# 128
# DBMS_METADATA

The DBMS_METADATA package provides a way for you to retrieve metadata from the database dictionary as XML or creation DDL and to submit the XML to re-create the object.

> **See Also:**
>
> *Oracle Database Utilities* for more information and for examples of using the DBMS_METADATA package.

This chapter contains the following topics:


- Overview
- Security Model
- Rules and Limits
- Data Structures - Object and Table Types
- Subprogram Groupings
  - Subprograms for Retrieving Multiple Objects From the Database
  - Subprograms for Submitting XML to the Database
- Summary of All DBMS_METADATA Subprograms


## DBMS_METADATA Overview

You can use the DBMS_METADATA package to retrieve metadata and to submit XML.

**Retrieving Metadata**

If you are retrieving metadata, you can specify:

- The kind of object to be retrieved. This can be either a particular object type (such as a table, index, or procedure) or a heterogeneous collection of object types that form a logical unit (such as a database export or schema export).
- Optional selection criteria, such as owner or name.
- Parse items (attributes of the returned objects to be parsed and returned separately).
- Optional transformations on the output, implemented by XSLT (Extensible Stylesheet Language Transformation) scripts. By default the output is represented in XML, but you can specify transformations (into SQL DDL, for example), which are implemented by XSLT stylesheets stored in the database or externally.

DBMS_METADATA provides the following retrieval interfaces:

- For programmatic use: OPEN, SET_FILTER, SET_COUNT, GET_QUERY, SET_PARSE_ITEM, ADD_TRANSFORM, SET_TRANSFORM_PARAM,SET_REMAP_PARAM, FETCH_xxx, and CLOSE retrieve multiple objects.

- For use in SQL queries and for browsing: `GET_XML`, `GET_DDL` and `GET_SXML` return metadata for a single named object. The `GET_DEPENDENT_XML`, `GET_DEPENDENT_DDL`, `GET_GRANTED_XML`, and `GET_GRANTED_DDL` interfaces return metadata for one or more dependent or granted objects. These procedures do not support heterogeneous object types.

**Submitting XML**

If you are submitting XML, you specify:

- The type of object
- Optional transform parameters to modify the object (for example, changing the object's owner)
- Parse items (attributes of the submitted objects to be parsed and submitted separately)
- Whether to execute the operation or simply return the generated DDL

`DBMS_METADATA` provides a programmatic interface for submission of XML. It is comprised of the following procedures: `OPENW`, `ADD_TRANSFORM`, `SET_TRANSFORM_PARAM`, `SET_REMAP_PARAM`, `SET_PARSE_ITEM`, `CONVERT`, `PUT`, and `CLOSE`.

# DBMS_METADATA Security Model

The `DBMS_METADATA` package considers a privileged user to be one who is connected as user `SYS` or who has the `SELECT_CATALOG_ROLE` role.

The object views of the Oracle metadata model implement security as follows:

- Nonprivileged users can see the metadata of only their own objects.
- Nonprivileged users can also retrieve public synonyms, system privileges granted to them, and object privileges granted to them or by them to others. This also includes privileges granted to `PUBLIC`.
- If callers request objects they are not privileged to retrieve, no exception is raised; the object is simply not retrieved.
- If nonprivileged users are granted some form of access to an object in someone else's schema, they will be able to retrieve the grant specification through the Metadata API, but not the object's actual metadata.
- In stored procedures, functions, and definers-rights packages, roles (such as `SELECT_CATALOG_ROLE`) are disabled. Therefore, such a PL/SQL program can only fetch metadata for objects in its own schema. If you want to write a PL/SQL program that fetches metadata for objects in a different schema (based on the invoker's possession of `SELECT_CATALOG_ROLE`), you must make the program invokers-rights.
- For all objects that have passwords, except database links (for example, users and roles), the following rules apply:
  - A user who has the `SELECT_CATALOG_ROLE` can see all metadata for an object except the passwords for that object.
  - The `SYS` user, users who have the `EXP_FULL_DATABASE` role, and users who own an object can see all metadata for that object, including passwords.
- For database links the password is never displayed. For security reasons Oracle restricts visibility of the password value to `SYS` users who query the `link$.passwordx` column directly. Instead of the password, `DBMS_METADATA` returns the following invalid syntax:

```
IDENTIFIED BY VALUES ':1'
```

A user who knows the password of the database link can manually replace the `:1` with the password.

# Rules and Limits

In an Oracle Shared Server (OSS) environment, the `DBMS_METADATA` package must disable session migration and connection pooling.

This results in any shared server process that is serving a session running the package to effectively become a default, dedicated server for the life of the session. You should ensure that sufficient shared servers are configured when the package is used and that the number of servers is not artificially limited by too small a value for the `MAX_SHARED_SERVERS` initialization parameter.

# DBMS_METADATA Data Structures - Object and Table Types

The `DBMS_METADATA` package defines, in the `SYS` schema, the `OBJECT` and `TABLE` types shown in this code.

```
CREATE TYPE sys.ku$_parsed_item AS OBJECT (
  item            VARCHAR2(30),
  value           VARCHAR2(4000),
  object_row      NUMBER )
/

CREATE PUBLIC SYNONYM ku$_parsed_item FOR sys.ku$_parsed_item;

CREATE TYPE sys.ku$_parsed_items IS TABLE OF sys.ku$_parsed_item
/

CREATE PUBLIC SYNONYM ku$_parsed_items FOR sys.ku$_parsed_items;

CREATE TYPE sys.ku$_ddl AS OBJECT (
   ddlText        CLOB,
parsedItems sys.ku$_parsed_items )
/

CREATE PUBLIC SYNONYM ku$_ddl FOR sys.ku$_ddl;

CREATE TYPE sys.ku$_ddls IS TABLE OF sys.ku$_ddl
/

CREATE PUBLIC SYNONYM ku$_ddls FOR sys.ku$_ddls;

CREATE TYPE sys.ku$_multi_ddl AS OBJECT (
   object_row     NUMBER,
   ddls           sys.ku$_ddls )
/

CREATE OR REPLACE PUBLIC SYNONYM ku$_multi_ddl FOR sys.ku$_multi_ddl;

CREATE TYPE sys.ku$_multi_ddls IS TABLE OF sys.ku$_multi_ddl;
/

CREATE OR REPLACE PUBLIC SYNONYM ku$_multi_ddls FOR
                          sys.ku$_multi_ddls;
```

```
CREATE TYPE sys.ku$_ErrorLine IS OBJECT (
   errorNumber    NUMBER,
   errorText      VARCHAR2(2000) )
/

CREATE PUBLIC SYNONYM ku$_ErrorLine FOR sys.ku$_ErrorLine;

CREATE TYPE sys.ku$_ErrorLines IS TABLE OF sys.ku$_ErrorLine
/
CREATE PUBLIC SYNONYM ku$ErrorLines FOR sys.ku$_ErrorLines;

CREATE TYPE sys.ku$_SubmitResult AS OBJECT (
   ddl          sys.ku$_ddl,
   errorLines   sys.ku$_ErrorLines );
/

CREATE TYPE sys.ku$_SubmitResults IS TABLE OF sys.ku$_SubmitResult
/

CREATE PUBLIC SYNONYM ku$_SubmitResults FOR sys.ku$_SubmitResults;
```

**Note:**

The maximum size of the `VARCHAR2`, `NVARCHAR2`, and `RAW` datatypes has been increased to 32 KB when the `COMPATIBLE` initialization parameter is set to 12.0 and the `MAX_STRING_SIZE` initialization parameter is set to `EXTENDED`. The `DBMS_METADATA` package supports this increased size unless the version of the metadata is earlier than Oracle Database 12*c* Release 1 (12.1).

# DBMS_METADATA Subprogram Groupings

The `DBMS_METADATA` subprograms retrieve objects from, and submit XML to, a database. Some subprograms are used for both activities, while others are used only for retrieval or only for submission.

- Table 128-1 provides a summary, in alphabetical order, of `DBMS_METADATA` subprograms used to retrieve multiple objects from a database.
- Table 128-2 provides a summary, in alphabetical order, of `DBMS_METADATA` subprograms used to submit XML metadata to a database.

## DBMS_METADATA Subprograms for Retrieving Multiple Objects From the Database

`DBMS_METADATA` uses these subprograms used for retrieving multiple objects from the database.

**Table 128-1 DBMS_METADATA Subprograms for Retrieving Multiple Objects**

| Subprogram | Description |
|---|---|
| ADD_TRANSFORM Function | Specifies a transform that `FETCH_xxx` applies to the XML representation of the retrieved objects |

**Table 128-1 (Cont.) DBMS_METADATA Subprograms for Retrieving Multiple Objects**

| Subprogram | Description |
|---|---|
| CLOSE ProcedureCLOSE Procedure | Invalidates the handle returned by `OPEN` and cleans up the associated state |
| FETCH_xxx Functions and Procedures | Returns metadata for objects meeting the criteria established by `OPEN`, `SET_FILTER`, `SET_COUNT`, `ADD_TRANSFORM`, and so on |
| GET_QUERY Function | Returns the text of the queries that are used by `FETCH_xxx` |
| GET_xxx Functions | Fetches the metadata for a specified object as XML, SXML, or DDL, using only a single call |
| OPEN Function | Specifies the type of object to be retrieved, the version of its metadata, and the object model |
| SET_COUNT Procedure | Specifies the maximum number of objects to be retrieved in a single `FETCH_xxx` call |
| SET_FILTER Procedure | Specifies restrictions on the objects to be retrieved, for example, the object name or schema |
| SET_PARSE_ITEM Procedure | Enables output parsing by specifying an object attribute to be parsed and returned |
| SET_TRANSFORM_PARAM and SET_REMAP_PARAM Procedures | Specifies parameters to the XSLT stylesheets identified by `transform_handle` |

## DBMS_METADATA Subprograms for Submitting XML to the Database

`DBMS_METADATA` uses these subprograms for submitting XML to the database.

