire arbitrary SQL *SELECT* statement may also be used as a source of results.

Reader Directives

The directives listed below are processed by the Oracle Database reader. The suffixes listed are prefixed by the current <ReaderKeyword> in a mapping file. By default, the <ReaderKeyword> for the Oracle Database reader is ORACLE8I_DB.

DATASET**Required/Optional:** *Required*

This specifies the SQL/Net service name for the Oracle database, which can be blank to use the default service. If it is specified, then the service must have been set up in the local SQL/Net configuration.

Example:

```
ORACLE8I_DB_DATASET citySource
```

USER_NAME**Required/Optional:** *Optional*

The name of user who will access the database.

```
ORACLE8I_DB_USER_NAME bond007
```

If the database is configured to use an external authentication adapter (such as Windows NT or Kerberos authentication), the username may be left blank, or may be completely omitted.

PASSWORD**Required/Optional:** *Required*

The password of the user accessing the database.

```
ORACLE8I_DB_PASSWORD moneypenny
```

If the database is configured to use an external authentication adapter (such as Windows NT or Kerberos authentication), the password may be left blank, or may be completely omitted.

WORKSPACE**Required/Optional:** *Optional*

The name of the Oracle Workspace Manager workspace which will be used by the reader. All tables read by the reader will be read using the same workspace. If this parameter is omitted, or left blank, the default LIVE workspace will be used.

```
ORACLE8I_DB_WORKSPACE B_focus_1
```

DEF**Required/Optional:** *Optional*

The syntax of the definition is:

```
ORACLE8I_DB_DEF <tableName> \
  [oracle_where_clause <whereClause>] \
  [oracle_sql <sqlQuery>] \
  [oracle_table_writer_mode (inherit_from_writer|insert| \
    update|delete)] \
  [<fieldName> <fieldType>] +
```

The `<fieldType>` of each field must be given, but it is not verified against the database definition for the field. In effect, it is ignored.

The `<tableName>` must match a table in the Oracle database. This will be used as the feature type of all the features read from the table.

The definition allows specification of separate search parameters for each table. If any of the configuration parameters are given, they will override, for that table, whatever global values have been specified by the reader directives listed above. If any of these parameters is not specified, the global values will be used.

The following table summarizes the definition line configuration parameters:

Parameter	Contents
<code>oracle_where_clause</code>	This specifies the SQL WHERE clause applied to the attributes of the layer's features to limit the set of features returned. If this is not specified, the value of the <code><Reader-Keyword>_WHERE_CLAUSE</code> directive is used.
<code>oracle_sql</code>	This specifies an SQL SELECT query to be used as the source for the results. If this is specified, the Oracle Database reader will execute the query, and use the resulting rows as the features instead of reading from the table <code><layerName></code> . All returned features will have a feature type of <code><layerName></code> , and attributes for all columns selected by the query. The <code>oracle_where_clause</code> are ignored if <code>oracle_sql</code> is supplied.
<code>oracle_table_writer_mode</code>	The the default operation mode of the feature type in terms of the types of SQL statements sent to the database. Valid values are INSERT, UPDATE, DELETE and INHERIT_FROM_WRITER. Note that INSERT mode allows for only INSERT operations where as UPDATE and DELETE can be overwritten at the feature levels. INHERIT_FROM_WRITER simply indicates to take this value from the writer level and not to override it at the feature type level. Default: INHERIT_FROM_WRITER

If no `<whereClause>` is specified, all rows in the table will be read and returned as individual features. If a `<whereClause>` is specified, only those rows which are selected by the clause will be read. Note that the `<whereClause>` does not include the word WHERE.

When using the object model, the FME allows one to use the `oracle_sql` parameter to specify an arbitrary SQL SELECT query. If this is specified, the FME will execute the query, and use each row of data returned from the query to define a feature. Each of these features will be given the feature type named in the DEF line, and will contain attributes for every column returned by the SELECT. In this case, all DEF line parameters regarding a WHERE clause are ignored, as it is possible to embed this information directly in the text of the `<sqlQuery>`.