**Table 128-2 DBMS_METADATA Subprograms for Submitting XML**

| Subprogram | Description |
|---|---|
| ADD_TRANSFORM Function | Specifies a transform for the XML documents |
| CLOSE ProcedureCLOSE Procedure | Closes the context opened with `OPENW` |
| CONVERT Functions and Procedures | Converts an XML document to DDL |
| OPENW Function | Opens a write context |
| PUT Function | Submits an XML document to the database |
| SET_PARSE_ITEM Procedure | Specifies an object attribute to be parsed |
| SET_TRANSFORM_PARAM and SET_REMAP_PARAM Procedures | `SET_TRANSFORM_PARAM` specifies a parameter to a transform<br>`SET_REMAP_PARAM` specifies a remapping for a transform |

# Summary of All DBMS_METADATA Subprograms

This table lists the `DBMS_METADATA` subprograms and briefly describes them.

**Table 128-3 DBMS_METADATA Package Subprograms**

| Subprogram | Description |
|---|---|
| ADD_TRANSFORM Function | Specifies a transform that FETCH_xxx applies to the XML representation of the retrieved objects |
| CLOSE ProcedureCLOSE Procedure | Invalidates the handle returned by OPEN and cleans up the associated state |
| CONVERT Functions and Procedures | Converts an XML document to DDL |
| FETCH_xxx Functions and Procedures | Returns metadata for objects meeting the criteria established by OPEN, SET_FILTER, SET_COUNT, ADD_TRANSFORM, and so on |
| GET_xxx Functions | Fetches the metadata for a specified object as XML, SXML, or DDL, using only a single call |
| GET_QUERY Function | Returns the text of the queries that are used by FETCH_xxx |
| OPEN Function | Specifies the type of object to be retrieved, the version of its metadata, and the object model |
| OPENW Function | Opens a write context |
| PUT Function | Submits an XML document to the database |
| SET_COUNT Procedure | Specifies the maximum number of objects to be retrieved in a single FETCH_xxx call |
| SET_FILTER Procedure | Specifies restrictions on the objects to be retrieved, for example, the object name or schema |
| SET_PARSE_ITEM Procedure | Enables output parsing by specifying an object attribute to be parsed and returned |
| SET_TRANSFORM_PARAM and SET_REMAP_PARAM Procedures | Specifies parameters to the XSLT stylesheets identified by transform_handle |

# ADD_TRANSFORM Function

The DBMS_METADATA.ADD_TRANSFORM function is used for both retrieval and submission.

- When this procedure is used to retrieve objects, it specifies a transform that FETCH_xxx applies to the XML representation of the retrieved objects.
- When used to submit objects, it specifies a transform that CONVERT or PUT applies to the XML representation of the submitted objects. It is possible to add more than one transform.

> **See Also:**
>
> – Subprograms for Retrieving Multiple Objects From the Database
> – Subprograms for Submitting XML to the Database
> – "SET_TRANSFORM_PARAM and SET_REMAP_PARAM Procedures" for information about how to modify and customize transform output

### Syntax

```
DBMS_METADATA.ADD_TRANSFORM (
   handle       IN NUMBER,
```

```
   name         IN VARCHAR2,
   encoding     IN VARCHAR2 DEFAULT NULL,
   object_type  IN VARCHAR2 DEFAULT NULL)
 RETURN NUMBER;
```

### Parameters

**Table 128-4 ADD_TRANSFORM Function Parameters**

| Parameters | Description |
|---|---|
| handle | The handle returned from OPEN when this transform is used to retrieve objects. Or the handle returned from OPENW when this transform is used in the submission of XML metadata. |
| name | The name of the transform. The name can be an internal keyword like DDL to use internally stored stylesheets. If the name contains a colon, it is interpreted as directory_object_name:file_name of a user-supplied Extensible Stylesheet Language Transformation (XSLT) script.<br>Otherwise, name designates a transform implemented by DBMS_METADATA.<br>See Table 128-5 for descriptions of available transforms. |
| encoding | The name of the Globalization Support character set in which the stylesheet pointed to by name is encoded. This is only valid if name is a URL. If left NULL and the URL is external to the database, UTF-8 encoding is assumed. If left NULL and the URL is internal to the database (that is, it begins with /oradb/), then the encoding is assumed to be the database character set. |
| object_type | The definition of this parameter depends upon whether you are retrieving objects or submitting XML metadata.<br>1. When you use ADD_TRANFORM to retrieve objects, the following definition of object_type applies:<br>Designates the object type to which the transform applies. (Note that this is an object type name, not a path name.) By default the transform applies to the object type of the OPEN handle. When the OPEN handle designates a heterogeneous object type, the following behavior can occur:<br>• if object_type is omitted, the transform applies to all object types within the heterogeneous collection<br>• if object_type is specified, the transform only applies to that specific object type within the collection<br>If you omit this parameter you can add the DDL transform to all objects in a heterogeneous collection with a single call. If you supply this parameter, you can add a transform for a specific object type.<br>2. When you use ADD_TRANSFORM in the submission of XML metadata, this parameter is the object type to which the transform applies. By default, it is the object type of the OPENW handle. Because the OPENW handle cannot designate a heterogeneous object type, the caller would normally leave this parameter NULL in the ADD_TRANSFORM calls. |

The following table describes the transforms available on the ADD_TRANSFORM function.

Because new transforms are occasionally added, you might want to query the DBMS_METADATA_TRANSFORMS view to see all valid Oracle-supplied transforms for specific object types.

**Table 128-5 Transforms Available on ADD_TRANSFORM Function**

| Object Type | Transform Name | Input Doc Type | Output Doc Type | Description |
|---|---|---|---|---|
| All | DDL | XML | DDL | Convert XML to SQL to create the object |
| All | MODIFY | XML | XML | Modify XML document according to transform parameters |
| Subset | SXML | XML | SXML | Convert XML to SXML |
| Subset | MODIFYSXML | SXML | SXML | Modify SXML document according to transform parameters |
| Subset | SXMLDDL | SXML | DDL | Convert SXML to DDL |
| Subset | ALTERXML | SXML difference document | ALTER_XML | Generate ALTER_XML from SXML difference document. (See the DBMS_METADATA_DIFF PL/SQL package for more information about SXML difference format.)<br>The following parameters are valid for the ALTERXML transform:<br>• XPATH - The XPATH of the object being altered<br>• NAME - Name of the object being altered<br>• ALTERABLE - Affirms that the object can be altered. If the object cannot be altered, a NOT_ALTERABLE element is inserted whose value indicates the reason.<br>• CLAUSE_TYPE - The type of clause (for example, ADD_COLUMN)<br>• COLUMN_ATTRIBUTE - The attribute being modified<br>• CONSTRAINT_TYPE - The type of constraint (for example, UNIQUE or PRIMARY) |
| Subset | ALTERDDL | ALTER_XML | ALTER_DDL | Convert ALTER_XML to ALTER_DDL |

### Return Values

The opaque handle that is returned is used as input to SET_TRANSFORM_PARAM and SET_REMAP_PARAM. Note that this handle is different from the handle returned by OPEN or OPENW; it refers to the transform, not the set of objects to be retrieved.

### Usage Notes

- With no transforms added, objects are returned by default as XML documents. You call ADD_TRANSFORM to specify the XSLT stylesheets to be used to transform the returned XML documents.

- You can call ADD_TRANSFORM more than once to apply multiple transforms to XML documents. The transforms are applied in the order in which they were specified, the output of the first transform being used as input to the second, and so on.
- The output of a DDL transform is *not* an XML document. Therefore, no transform should be added after the DDL transform.
- Each transform expects a certain format XML document as input. If the input document is unspecified, metadata XML format is assumed.
- When the ALTERXML transform is used, parse items are returned in a PARSE_LIST element of the ALTER_XML document. Each PARSE_LIST_ITEM element contains an ITEM and a VALUE. For example:

```
<PARSE_LIST>
 <PARSE_LIST_ITEM>
  <ITEM>XPATH</ITEM>
  <VALUE>/sxml:TABLE/sxml:RELATIONAL_TABLE/sxml:COL_LIST/sxml:COL_LIST_ITEM[14]</
VALUE>
 </PARSE_LIST_ITEM>
 <PARSE_LIST_ITEM>
  <ITEM>NAME</ITEM>
  <VALUE>Z1</VALUE>
 </PARSE_LIST_ITEM>
 <PARSE_LIST_ITEM>
  <ITEM>CLAUSE_TYPE</ITEM>
  <VALUE>ADD_COLUMN</VALUE>
 </PARSE_LIST_ITEM>
 <PARSE_LIST_ITEM>
  <ITEM>COLUMN_ATTRIBUTE</ITEM>
  <VALUE>NOT_NULL</VALUE>
 </PARSE_LIST_ITEM>
</PARSE_LIST>
```

### Exceptions

- INVALID_ARGVAL. A NULL or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- INVALID_OPERATION. ADD_TRANSFORM was called after the first call to FETCH_xxx for the OPEN context. After the first call to FETCH_xxx is made, no further calls to ADD_TRANSFORM for the current OPEN context are permitted.
- INCONSISTENT_ARGS. The arguments are inconsistent. Possible inconsistencies include the following:
  - encoding is specified even though name is not a URL.
  - object_type is not part of the collection designated by handle.