The following example joins the tables ROADS and ROADNAMES, placing the resulting data into FME features with a feature type of MYROADS. Imagine that ROADS defines some

attributes for the roads, and has a numeric field named `ID`, and that `ROADNAMES` joins the numeric field `ID` with character arrays with the roads' names.

```
ORACLE8I_DB_DEF MYROADS \
  oracle_sql "SELECT * FROM ROADS, \
             ROADNAMES WHERE ROADS.ID = ROADNAMES.ID"
```

IDs

Required/Optional: *Optional*

This optional specification is used to limit the available and defined database tables files that will be read. If no `IDs` are specified, then all defined and available tables are read. The syntax of the `IDs` directive is:

```
ORACLE8I_DB_IDS <featureType1> \
  <featureType2> ... \
  <featureTypeN>
```

The feature types must match those used in `DEF` lines.

The example below selects only the `ROADS` table for input during a translation:

```
ORACLE_IDS ROADS
```

WHERE_CLAUSE

Required/Optional: *Optional*

This specifies an SQL `WHERE` clause, which is applied to the table's columns to limit the resulting features. This feature is currently limited to apply only to the attributes of the target table, and does not allow for joining multiple tables together. The effect of table joins can be achieved by specifying the entire queries in the `DEF` line with an `oracle_sql` parameter.

By default, there is no `WHERE` clause applied to the results, so all features in the table are returned.

CHUNK_SIZE

Required/Optional: *Optional*

The features are read from the Oracle database using a bulk reading technique to maximize performance. Normally 1000 rows of data are read from the database at a time.

This directive allows one to tune the performance of the reader. It specifies how many rows are read from the database at a time.

BEGIN_SQL{n}

Required/Optional: *Optional*

Occasionally one must execute some ad-hoc SQL prior to opening an Oracle table. For example, it may be necessary to ensure that a view exists prior to attempting to read from it.

Upon opening a connection to read from an Oracle database, the Oracle reader looks for the directive `<ReaderKeyword>_BEGIN_SQL{n}` (for $n=0, 1, 2, \dots$), and executes each such directive's value as an SQL statement on the database connection.

Any errors occurring during the execution of these SQL statements will normally terminate the reader with an error. If the specified statement is preceded by a hyphen ("-"), such errors are ignored.

END_SQL{n}

Required/Optional: *Optional*

Occasionally one must execute some ad-hoc SQL after closing a set of Oracle tables. For example, it may be necessary to clean up a temporary view after writing to the database.

Just prior to closing a connection on an Oracle database, the Oracle reader looks for the directive `<ReaderKeyword>_END_SQL{n}` (for $n=0, 1, 2, \dots$), and executes each such directive's value as an SQL statement on the database connection.

Any errors occurring during the execution of these SQL statements will normally terminate the reader with an error. If the specified statement is preceded by a hyphen ("-"), such errors are ignored.

REMOVE_SCHEMA_QUALIFIER

Required/Optional: *Optional*

Specifies whether to keep or remove the schema qualifier. The full name of a table in an Oracle database is of the format `<schema_name>.<table_name>`. Setting this keyword to `YES` indicates that the reader should return the table name without any prefixes. This is useful when:

- creating a workspace that will be passed on to another organization using the same table names,

When this keyword is set to `YES` during the generation of a mapping file or workspace, the source feature types will be the table names without any prefix; otherwise, they will contain the owner name as a prefix. It is recommended that this keyword not be changed in value after generating the mapping file/workspace as it is possible for no features to be successfully passed onto the writer (since the writer is expecting feature types with different names).

Note that even when `REMOVE_SCHEMA_QUALIFIER` is set to `YES`, if the table is owned by a user other than the current user, the `<owner_name>` prefix will **not** be dropped so that the reader will find the correct table; however, the `<database_name>` prefix will still be dropped.