# CLOSE Procedure

This procedure is used for both retrieval and submission. This procedure invalidates the handle returned by `OPEN` (or `OPENW`) and cleans up the associated state.

> **See Also:**
>
> For more information about related subprograms:
>
> - Subprograms for Retrieving Multiple Objects From the Database
> - Subprograms for Submitting XML to the Database

### Syntax

```
DBMS_METADATA.CLOSE (
   handle  IN NUMBER);
```

### Parameters

**Table 128-6 CLOSE Procedure Parameters**

| Parameter | Description |
|---|---|
| `handle` | The handle returned from `OPEN` (or `OPENW`). |

### Usage Notes

> **Note:**
>
> The following notes apply only to object retrieval

You can prematurely terminate the stream of objects established by `OPEN` or (`OPENW`).

- If a call to `FETCH_xxx` returns `NULL`, indicating no more objects, a call to `CLOSE` is made transparently. In this case, you can still call `CLOSE` on the handle and not get an exception. (The call to `CLOSE` is not required.)
- If you know that only one specific object will be returned, you should explicitly call `CLOSE` after the single `FETCH_xxx` call to free resources held by the handle.

### Exceptions

- `INVALID_ARGVAL`. The value for the `handle` parameter is `NULL` or invalid.

# CONVERT Functions and Procedures

The `DBMS_METADATA.CONVERT` functions and procedures transform input XML documents.

The `CONVERT` functions return creation DDL. The `CONVERT` procedures return either XML or DDL, depending on the specified transforms.

> **See Also:**
>
> - Subprograms for Submitting XML to the Database

### Syntax

The `CONVERT` functions are as follows:

```
DBMS_METADATA.CONVERT (
   handle   IN NUMBER,
   document IN sys.XMLType)
 RETURN sys.ku$_multi_ddls;

DBMS_METADATA.CONVERT (
  handle   IN NUMBER,
  document IN CLOB)
 RETURN sys.ku$_multi_ddls;
```

The `CONVERT` procedures are as follows:

```
DBMS_METADATA.CONVERT (
  handle   IN NUMBER,
  document IN sys.XMLType,
  result   IN OUT NOCOPY CLOB);

DBMS_METADATA.CONVERT (
  handle   IN NUMBER,
  document IN CLOB,
  result   IN OUT NOCOPY CLOB);
```

### Parameters

**Table 128-7 CONVERT Subprogram Parameters**

| Parameter | Description |
|---|---|
| `handle` | The handle returned from `OPENW` |
| `document` | The XML document containing object metadata of the type of the `OPENW` handle |
| result | The converted document |

### Return Values

Either XML or DDL, depending on the specified transforms.

### Usage Notes

You can think of `CONVERT` as the second half of `FETCH_xxx`, either `FETCH_DDL` (for the function variants) or `FETCH_CLOB` (for the procedure variants). There are two differences:

- `FETCH_xxx` gets its XML document from the database, but `CONVERT` gets its XML document from the caller
- `FETCH_DDL` returns its results in a `sys.ku$_ddls` nested table, but `CONVERT` returns a `sys.ku$_multi_ddls` nested table

The transforms specified with `ADD_TRANSFORM` are applied in turn, and the result is returned to the caller. For the function variants, the DDL transform must be specified. If parse items were specified, they are returned in the `parsedItems` column. Parse items are ignored by the procedure variants.

The encoding of the XML document is embedded in its CLOB or XMLType representation. The version of the metadata is embedded in the XML. The generated DDL is valid for the database version specified in `OPENW`.

### Exceptions

- `INVALID_ARGVAL`. A `NULL` or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- `INCONSISTENT_OPERATION`. No transform was specified. The DDL transform was not specified (function variants only).
- `INCOMPATIBLE_DOCUMENT`. The version of the XML document is not compatible with this version of the software.

## FETCH_xxx Functions and Procedures

These functions and procedures return metadata for objects meeting the criteria established by `OPEN`, `SET_FILTER`, `SET_COUNT`, `ADD_TRANSFORM`, and so on.

See "Usage Notes" for the variants.

> **See Also:**
>
> For more information about related subprograms:
>
> - Subprograms for Retrieving Multiple Objects From the Database

### Syntax

The `FETCH` functions are as follows:

```
DBMS_METADATA.FETCH_XML (
   handle  IN NUMBER)
RETURN sys.XMLType;

DBMS_METADATA.FETCH_DDL (
   handle  IN NUMBER)
RETURN sys.ku$_ddls;

DBMS_METADATA.FETCH_CLOB (
   handle       IN NUMBER,
   cache_lob    IN BOOLEAN DEFAULT TRUE,
   lob_duration IN PLS INTEGER DEFAULT DBMS_LOB.SESSION)
RETURN CLOB;
```

The `FETCH` procedures are as follows:

```
DBMS_METADATA.FETCH_CLOB (
   handle  IN NUMBER,
   doc     IN OUT NOCOPY CLOB);

DBMS_METADATA.FETCH_XML_CLOB (
```

```
handle  IN NUMBER,
doc     IN OUT NOCOPY CLOB,
parsed_items OUT sys.ku$_parsed_items,
object_type_path OUT VARCHAR2);
```

### Parameters

**Table 128-8 FETCH_xxx Function Parameters**

| Parameters | Description |
|---|---|
| handle | The handle returned from OPEN. |
| cache_lob | TRUE=read LOB into buffer cache |
| lob_duration | The duration for the temporary LOB created by FETCH_CLOB, either DBMS_LOB.SESSION (the default) or DBMS_LOB.CALL. |
| doc | The metadata for the objects, or NULL if all objects have been returned. |
| parsed_items | A nested table containing the items specified by SET_PARSE_ITEM. If SET_PARSE_ITEM was not called, a NULL is returned. |
| object_type_path | For heterogeneous object types, this is the full path name of the object type for the objects returned by the call to FETCH_XXX. If handle designates a homogeneous object type, a NULL is returned. |

### Return Values

The metadata for the objects or NULL if all objects have been returned.

### Usage Notes

These functions and procedures return metadata for objects meeting the criteria established by the call to OPEN that returned the handle, and subsequent calls to SET_FILTER, SET_COUNT, ADD_TRANSFORM, and so on. Each call to FETCH_xxx returns the number of objects specified by SET_COUNT (or less, if fewer objects remain in the underlying cursor) until all objects have been returned. After the last object is returned, subsequent calls to FETCH_xxx return NULL and cause the stream created by OPEN to be transparently closed.

There are several different FETCH_xxx functions and procedures:

- The FETCH_XML function returns the XML metadata for an object as an XMLType. It assumes that if any transform has been specified, that transform will produce an XML document. In particular, it assumes that the DDL transform has not been specified.
- The FETCH_DDL function returns the DDL (to create the object) in a sys.ku$_ddls nested table. It assumes that the DDL transform has been specified. Each row of the sys.ku$_ddls nested table contains a single DDL statement in the ddlText column; if requested, parsed items for the DDL statement will be returned in the parsedItems column. Multiple DDL statements may be returned under the following circumstances:
  - When you call SET_COUNT to specify a count greater than 1
  - When an object is transformed into multiple DDL statements. For example, A TYPE object that has a DDL transform applied to it can be transformed into both CREATE TYPE and CREATE TYPE BODY statements. A TABLE object can be transformed into a CREATE TABLE, and one or more ALTER TABLE statements
- The FETCH_CLOB function simply returns the object, transformed or not, as a CLOB. By default, the CLOB is read into the buffer cache and has session duration, but these defaults can be overridden with the cache_lob and lob_duration parameters.