Value: `YES` | `NO`

Default Value: `NO`

Example:

```
ORACLE8I_DB_REMOVE_SCHEMA_QUALIFIER YES
```

USE_UNIFIED_DATE_ATTRS**Required/Optional:** *Optional*

Specifies whether we want to use unified date attributes, where the date and time are read into one attribute, or whether we want to use split date attributes, where two attributes are produced, one with only the date and another with both the date and time.

The value of this keyword should not be changed. It is automatically set to YES in new mapping files and workspaces. To maintain backwards compability, if this keyword is not present, the reader will behave as though the keyword is set to NO.

Value: *YES | NO***Default Value:** *YES (in new mapping files and workspaces), NO otherwise*

Writer Overview

The Oracle Database writer module stores attribute data in an Oracle database. The Oracle Database writer provides the following capabilities:

- **Transaction Support:** The Oracle Database writer provides transaction support that eases the data loading process. Occasionally, a data load operation terminates prematurely due to data difficulties. The transaction support provides a mechanism for reloading corrected data without data loss or duplication.
- **Table Creation:** The Oracle Database writer uses the information within the FME mapping file to automatically create database tables as needed.
- **Table Dropping:** The Oracle Database writer has an option that allows each table to be written to be dropped if recreating, or truncated if appending.
- **Index Creation:** The Oracle Database writer will set up and populate all needed indexes and index tables as part of the loading process.
- **Bulk Loading:** The Oracle Database writer uses a bulk loading technique to ensure speedy data load.

Writer Directives

The directives processed by the Oracle Database writer are listed below. The suffixes shown are prefixed by the current `<WriterKeyword>` in a mapping file. By default, the `<WriterKeyword>` for the Oracle Database writer is `ORACLE8I_DB`.

DATASET, USER_NAME, WORKSPACE, BEGIN_SQL{ }, and END_SQL{ }

These directives operate in the same manner as they do for the Oracle Database reader.

DEF**Required/Optional:** *Required*

Each Oracle Database table must be defined before it can be written. The general form of an Oracle Database definition statement is:

```
ORACLE8I_DB_DEF <tableName> \
    [oracle_sql           <sqlQuery>] \
```

```

[oracle_update_key_columns <column>[,<column>]...           \
[oracle_delete_key_columns <column>[,<column>]...         \
[oracle_drop_table      (yes|no)]                          \
[oracle_truncate_table  (yes|no)]                          \
[oracle_params          <creationParams>]                 \
[oracle_sequenced_cols  column[:seqname][;column[seqname]]... \
[<fieldName>          <fieldType>]*

```

If the user wishes to create a table, the table definition allows control of the table that will be created. Otherwise, the table definition serves to provide options for inserting, updating or deleting data in an existing table. In each case, some parameters may be unused. The recommended approach is to take advantage of the updated WorkBench GUI to limit mistakes when setting the layer parameters.

If the table already exists in the database, then it is not necessary to list the fields and their types – FME will use the schema information in the database to determine this. If the fields and types are listed, they must match those in the database, however, not all fields must be listed.

If the table does not exist, then the field names and types are used to first create the table. In any case, if a <fieldType> is given, it must be a field type supported by the target database.

The configuration parameters present on the definition line are described in the following table:

Parameter	Contents
oracle_sql	<p>This specifies an SQL <code>INSERT</code> or <code>UPDATE</code> query to be used to define the results. If this is specified, the Oracle Database writer will execute the query, defining one row for each feature from the FME.</p> <p>The values in the query are specified by embedding <code>:attrName</code> in the query itself, where <code>attrName</code> is the name of the FME feature's attribute; for example:</p> <pre>INSERT INTO BLAH VALUES :a, :b</pre> <p>In this example, the attributes named <code>a</code> and <code>b</code> will be taken from each feature written to <code><tableName></code>.</p> <p>The attributes named in the query must be listed on the <code>DEF</code> line so that the FME knows what type to use. There is no necessary or implied correlation between the FME attribute name and the Oracle column name. Take, for example, this <code>UPDATE</code> query: <pre>UPDATE RR SET TEXTSTRING=:mytext WHERE ID=:myid</pre> <p>In this example, the Oracle column named <code>ID</code> is compared to the value of each feature's attribute named <code>myid</code>, and the value of the table's column named <code>TEXTSTRING</code> is set from the feature attribute named <code>mytext</code>.</p> </p>
oracle_params	<p>This specifies additional parameters to be appended to the Oracle <code>CREATE</code> query used to create the output table. It is used to specify table allocation characteristics and the like.</p> <p>If this is specified, it will override the global <code>CREATE_TABLE_PARAMS</code> directive.</p>

Parameter	Contents
<code>oracle_update_key_columns</code>	<p>This instructs the Oracle Database writer to perform an <code>UPDATE</code> operation on the table, rather than performing an <code>INSERT</code>. The argument is a comma-separated list of the columns which are matched against the corresponding FME attributes' values to specify which rows are to be updated with the other attribute values.</p> <p>For example: <code>oracle_update_key_columns ID</code> would have a similar effect to the "UPDATE" example in the above discussion of the <code>oracle_sql</code> directive. In this case, however, the FME attribute is always matched against the Oracle column with the same name. Also, the target table is always the feature type specified in the DEF line.</p> <p>Each column listed with the <code>oracle_update_key_columns</code> directive must be defined with a type on the DEF line, in addition to the columns whose values will be updated by the operation.</p>
<code>oracle_delete_key_columns</code>	<p>This instructs the Oracle Database writer to perform a <code>DELETE</code> operation on the table, rather than performing an <code>INSERT</code>. The argument is a comma-separated list of the columns which are matched against the corresponding FME attributes' values to specify which rows are to be updated with the other attribute values.</p> <p>For example: <code>oracle_delete_key_columns ID</code> In this case, the FME attribute is always matched against the Oracle column with the same name. Also, the target table is always the feature type specified in the DEF line.</p> <p>Each column listed with the <code>oracle_delete_key_columns</code> directive must be defined with a type on the DEF line.</p>
<code>oracle_drop_table</code>	<p>This specifies whether a table should be dropped, if it exists, before being recreated. If the table does not exist, then the operation is ignored and the user is warned. Note that the drop table option is only available when specifying table creation parameters.</p>
<code>oracle_truncate_table</code>	<p>This specifies whether a table should be truncated, if it exists, before data is inserted. If the table does not exist, then the operation is ignored and the user is warned. Note that the truncate table option is only available when not specifying table creation parameters.</p>
<code>oracle_contains_measures</code>	<p>This directs the writer to write measures to the destination table. When this directive is set to yes and the incoming feature does not have any measures, then null values are written. This parameter applies when writing to existing tables. Default is NO.</p>

START_TRANSACTION

Required/Optional: *Optional*

This statement tells the Oracle Database writer module when to start actually writing features into the database. The Oracle Database writer does not write any features until the feature is reached that belongs to `<last successful transaction> + 1`. Specifying a value of zero causes every feature to be output. Normally, the value specified is zero – a non-zero value is only specified when a data load operation is being resumed after failing partway through.

Parameter	Contents
<code><last successful transaction></code>	The transaction number of the last successful transaction. When loading data for the first time, set this value to 0.

Example:

```
ORACLE8I_DB_START_TRANSACTION 0
```

TRANSACTION_INTERVAL

Required/Optional: *Optional*

This statement informs the FME about the number of features to be placed in each transaction before a transaction is committed to the database.

If the `TRANSACTION_INTERVAL` statement is not specified, then a value of 2000 is used as the transaction interval.

Parameter	Contents
<code><transaction_interval></code>	The number of features in a single transaction.

Example:

```
ORACLE8I_DB_TRANSACTION_INTERVAL 5000
```

CHUNK_SIZE

See the `CHUNK_SIZE` directive in the Reader Directives section.

BEGIN_SQL{n}

This directive is described in the *Reader Directives* section. In the case of the writer, the statements will be executed only when the first feature actually written to the dataset.

END_SQL{n}

This directive is described in the *Reader Directives* section. In the case of the writer, the statements will be executed only if at least one feature has been written to the dataset.