- The FETCH_CLOB procedure returns the objects by reference in an IN OUT NOCOPY parameter. This is faster than the function variant, which returns LOBs by value, a practice that involves an expensive LOB copy.
- The FETCH_XML_CLOB procedure returns the XML metadata for the objects as a CLOB in an IN OUT NOCOPY parameter. This helps to avoid LOB copies, which can consume a lot of resources. It also returns a nested table of parse items and the full path name of the object type of the returned objects.
- All LOBs returned by FETCH_xxx are temporary LOBs. You must free the LOB. If the LOB is supplied as an IN OUT NOCOPY parameter, you must also create the LOB.
- If SET_PARSE_ITEM was called, FETCH_DDL and FETCH_XML_CLOB return attributes of the object's metadata (or the DDL statement) in a sys.ku$_parsed_items nested table. For FETCH_XML_CLOB, the nested table is an OUT parameter. For FETCH_DDL, it is a column in the returned sys.ku$_ddls nested table. Each row of the nested table corresponds to an item specified by SET_PARSE_ITEM and contains the following columns:
  - item—the name of the attribute as specified in the name parameter to SET_PARSE_ITEM.
  - value—the attribute value, or NULL if the attribute is not present in the DDL statement.
  - object-row—a positive integer indicating the object to which the parse item applies. If multiple objects are returned by FETCH_xxx, (because SET_COUNT specified a count greater than 1) then object_row=1 for all items for the first object, 2 for the second, and so on.
- The rows of the sys.ku$_parsed_items nested table are ordered by ascending object_row, but otherwise the row order is undetermined. To find a particular parse item within an object row the caller must search the table for a match on item.
- In general there is no guarantee that a requested parse item will be returned. For example, the parse item may not apply to the object type or to the particular line of DDL, or the item's value may be NULL.
- If SET_PARSE_ITEM was not called, NULL is returned as the value of the parsed items nested table.
- It is expected that the same variant of FETCH_xxx will be called for all objects selected by OPEN. That is, programs will not intermix calls to FETCH_XML, FETCH_DDL, FETCH_CLOB, and so on using the same OPEN handle. The effect of calling different variants is undefined; it might do what you expect, but there are no guarantees.
- Every object fetched will be internally consistent with respect to on-going DDL (and the subsequent recursive DML) operations against the dictionary. In some cases, multiple queries may be issued, either because the object type is heterogeneous or for performance reasons (for example, one query for heap tables, one for index-organized tables). Consequently the FETCH_xxx calls may in fact be fetches from different underlying cursors (meaning that read consistency is not guaranteed).

> **Caution:**
>
> Do not run functions and procedures within a query, because they can fail when the handle is for an open context for a heterogeneous object.

### Exceptions

Most exceptions raised during execution of the query are propagated to the caller. Also, the following exceptions may be raised:

- `INVALID_ARGVAL`. A `NULL` or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- `INCONSISTENT_OPERATION`. Either `FETCH_XML` was called when the DDL transform had been specified, or `FETCH_DDL` was called when the DDL transform had *not* been specified.

## GET_xxx Functions

`GET_xxx` functions let you fetch metadata for objects with a single call.

### Functions

The `GET_xxx` functions are:

- `GET_XML`
- `GET_DDL`
- `GET_SXML`
- `GET_DEPENDENT_XML`
- `GET_DEPENDENT_DDL`
- `GET_GRANTED_XML`
- `GET_GRANTED_DDL`

> **See Also:**
>
> Subprograms for Retrieving Multiple Objects From the Database for more information about related subprograms

### Syntax

```
DBMS_METADATA.GET_XML (
object_type     IN VARCHAR2,
name            IN VARCHAR2,
schema          IN VARCHAR2 DEFAULT NULL,
version         IN VARCHAR2 DEFAULT 'COMPATIBLE',
model           IN VARCHAR2 DEFAULT 'ORACLE',
transform       IN VARCHAR2 DEFAULT NULL)
RETURN CLOB;

DBMS_METADATA.GET_DDL (
object_type     IN VARCHAR2,
name            IN VARCHAR2,
schema          IN VARCHAR2 DEFAULT NULL,
version         IN VARCHAR2 DEFAULT 'COMPATIBLE',
model           IN VARCHAR2 DEFAULT 'ORACLE',
transform       IN VARCHAR2 DEFAULT 'DDL')
```

```
RETURN CLOB;

DBMS_METADATA.GET_SXML (
object_type     IN VARCHAR2,
name            IN VARCHAR2 DEFAULT NULL,
schema          IN VARCHAR2 DEFAULT NULL,
version         IN VARCHAR2 DEFAULT 'COMPATIBLE',
model           IN VARCHAR2 DEFAULT 'ORACLE',
transform       IN VARCHAR2 DEFAULT 'SXML')
RETURN CLOB;

DBMS_METADATA.GET_DEPENDENT_XML (
object_type        IN VARCHAR2,
base_object_name   IN VARCHAR2,
base_object_schema IN VARCHAR2 DEFAULT NULL,
version            IN VARCHAR2 DEFAULT 'COMPATIBLE',
model              IN VARCHAR2 DEFAULT 'ORACLE',
transform          IN VARCHAR2 DEFAULT NULL,
object_count       IN NUMBER   DEFAULT 10000)
RETURN CLOB;

DBMS_METADATA.GET_DEPENDENT_DDL (
object_type         IN VARCHAR2,
base_object_name    IN VARCHAR2,
base_object_schema  IN VARCHAR2 DEFAULT NULL,
version             IN VARCHAR2 DEFAULT 'COMPATIBLE',
model               IN VARCHAR2 DEFAULT 'ORACLE',
transform           IN VARCHAR2 DEFAULT 'DDL',
object_count        IN NUMBER   DEFAULT 10000)
RETURN CLOB;

DBMS_METADATA.GET_GRANTED_XML (
object_type     IN VARCHAR2,
grantee         IN VARCHAR2 DEFAULT NULL,
version         IN VARCHAR2 DEFAULT 'COMPATIBLE',
model           IN VARCHAR2 DEFAULT 'ORACLE',
transform       IN VARCHAR2 DEFAULT NULL,
object_count    IN NUMBER   DEFAULT 10000)
RETURN CLOB;

DBMS_METADATA.GET_GRANTED_DDL (
object_type     IN VARCHAR2,
grantee         IN VARCHAR2 DEFAULT NULL,
version         IN VARCHAR2 DEFAULT 'COMPATIBLE',
model           IN VARCHAR2 DEFAULT 'ORACLE',
transform       IN VARCHAR2 DEFAULT 'DDL',
object_count    IN NUMBER   DEFAULT 10000)
RETURN CLOB;
```

### Parameters

**Table 128-9 GET_xxx Function Parameters**

| Parameter | Description |
|---|---|
| object_type | The type of object that you want to be retrieved. This parameter takes the same values as the OPEN object_type parameter, except that it cannot be a heterogeneous object type. The attributes of the object type must be appropriate to the function. That is, for GET_*xxx* it must be a named object. |
| name | The object name. This object name is used internally in a NAME filter. (If the name is longer than 30 characters, then it will be used in a LONGNAME filter.) If this parameter is NULL, then no NAME or LONGNAME filter is specified. See SET_FILTER Procedure for a list of filters. |
| schema | The object schema. This object schema is used internally in a SCHEMA filter. The default is the current user. |
| version | The version of metadata that you want to be extracted. This parameter takes the same values as the OPEN version parameter. |
| model | The object model that you want to use. This parameter takes the same values as the OPEN model parameter. |
| transform | The name of a transformation on the output. This parameter takes the same values as the ADD_TRANSFORM *name* parameter. For GET_XML this name must not be DDL. |
| base_object_name | The base object name. The base object name is used internally in a BASE_OBJECT_NAME filter. |
| base_object_schema | The base object schema. The base object schema is used internally in a BASE_OBJECT_SCHEMA filter. The default is the current user. |
| grantee | The grantee. The grantee is used internally in a GRANTEE filter. The default is the current user. |
| object_count | The maximum number of objects to return. See SET_COUNT Procedure. |

### Return Values

The metadata for the specified object as XML or DDL.

### Usage Notes

- These functions enable you to fetch metadata for objects with a single call. They encapsulate calls to OPEN, SET_FILTER, and so on. The function that you use depends on the characteristics of the object type, and on whether you want XML, SXML, or DDL.
  - GET_*xxx* is used to fetch named objects, especially schema objects (tables, views).
  - GET_DEPENDENT_*xxx* is used to fetch dependent objects (audits, object grants).
  - GET_GRANTED_*xxx* is used to fetch granted objects (system grants, role grants).
- For some object types, you can use more than one function. For example, you can use GET_*xxx* to fetch an index by name, or GET_DEPENDENT_*xxx* to fetch the same index by specifying the table on which it is defined.
- GET_*xxx* only returns a single named object.

- For GET_DEPENDENT_*xxx* and GET_GRANTED_*xxx*, an arbitrary number of dependent or granted objects can match the input criteria. You can specify an object count when fetching these objects. (The default count of 10000 should be adequate in most cases.)
- If the DDL transform is specified, then session-level transform parameters are inherited.
- If you start these functions from SQL*Plus, then to obtain complete, uninterrupted output, you should set the PAGESIZE to 0, and set LONG to some large number.
- Currently, GET_DDL for a USER object does not return the CONTAINER clause.

### Exceptions

- INVALID_ARGVAL. A NULL or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- OBJECT_NOT_FOUND. The specified object was not found in the database.

### Examples

Example: Fetch the XML Representation of SCOTT.EMP

To generate complete, uninterrupted output, set the PAGESIZE to 0 and set LONG to some large number, as shown, before executing your query.

```
SET LONG 2000000
SET PAGESIZE 0
SELECT DBMS_METADATA.GET_XML('TABLE','EMP','SCOTT')
FROM DUAL;
```

Example: Fetch the DDL for all Complete Tables in the Current Schema, Filter Out Nested Tables and Overflow Segments

This example fetches the DDL for all "complete" tables in the current schema, filtering out nested tables and overflow segments. The example uses SET_TRANSFORM_PARAM (with the handle value = DBMS_METADATA.SESSION_TRANSFORM meaning "for the current session") to specify that storage clauses are not to be returned in the SQL DDL. Afterwards, the example resets the session-level parameters to their defaults.

To generate complete, uninterrupted output, set the PAGESIZE to 0 and set LONG to some large number, as shown, before running your query.

```
SET LONG 2000000
SET PAGESIZE 0
EXECUTE
DBMS_METADATA.SET_TRANSFORM_PARAM(DBMS_METADATA.SESSION_TRANSFORM,'STORAGE',fa
lse);
SELECT DBMS_METADATA.GET_DDL('TABLE',u.table_name)
     FROM USER_ALL_TABLES u
     WHERE u.nested='NO'
     AND (u.iot_type is null or u.iot_type='IOT');
EXECUTE
DBMS_METADATA.SET_TRANSFORM_PARAM(DBMS_METADATA.SESSION_TRANSFORM,'DEFAULT');
```

Example: Fetch the DDL For All Object Grants On HR.EMPLOYEES

```
SELECT DBMS_METADATA.GET_DEPENDENT_DDL('OBJECT_GRANT','EMPLOYEES','HR')
    FROM DUAL;
```

Example: Fetch the DDL For System Grants Granted To `SCOTT`

```
SELECT DBMS_METADATA.GET_GRANTED_DDL('SYSTEM_GRANT','SCOTT')
    FROM DUAL;
```

> **Note:**
>
> If you use the package `DBMS_METADATA.GET_GRANTED_DDL` to grant system privileges to a nonprivileged user granted system privileges, and you then attempt to use that nonprivileged user to grant privileges to other users by using `GRANT INHERIT PRIVILEGES`, then you receive the error "ORA-31703: cannot grant `INHERIT PRIVILEGES` privilege on behalf of other users". The `DBMS_METADATA.GET_GRANTED_DDL` procedure returns only privileges directly granted to the user.

Example: Fetch the DDL For All User `SCOTT` Object Grants on `HR.EMPLOYEES`

```
SQL> SELECT DBMS_METADATA.GET_GRANTED_DDL('OBJECT_GRANT','SCOTT')
     FROM dual ;
DBMS_METADATA.GET_GRANTED_DDL('OBJECT_GRANT','SCOTT')
```

Example: Fetch the DDL for the Data Use Case Domain `DAY_OF_WEEK`

```
SQL> SET LONG 2000000
SELECT DBMS_METADATA.GET_DDL('SQL_DOMAIN', 'DAY_OF_WEEK');
```

# GET_QUERY Function

This function returns the text of the queries that are used by `FETCH_xxx`. This function assists in debugging.

> **See Also:**
>
> For more information about related subprograms:
>
> - Subprograms for Retrieving Multiple Objects From the Database

### Syntax

```
DBMS_METADATA.GET_QUERY (
   handle  IN NUMBER)
 RETURN VARCHAR2;
```

### Parameters

**Table 128-10 GET_QUERY Function Parameters**

| Parameter | Description |
|---|---|
| handle | The handle returned from OPEN. It cannot be the handle for a heterogeneous object type. |

### Return Values

The text of the queries that will be used by FETCH_xxx.

### Exceptions

- INVALID_ARGVAL. A NULL or invalid value was supplied for the handle parameter.

## OPEN Function

The DBMS_METADATA.OPEN function specifies the type of object to be retrieved, the version of its metadata, and the object model.

The return value is an opaque context handle for the set of objects to be used in subsequent calls.

> **See Also:**
>
> For more information about related subprograms, see:
>
> Subprograms for Retrieving Multiple Objects From the Database

### Syntax

```
DBMS_METADATA.OPEN (
   object_type  IN VARCHAR2,
   version      IN VARCHAR2 DEFAULT 'COMPATIBLE',
   model        IN VARCHAR2 DEFAULT 'ORACLE',
   network_link IN VARCHAR2 DEFAULT NULL)
 RETURN NUMBER;
```

### Parameters

**Table 128-11 Open Function Parameters**

| Parameter | Description |
|---|---|
| object_type | The type of object to be retrieved. The table "DBMS_METADATA: Object Types" lists the valid object type names and their meanings. These object types are supported for the ORACLE model of metadata (see model in this table).<br>The Attributes column in the DBMS_METADATA: Object Types table specifies some object type attributes:<br>• Schema objects, such as tables, belong to schemas.<br>• Named objects have unique names (if they are schema objects, the name is unique to the schema).<br>• Dependent objects, such as indexes, are defined with reference to a base schema object.<br>• Granted objects are granted or assigned to a user or role and therefore have a named grantee.<br>• Heterogeneous object types denote a collection of related objects of different types. See the table "Object Types Returned for the Heterogeneous Object Type" for a listing of object types returned for the heterogeneous object type.<br>These attributes are relevant when choosing object selection criteria. See "SET_FILTER Procedure" for more information. |
| version | The version of metadata to be extracted. Database objects or attributes that are incompatible with the version will not be extracted. Legal values for this parameter are as follows:<br>COMPATIBLE (default)—the version of the metadata corresponds to the database compatibility level.<br>LATEST—the version of the metadata corresponds to the database version.<br>A specific database version. The value cannot be lower than 9.2.0. |
| model | Specifies which view to use, because the API can support multiple views on the metadata. Only the ORACLE model is supported. |
| network_link | The name of a database link to the database whose metadata is to be retrieved. If NULL (the default), then metadata is retrieved from the database on which the caller is running |

The table Table 128-12 provides the name, meaning, attributes, and notes for the DBMS_METADATA package object types. In the attributes column, S represents a schema object, N represents a named object, D represents a dependent object, G represents a granted object, and H represents a heterogeneous object.

**Table 128-12 DBMS_METADATA: Object Types**

| Type Name | Meaning | Attributes | Notes |
|---|---|---|---|
| AQ_QUEUE | queues | SND | Dependent on table |
| AQ_QUEUE_TABLE | additional metadata for queue tables | ND | Dependent on table |
| AQ_TRANSFORM | transforms | SN | None |
| ASSOCIATION | associate statistics | D | None |

**Table 128-12 (Cont.) DBMS_METADATA: Object Types**

| Type Name | Meaning | Attributes | Notes |
|---|---|---|---|
| AUDIT | audits of SQL statements | DG | Modeled as dependent, granted object. The base object name is the statement audit option name (for example, ALTER SYSTEM). There is no base object schema. The grantee is the user or proxy whose statements are audited. |
| AUDIT_OBJ | audits of schema objects | D | None |
| CLUSTER | clusters | SN | None |
| COMMENT | comments | D | None |
| CONSTRAINT | constraints | SND | Constraints do not include the following:<br>• Primary key constraint for IOT<br>• Column NOT NULL constraints<br>• Certain REF SCOPE and WITH ROWID constraints for tables with REF columns |
| CONTEXT | application contexts | N | None |
| DATABASE_EXPORT | all metadata objects in a database | H | Corresponds to a full database export |
| DB_LINK | database links | SN | Modeled as schema objects because they have owners. For public links, the owner is PUBLIC. For private links, the creator is the owner. |
| DEFAULT_ROLE | default roles | G | Granted to a user by ALTER USER |
| DIMENSION | dimensions | SN | None |
| DIRECTORY | directories | N | None |
| FGA_POLICY | fine-grained audit policies | D | Not modeled as named object because policy names are not unique. |
| FUNCTION | stored functions | SN | None |
| INDEX_STATISTICS | precomputed statistics on indexes | D | The base object is the index's table. |
| INDEX | indexes | SND | None |
| INDEXTYPE | indextypes | SN | None |
| JAVA_SOURCE | Java sources | SN | None |
| JOB | jobs | S | None |
| LIBRARY | external procedure libraries | SN | None |
| MATERIALIZED_VIEW | materialized views | SN | None |
| MATERIALIZED_VIEW_LOG | materialized view logs | D | None |
| OBJECT_GRANT | object grants | DG | None |
| ON_USER_GRANT | Grants | G | Modeled as user grants. Grants the privileges of one user to other user in the form GRANT ... ON USER .... The grantee is the user. Example:<br>GRANT INHERIT PRIVILEGES ON USER "USER1" TO "USER2". |
| OPERATOR | operators | SN | None |