STRICT_ATTR_CONVERSION

This directive instructs the Oracle writer on how to proceed when a problem arises while converting a value from one of a feature's attributes to an oracle column value. Examples of such problems would be the truncation of a string value to fit into the target character column, an error in converting a non-numeric attribute to write to a numeric column.

In normal operation, the Oracle writer will silently truncate strings which are too long, or null out values which cannot be successfully converted. It can optionally log features which have conversion problems, or stop the translation entirely when such features are encountered.

Possible values for this directive are summarized in the following table:

Parameter	Contents
NO	Silently ignore conversion errors. (This is the default behaviour.)
YES	Immediately stop the translation with an error when conversion errors are detected.
WARN	Log any features causing conversion errors, and then continue the translation as usual

Example:

```
ORACLE8I_DB_STRICT_ATTR_CONVERSION WARN
```

WRITER_MODE

Required/Optional: *Optional*

Note: For more information on this directive, see the chapter *Database Writer Mode*.

This directive informs the Oracle Database writer which SQL operations will be performed by default by this writer. This operation can be set to `INSERT`, `UPDATE` or `DELETE`. The default writer level value for this operation can be overwritten at the feature type or table level. The corresponding feature type `DEF` parameter name is called `oracle_table_writer_mode`. It has the same valid options as the writer level mode and additionally the value `INHERIT_FROM_WRITER` which causes the writer level mode to be inherited by the feature type as the default for features contained in that table.

The operation can be set specifically for individual feature as well. Note that when the writer mode is set to `INSERT` this prevents the mode from being interpreted from individual features and all features are inserted unless otherwise marked as `UPDATE` or `DELETE` features. These are skipped.

If the `WRITER_MODE` statement is not specified, then a value of `INSERT` is given.

Parameter	Contents
<code><writer_mode></code>	The type of SQL operation that should be performed by the writer. The valid list of values are below: <code>INSERT</code> <code>UPDATE</code> <code>DELETE</code> Default: <code>INSERT</code>

Example:

```
ORACLE8I_DB_WRITER_MODE INSERT
```

Writer Mode Specification

The Oracle Database writer allows the user to specify a writer mode, which determines what database command should be issued for each feature received. Valid writer modes are `INSERT`, `UPDATE` and `DELETE`.

Writer Modes

In `INSERT` mode, the attribute values of each received feature are written as a new database record.

In `UPDATE` mode, the attribute values of each received feature are used to update existing records in the database. The records which are updated are determined via the `oracle_update_key_columns` DEF line parameter, or via the `fme_where` attribute on the feature.

In `DELETE` mode, existing database records are deleted according to the information specified in the received feature. Records are selected for deletion using the same technique as records are selected for updating in `UPDATE` mode.

Writer Mode Constraints

In `UPDATE` and `DELETE` mode, the `fme_where` attribute always takes precedence over the `oracle_update_key_columns` DEF line parameter. If both the `fme_where` attribute and the `oracle_update_key_columns` DEF line parameter are not present, then `UPDATE` or `DELETE` mode will generate an error.

When the `fme_where` attribute is present, it is used verbatim as the `WHERE` clause on the generated `UPDATE` or `DELETE` command. For example, if `fme_where` were set to `'id<5'`, then all database records with field `id` less than 5 will be affected by the command.

When the `fme_where` attribute is not present, the writer looks for the `oracle_update_key_columns` DEF line parameter and uses it to determine which records should be affected by the command. Please refer to DEF on page ??? for more information about the `oracle_update_key_columns` DEF line parameter.

Writer Mode Selection

The writer mode can be specified at three unique levels: on the writer level, on the feature type, or on individual features.

At the writer level, the writer mode is specified by the `WRITER_MODE` keyword. This keyword can be superseded by the feature type writer mode specification. **Note:** For more information on this directive, see the chapter *Database Writer Mode*.

At the feature type level, the writer mode is specified by the `oracle_writer_mode` DEF line parameter. This parameter supersedes the `WRITER_MODE` keyword. Unless this parameter is set to `INSERT`, it may be superseded on individual features by the `fme_db_operation` attribute. Please refer to the DEF line documentation for more information about this parameter.