**Table 128-12 (Cont.) DBMS_METADATA: Object Types**

| Type Name | Meaning | Attributes | Notes |
|---|---|---|---|
| PACKAGE | stored packages | SN | By default, both package specification and package body are retrieved. See "SET_FILTER Procedure". |
| PACKAGE_SPEC | package specifications | SN | None |
| PACKAGE_BODY | package bodies | SN | None |
| PROCEDURE | stored procedures | SN | None |
| PROFILE | profiles | N | None |
| PROXY | proxy authentications | G | Granted to a user by ALTER USER |
| REF_CONSTRAINT | referential constraint | SND | None |
| REFRESH_GROUP | refresh groups | SN | None |
| RESOURCE_COST | resource cost info | H | None |
| RLS_CONTEXT | driving contexts for enforcement of fine-grained access-control policies | D | Corresponds to the DBMS_RLS.ADD_POLICY_CONTENT procedure |
| RLS_GROUP | fine-grained access-control policy groups | D | Corresponds to the DBMS_RLS.CREATE_GROUP procedure |
| RLS_POLICY | fine-grained access-control policies | D | Corresponds to DBMS_RLS.ADD_GROUPED_POLICY. Not modeled as named objects because policy names are not unique. |
| RMGR_CONSUMER_GROUP | resource consumer groups | SN | Oracle Data Pump does not use these object types. Instead, it exports resource manager objects as procedural objects. |
| RMGR_INTITIAL_CONSUMER_GROUP | assign initial consumer groups to users | G | None |
| RMGR_PLAN | resource plans | SN | None |
| RMGR_PLAN_DIRECTIVE | resource plan directives | D | Dependent on resource plan |
| ROLE | roles | N | None |
| ROLE_GRANT | role grants | G | None |
| ROLLBACK_SEGMENT | rollback segments | N | None |
| SCHEMA_EXPORT | all metadata objects in a schema | H | Corresponds to user-mode export. |
| SEQUENCE | sequences | SN | None |
| SQL_DOMAIN | Data Use Case Domains | SN | A **Data Use Case Domain** is a high-level dictionary object that belongs to a schema and encapsulates a set of properties and constraints. The attributes and constraints are defined and managed only for the domain, and are automatically applied to all the columns of the given domain. |

**Table 128-12 (Cont.) DBMS_METADATA: Object Types**

| Type Name | Meaning | Attributes | Notes |
|---|---|---|---|
| SYNONYM | synonyms | See notes | Private synonyms are schema objects. Public synonyms are not, but for the purposes of this API, their schema name is PUBLIC. The name of a synonym is considered to be the synonym itself. For example, in CREATE PUBLIC SYNONYM FOO FOR BAR, the resultant object is considered to have name FOO and schema PUBLIC. |
| SYSTEM_GRANT | system privilege grants | G | None |
| TABLE | tables | SN | None |
| TABLE_DATA | metadata describing row data for a table, nested table, or partition | SND | For partitions, the object name is the partition name.<br>For nested tables, the object name is the storage table name. The base object is the top-level table to which the table data belongs. For nested tables and partitioning, this is the top-level table (*not* the parent table or partition). For nonpartitioned tables and non-nested tables this is the table itself. |
| TABLE_EXPORT | metadata for a table and its associated objects | H | Corresponds to table-mode export |
| TABLE_STATISTICS | precomputed statistics on tables | D | None |
| TABLESPACE | tablespaces | N | None |
| TABLESPACE_QUOTA | tablespace quotas | G | Granted with ALTER USER |
| TRANSPORTABLE_EXPORT | metadata for objects in a transportable tablespace set | H | Corresponds to transportable tablespace export |
| TRIGGER | triggers | SND | None |
| TRUSTED_DB_LINK | trusted links | N | None |
| TYPE | user-defined types | SN | By default, both type and type body are retrieved. See "SET_FILTER Procedure". . |
| TYPE_SPEC | type specifications | SN | None |
| TYPE_BODY | type bodies | SN | None |
| USER | users | N | None |
| VIEW | views | SN | None |
| XMLSCHEMA | XML schema | SN | The object's name is its URL (which can be longer than 30 characters). Its schema is the user who registered it. |
| XS_USER | Real Application Security (RAS) user | N | Corresponds to RAS users |
| XS_ROLE | Real Application Security (RAS) role | N | Corresponds to RAS roles |
| XS_ROLESET | Real Application Security (RAS) rolesets | N | Corresponds to RAS rolesets |
| XS_ROLE_GRANT | Real Application Security (RAS) role grants | N | Corresponds to RAS role grants |

**Table 128-12 (Cont.) DBMS_METADATA: Object Types**

| Type Name | Meaning | Attributes | Notes |
|---|---|---|---|
| XS_SECURITY_CLASS | Real Application Security (RAS) security class | SN | Corresponds to RAS security classes |
| XS_DATA_SECURITY | Real Application Security (RAS) data security policy | SN | Corresponds to RAS data security policies |
| XS_ACL | Real Application Security (RAS) ACL | SN | Corresponds to RAS access control lists (ACLs) and associated access control entries (ACEs) |
| XS_ACL_PARAM | Real Application Security (RAS) ACL parameter | N | Corresponds to RAS access control lists (ACL) parameters |
| XS_NAMESPACE | Real Application Security (RAS) namespace | N | Corresponds to RAS namespaces. |

Table Table 128-13 lists the types of objects returned for the major heterogeneous object types. For SCHEMA_EXPORT, certain object types are only returned if the INCLUDE_USER filter is specified at TRUE. In the table, such object types are marked INCLUDE_USER.

**Table 128-13 Object Types Returned for the Heterogeneous Object Type**

| Object Type | DATABASE_EXPORT | SCHEMA_EXPORT | TABLE_EXPORT | TRANSPORTABLE_EXPORT |
|---|---|---|---|---|
| ASSOCIATION | Yes | No | No | No |
| AUDIT | Yes | No | No | No |
| AUDIT_OBJ | Yes | Yes | Yes | Yes |
| CLUSTER | Yes | Yes | No | Yes |
| COMMENT | Yes | Yes | Yes | Yes |
| CONSTRAINT | Yes | Yes | Yes | Yes |
| CONTEXT | Yes | No | No | No |
| DB_LINK | Yes | Yes | No | No |
| DEFAULT_ROLE | Yes | INCLUDE_USER | No | No |
| DIMENSION | Yes | Yes | No | No |
| DIRECTORY | Yes | No | No | No |
| FGA_POLICY | Yes | No | No | Yes |
| FUNCTION | Yes | Yes | No | No |
| INDEX_STATISTICS | Yes | Yes | Yes | Yes |
| INDEX | Yes | Yes | Yes | Yes |
| INDEXTYPE | Yes | Yes | No | No |
| JAVA_SOURCE | Yes | Yes | No | No |
| JOB | Yes | Yes | No | No |
| LIBRARY | Yes | Yes | No | No |
| MATERIALIED_VIEW | Yes | Yes | No | No |
| MATERIALIZED_VIEW_LOG | Yes | Yes | No | No |
| OBJECT_GRANT | Yes | Yes | Yes | Yes |

**Table 128-13 (Cont.) Object Types Returned for the Heterogeneous Object Type**

| Object Type | DATABASE_EXPORT | SCHEMA_EXPORT | TABLE_EXPORT | TRANSPORTABLE_EXPORT |
|---|---|---|---|---|
| OPERATOR | Yes | Yes | No | No |
| PACKAGE | Yes | Yes | No | No |
| PACKAGE_SPEC | Yes | Yes | No | No |
| PACKAGE_BODY | Yes | Yes | No | No |
| PASSWORD_HISTORY | Yes | INCLUDE_USER | No | No |
| PASSWORD_VERIFY_FUNCTION | Yes | No | No | No |
| PROCEDURE | Yes | Yes | No | No |
| PROFILE | Yes | No | No | No |
| PROXY | Yes | No | No | No |
| REF_CONSTRAINT | Yes | Yes | Yes | Yes |
| REFRESH_GROUP | Yes | Yes | No | No |
| RESOURCE_COST | Yes | No | No | No |
| RLS_CONTEXT | Yes | No | No | Yes |
| RLS_GROUP | Yes | No | No | Yes |
| RLS_POLICY | Yes | Table data is retrieved according to policy | Table data is retrieved according to policy | Yes |
| ROLE | Yes | No | No | No |
| ROLE_GRANT | Yes | No | No | No |
| ROLLBACK_SEGMENT | Yes | No | No | No |
| SEQUENCE | Yes | Yes | No | No |
| SYNONYM | Yes | Yes | No | No |
| SYSTEM_GRANT | Yes | INCLUDE_USER | No | No |
| TABLE | Yes | Yes | Yes | Yes |
| TABLE_DATA | Yes | Yes | Yes | Yes |
| TABLE_STATISTICS | Yes | Yes | Yes | Yes |
| TABLESPACE | Yes | No | No | No |
| TABLESPACE_QUOTA | Yes | INCLUDE_USER | No | No |
| TRIGGER | Yes | Yes | Yes | Yes |
| TRUSTED_DB_LINK | Yes | No | No | No |
| TYPE | Yes | Yes | No | Yes, if the types are used by tables in the transportable set |
| TYPE_SPEC | Yes | Yes | No | Yes, if the types are used by tables in the transportable set |

**Table 128-13 (Cont.) Object Types Returned for the Heterogeneous Object Type**

| Object Type | DATABASE_EXPORT | SCHEMA_EXPORT | TABLE_EXPORT | TRANSPORTABLE_EXPORT |
|---|---|---|---|---|
| TYPE_BODY | Yes | Yes | No | Yes, if the types are used by tables in the transportable set |
| USER | Yes | INCLUDE_USER | No | No |
| VIEW | Yes | Yes | No | No |
| XMLSCHEMA | Yes | Yes | No | No |

### Return Values

An opaque handle to the class of objects. This handle is used as input to SET_FILTER, SET_COUNT, ADD_TRANSFORM, GET_QUERY, SET_PARSE_ITEM, FETCH_xxx, and CLOSE.

### Exceptions

- INVALID_ARGVAL. A NULL or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- INVALID_OBJECT_PARAM. The version or model parameter was not valid for the object_type.

## OPENW Function

This function specifies the type of object to be submitted and the object model. The return value is an opaque context handle.

> **See Also:**
>
> For more information about related subprograms:
>
> - Subprograms for Submitting XML to the Database

### Syntax

```
DBMS_METADATA.OPENW
  (object_type  IN VARCHAR2,
  version       IN VARCHAR2 DEFAULT 'COMPATIBLE',
  model         IN VARCHAR2 DEFAULT 'ORACLE')
 RETURN NUMBER;
```

### Parameters

**Table 128-14 OPENW Function Parameters**

| Parameter | Description |
|---|---|
| object_type | The type of object to be submitted. Valid types names and their meanings are listed in Table 128-12. The type cannot be a heterogeneous object type. |

**Table 128-14 (Cont.) OPENW Function Parameters**

| Parameter | Description |
|---|---|
| version | The version of DDL to be generated by the CONVERT function. DDL clauses that are incompatible with the version will not be generated. The legal values for this parameter are as follows:<br>• COMPATIBLE - This is the default. The version of the DDL corresponds to the database compatibility level. Database compatibility must be set to 9.2.0 or higher.<br>• LATEST - The version of the DDL corresponds to the database version.<br>• A specific database version. The value cannot be lower than 9.2.0. |
| model | Specifies which view to use. Only the Oracle proprietary (ORACLE) view is supported by DBMS_METADATA. |

### Return Values

An opaque handle to write context. This handle is used as input to the ADD_TRANSFORM, CONVERT, PUT, and CLOSE procedures.

### Exceptions

- INVALID_ARGVAL. A NULL or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- INVALID_OBJECT_PARAM. The model parameter was not valid for the object_type.

## PUT Function

This function submits an XML document containing object metadata to the database to create the object.

> **See Also:**
>
> For more information about related subprograms:
>
> - Subprograms for Submitting XML to the Database

### Syntax

```
DBMS_METADATA.PUT (
   handle     IN             NUMBER,
   document   IN             sys.XMLType,
   flags      IN             NUMBER,
   results    IN OUT NOCOPY  sys.ku$_SubmitResults)
  RETURN BOOLEAN;

DBMS_METADATA.PUT (
   handle     IN             NUMBER,
   document   IN             CLOB,
   flags      IN             NUMBER,
   results    IN OUT NOCOPY  sys.ku$_SubmitResults)
 RETURN BOOLEAN;
```

### Parameters

**Table 128-15 PUT Function Parameters**

| Parameter | Description |
|---|---|
| handle | The handle returned from OPENW. |
| document | The XML document containing object metadata for the type of the OPENW handle. |
| flags | Reserved for future use |
| results | Detailed results of the operation. |

### Return Values

TRUE if all SQL operations succeeded; FALSE if there were any errors.

### Usage Notes

The PUT function converts the XML document to DDL just as CONVERT does (applying the specified transforms in turn) and then submits each resultant DDL statement to the database. As with CONVERT, the DDL transform must be specified. The DDL statements and associated parse items are returned in the sys.ku$_SubmitResults nested table. With each DDL statement is a nested table of error lines containing any errors or exceptions raised by the statement.

The encoding of the XML document is embedded in its CLOB or XMLType representation. The version of the metadata is embedded in the XML. The generated DDL is valid for the database version specified in OPENW.

### Exceptions

- INVALID_ARGVAL. A NULL or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- INCONSISTENT_OPERATION. The DDL transform was not specified.
- INCOMPATIBLE_DOCUMENT. The version of the XML document is not compatible with this version of the software.

## SET_COUNT Procedure

This procedure specifies the maximum number of objects to be retrieved in a single FETCH_xxx call.

By default, each call to FETCH_xxx returns one object. You can use the SET_COUNT procedure to override this default. If FETCH_xxx is called from a client, specifying a count value greater than 1 can result in fewer server round trips and, therefore, improved performance.

For heterogeneous object types, a single FETCH_xxx operation only returns objects of a single object type.

> **See Also:**
>
> For more information about related subprograms:
>
> - Subprograms for Retrieving Multiple Objects From the Database

### Syntax

```
DBMS_METADATA.SET_COUNT (
   handle           IN NUMBER,
   value            IN NUMBER,
   object_type_path IN VARCHAR2 DEFAULT NULL);
```

### Parameters

**Table 128-16 SET_COUNT Procedure Parameters**

| Parameter | Description |
|---|---|
| `handle` | The handle returned from `OPEN`. |
| `value` | The maximum number of objects to retrieve. |
| `object_type_path` | A path name designating the object types to which the count value applies. By default, the count value applies to the object type of the `OPEN` handle. When the `OPEN` handle designates a heterogeneous object type, behavior can be either of the following:<br>• If `object_type_path` is omitted, then the count applies to all object types within the heterogeneous collection.<br>• If `object_type_path` is specified, then the count only applies to the specific node (or set of nodes) within the tree of object types forming the heterogeneous collection. |

### Exceptions

- `INVALID_ARGVAL`. A `NULL` or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- `INVALID_OPERATION`. `SET_COUNT` was called after the first call to `FETCH_xxx` for the `OPEN` context. After the first call to `FETCH_xxx` is made, no further calls to `SET_COUNT` for the current `OPEN` context are permitted.
- `INCONSISTENT_ARGS`. `object_type` parameter is not consistent with handle.

## SET_FILTER Procedure

This procedure specifies restrictions on the objects to be retrieved, for example, the object name or schema.

> **See Also:**
>
> For more information about related subprograms:
>
> - Subprograms for Retrieving Multiple Objects From the Database

### Syntax

```
DBMS_METADATA.SET_FILTER (
   handle           IN NUMBER,
   name             IN VARCHAR2,
   value            IN VARCHAR2,
   object_type_path IN VARCHAR2 DEFAULT NULL);

DBMS_METADATA.SET_FILTER (
   handle            IN NUMBER,
   name              IN VARCHAR2,
   value             IN BOOLEAN DEFAULT TRUE,
   object_type_path  IN VARCHAR2 DEFAULT NULL);

DBMS_METADATA.SET_FILTER (
   handle            IN NUMBER,
   name              IN VARCHAR2,
   value             IN NUMBER,
   object_type_path  IN VARCHAR2 DEFAULT NULL);
```

### Parameters

**Table 128-17 SET_FILTER Procedure Parameters**

| Parameter | Description |
|---|---|
| handle | The handle returned from OPEN. |
| name | The name of the filter. For each filter, Table 128-18 lists the object_type it applies to, its name, its datatype (text or Boolean), and its meaning or effect (including its default value, if any).<br>The Datatype column of Table 128-18 also indicates whether a text filter is an expression filter. An expression filter is the right-hand side of a SQL comparison (that is, a SQL comparison operator (=, !=, and so on.)) and the value compared against. The value must contain parentheses and quotation marks where appropriate. Note that in PL/SQL and SQL*Plus, two single quotes (*not* a double quote) are needed to represent an apostrophe. For example, an example of a NAME_EXPR filter in PL/SQL is as follows:<br>'IN (''DEPT'',''EMP'')'<br>The filter value is combined with a particular object attribute to produce a WHERE condition in the query that fetches the objects. In the preceding example, the filter is combined with the attribute corresponding to an object name; objects named 'DEPT' and 'EMP' are selected. |
| value | The value of the filter. Text, Boolean, and Numeric filters are supported. |
| object_type_path | A path name designating the object types to which the filter applies. By default, the filter applies to the object type of the OPEN handle. When the OPEN handle designates a heterogeneous object type, you can use this parameter to specify a filter for a specific node or set of nodes within the tree of object types that form the heterogeneous collection. See Table 128-19 for a listing of some of the values for this parameter. |

Table 128-18 describes the object type, name, datatype, and meaning of the filters available with the SET_FILTER procedure.