At the feature level, the writer mode is specified by the `fme_db_operation` attribute. Unless the parameter at the feature type level is set to `INSERT`, the writer mode specified by this attribute always supersedes all other values. Accepted values for the `fme_db_operation` attribute are `INSERT`, `UPDATE` or `DELETE`.

Feature Representation

Features read from Oracle Databases consist of a series of attribute values. They have no geometry. The feature type of each Database feature is as defined on its DEF line.

Features written to the database have the destination table as their feature type, and attributes as defined by on the DEF line.

In addition to the generic FME feature attributes that FME Workbench adds to all features (see *About Feature Attributes* on page 7), the Oracle Database module makes use of the following special attribute names:

Attribute Name	Contents
<code>oracle_type</code>	The type of geometric entity stored within the feature. This is always set to: <code>oracle_nil</code>

Features read from, or written to, Oracle Databases have an attribute for each column in the database table. The feature attribute name will be the same as the source or destination column name. The attribute and column names are case-sensitive.

Troubleshooting

Problems sometimes arise when attempting to connect to an Oracle database. This is almost always due to a misconfiguration in the user's environment. The following suggestions can often help detect and overcome such problems.

- Ensure you can connect to the database with the service name, user name, and password using `SQL*Plus`.
- Ensure that you have the correct version of the Oracle client software installed. Oracle 8.1.5 or newer is recommended. Note that many clients have had problems if they have both 8.0.4 and 8.1.x installed on the same computer.

- Ensure that your `ORACLE_HOME` environment variable is correctly set: see the Oracle documentation for details. This is required for some specific versions of Oracle 8i, and may be required even if `SQL*Plus` appears to operate correctly without it.
- If you have had older versions of the Oracle client software installed, make sure that your `PATH` variable has the current version's Oracle directory **first**, before any other Oracle software, including the WebDB package.
- It is sometimes helpful to define an environment variable named `ORACLE`, with the same value as the `ORACLE_HOME` variable. With some installations, it often helps to ensure that the variable named `ORACLE` is *not* defined.
- When running on UNIX, the following environment variables should be defined:

Variable	Contents	Sample Value
<code>ORACLE_BASE</code>	Top level of directory into which Oracle client software is installed.	<code>/opt2/oracle8i/app/oracle</code>
<code>ORACLE_HOME</code>	The Oracle product directory.	<code>/opt2/oracle8i/app/oracle/product.8.1.5</code>
<code>ORACLE_SID</code>	The system ID for the host's database instance.	<code>FME</code>
<code>LD_LIBRARY_PATH</code>	A list of directories which will be searched for shared objects. This list must include the <code>FME_HOME</code> path, as well as the <code>lib</code> subdirectory of <code>ORACLE_HOME</code> .	<code>\${LD_LIBRARY_PATH}:</code> <code>\${FME_HOME}:</code> <code>\${ORACLE_HOME}/lib</code>

- In most cases, the `ORACLE_SERVER_NAME` and `ORACLE_DATABASE` directives should be left with blank values, with the `ORACLE_DATASET` directive containing the Oracle service name of the database.
- To test connectivity to Oracle, get `ora8ilist.zip` from `ftp://ftp.safe.com/fme/misc` and unzip `ora8ilist.exe` into your FME installation directory. Change to the FME installation directory and execute the following command. On UNIX, it is necessary to first define the environment variables listed above. (The `ora8ilist` script sets up the required environment variables prior to running the actual `ora8ilist` executable.)

```
ora8ilist <service> <username> <password> -
```

For example,

```
ora8ilist BONES scott tiger -
```

Note that the last word must be a hyphen (-). When executed, the list of tables with a geometry column should be output.

The `ora8ilist` program is a simple Oracle Call Interface (OCI) program which connects to the database and executes a single SQL query. If this program cannot be made to work, then FME will not be able to connect to the database either. Safe Software will provide the source code for this program to any party that continues to have problems making a connection to their database to assist in locating the problem with their Oracle configuration.