**Table 128-18 SET_FILTER: Filters**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| Named objects | NAME | Text | Objects with this exact name are selected. |
| Named objects | NAME_EXPR | Text expression | The filter value is combined with the object attribute corresponding to the object name to produce a WHERE condition in the query that fetches the objects.<br>By default, all named objects of object_type are selected. |
| Named objects | EXCLUDE_NAME_EXPR | Text expression | The filter value is combined with the attribute corresponding to the object name to specify objects that are to be excluded from the set of objects fetched.<br>By default, all named objects of the object type are selected. |
| TABLE_EXPORT | EXCLUDE_TABLES | Boolean | If TRUE, all paths associated with tables are excluded from the set of objects fetched. If FALSE (the default), all paths associated with tables are fetched. |
| Schema objects | SCHEMA | Text | Objects in this schema are selected. If the object type is SYNONYM, then specify PUBLIC to select public synonyms. |
| Schema objects | SCHEMA_EXPR | Text expression | The filter value is combined with the attribute corresponding to the object's schema.<br>The default is determined as follows:<br>- if BASE_OBJECT_SCHEMA is specified, then objects in that schema are selected;<br>- otherwise, objects in the current schema are selected. |
| PACKAGE, TYPE | SPECIFICATION | Boolean | If TRUE, retrieve the package or type specification. Defaults to TRUE. |
| PACKAGE, TYPE | BODY | Boolean | If TRUE, retrieve the package or type body. Defaults to TRUE. |
| TABLE, CLUSTER, INDEX, TABLE_DATA, TABLE_EXPORT, TRANSPORTABLE_EXPORT | TABLESPACE | Text | Objects in this tablespace (or having a partition in this tablespace) are selected. |
| TABLE, CLUSTER, INDEX,TABLE_DATA, TABLE_EXPORT, TRANSPORTABLE_EXPORT | TABLESPACE_EXPR | Text expression | The filter value is combined with the attribute corresponding to the object's tablespace (or in the case of a partitioned table or index, the partition's tablespaces). By default, objects in all tablespaces are selected. |
| TABLE, objects dependent on tables | PRIMARY | Boolean | If TRUE, retrieve primary tables (that is, tables for which the secondary object bit in obj$ is clear.<br>Defaults to TRUE. |

**Table 128-18 (Cont.) SET_FILTER: Filters**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| TABLE, objects dependent on tables | SECONDARY | Boolean | If TRUE, retrieve secondary tables (that is, tables for which the secondary object bit in obj$ is set).<br>Defaults to TRUE. |
| Dependent Objects | BASE_OBJECT_NAME | Text | Objects are selected that are defined or granted on objects with this name. Specify SCHEMA for triggers on schemas. Specify DATABASE for database triggers. Column-level comments cannot be selected by column name; the base object name must be the name of the table, view, or materialized view containing the column. |
| Dependent Objects | BASE_OBJECT_SCHEMA | Text | Objects are selected that are defined or granted on objects in this schema. If BASE_OBJECT_NAME is specified with a value other than SCHEMA or DATABASE, this defaults to the current schema. |
| Dependent Objects | BASE_OBJECT_NAME_EXPR | Text expression | The filter value is combined with the attribute corresponding to the name of the base object.<br>Not valid for schema and database triggers. |
| Dependent Objects | EXCLUDE_BASE_OBJECT_NAME_EXPR | Text expression | The filter value is combined with the attribute corresponding to the name of the base object to specify objects that are to be excluded from the set of objects fetched.<br>Not valid for schema and database triggers. |
| Dependent Objects | BASE_OBJECT_SCHEMA_EXPR | Text expression | The filter value is combined with the attribute corresponding to the schema of the base object. |
| Dependent Objects | BASE_OBJECT_TYPE | Text | The object type of the base object. |
| Dependent Objects | BASE_OBJECT_TYPE_EXPR | Text expression | The filter value is combined with the attribute corresponding to the object type of the base object.<br>By default no filtering is done on object type. |
| Dependent Objects | BASE_OBJECT_TABLESPACE | Text | The tablespace of the base object. |
| Dependent Objects | BASE_OBJECT_TABLESPACE_EXPR | Text expression | The filter value is combined with the attribute corresponding to the tablespaces of the base object. By default, no filtering is done on the tablespace. |
| INDEX, TRIGGER | SYSTEM_GENERATED | Boolean | If TRUE, select indexes or triggers even if they are system-generated. If FALSE, omit system-generated indexes or triggers. Defaults to TRUE. |
| Granted Objects | GRANTEE | Text | Objects are selected that are granted to this user or role. Specify PUBLIC for grants to PUBLIC. |
| Granted Objects | PRIVNAME | Text | The name of the privilege or role to be granted. For TABLESPACE_QUOTA, only UNLIMITED can be specified. |
| Granted Objects | GRANTEE_EXPR | Text expression | The filter value is combined with the attribute corresponding to the grantee name. |

**Table 128-18 (Cont.) SET_FILTER: Filters**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| Granted Objects | EXCLUDE_GRANTEE_EXPR | Text expression | The filter value is combined with the attribute corresponding to the grantee name to specify objects that are to be excluded from the set of objects fetched. |
| OBJECT_GRANT | GRANTOR | Text | Object grants are selected that are granted by this user. |
| SYNONYM, JAVA_SOURCE, XMLSCHEMA | LONGNAME | Text | A name longer than 30 characters. Objects with this exact name are selected. If the object name is 30 characters or less, the NAME filter must be used. |
| SYNONYM, JAVA_SOURCE, XMLSCHEMA | LONGNAME_EXPR | Text | The filter value is combined with the attribute corresponding to the object's long name. By default, no filtering is done on the long name of an object. |
| All objects | CUSTOM_FILTER | Text | The text of a WHERE condition. The condition is appended to the query that fetches the objects. By default, no custom filter is used.<br>The other filters are intended to meet the needs of the majority of users. Use CUSTOM_FILTER when no defined filters exists for your purpose. Of necessity such a filter depends on the detailed structure of the UDTs and views used in the query. Because filters may change from version to version, upward compatibility is not guaranteed. |
| All objects | EDITION | Text | The edition filter is accepted for any object type, but affects only objects that support editions. The filter is only accepted for local objects (that is, the network_link parameter is not specified in the OPEN call). The edition name must be a valid edition name. If an edition is not specified, the edition of the active session is used. |
| SCHEMA_EXPORT | SCHEMA | Text | The schema whose objects are selected. |
| SCHEMA_EXPORT | SCHEMA_EXPR | Text expression | The filter value is either:<br>combined with the attribute corresponding to a schema name to produce a WHERE condition in the query that fetches schema objects,<br>combined with the attribute corresponding to a base schema name to produce a WHERE condition in the query that fetches dependent objects.<br>By default the current user's objects are selected. |
| SCHEMA_EXPORT | INCLUDE_USER | Boolean | If TRUE, retrieve objects containing privileged information about the user. For example, USER, PASSWORD_HISTORY, TABLESPACE_QUOTA.<br>Defaults to FALSE. |
| TABLE_EXPORT | SCHEMA | Text | Objects (tables and their dependent objects) in this schema are selected. |

**Table 128-18 (Cont.) SET_FILTER: Filters**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| TABLE_EXPORT | SCHEMA_EXPR | Text expression | The filter value is either:<br>combined with the attribute corresponding to a schema name to produce a WHERE condition in the query that fetches the tables,<br>combined with the attribute corresponding to a base schema name to produce a WHERE condition in the query that fetches the tables' dependent objects.<br>By default the current user's objects are selected. |
| TABLE_EXPORT | NAME | Text | The table with this exact name is selected along with its dependent objects. |
| TABLE_EXPORT | NAME_EXPR | Text expression | The filter value is combined with the attribute corresponding to a table name in the queries that fetch tables and their dependent objects.<br>By default all tables in the selected schemas are selected, along with their dependent objects. |
| Heterogeneous objects | BEGIN_WITH | Text | The fully qualified path name of the first object type in the heterogeneous collection to be retrieved. Objects normally fetched prior to this object type will not be retrieved. |
| Heterogeneous objects | BEGIN_AFTER | Text | The fully qualified path name of an object type after which the heterogeneous retrieval should begin. Objects of this type will not be retrieved, nor will objects normally fetched prior to this object type. |
| Heterogeneous objects | END_BEFORE | Text | The fully qualified path name of an object type where the heterogeneous retrieval should end. Objects of this type will not be retrieved, nor will objects normally fetched after this object type. |
| Heterogeneous objects | END_WITH | Text | The fully qualified path name of the last object type in the heterogeneous collection to be retrieved. Objects normally fetched after this object type will not be retrieved. |

**Table 128-18 (Cont.) SET_FILTER: Filters**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| Heterogeneous objects | INCLUDE_PATH_EXPR, EXCLUDE_PATH_EXPR | Text expression | For these two filters, the filter value is combined with the attribute corresponding to an object type path name to produce a WHERE condition in the query that fetches the object types belonging to the heterogeneous collection. Objects of types satisfying this condition are included (INCLUDE_PATH_EXPR) or excluded (EXCLUDE_PATH_EXPR) from the set of object types fetched. Path names in the filter value do not have to be fully qualified. See Table 128-19 for valid path names that can be used with these filters.<br>BEGIN_WITH, BEGIN_AFTER, END_BEFORE, END_WITH, INCLUDE_PATH_EXPR, and EXCLUDE_PATH_EXPR all restrict the set of object types in the heterogeneous collection. By default, objects of all object types in the heterogeneous collection are retrieved. |

### Usage Notes

- Each call to SET_FILTER causes a WHERE condition to be added to the underlying query that fetches the set of objects. The WHERE conditions are concatenated with the AND keyword so that you can use multiple SET_FILTER calls to refine the set of objects to be returned. For example to specify that you want the object named EMP in schema SCOTT, do the following:

```
SET_FILTER(handle,'SCHEMA','SCOTT');
SET_FILTER(handle,'NAME','EMP');
```

- You can use the same text expression filter multiple times with different values. All the filter conditions will be applied to the query. For example, to get objects with names between Felix and Oscar, do the following:

```
SET_FILTER(handle,'NAME_EXPR','>=''FELIX''');
SET_FILTER(handle,'NAME_EXPR','<=''OSCAR''');
```

- With SET_FILTER, you can specify the schema of objects to be retrieved, but security considerations may override this specification. If the caller is SYS or has the SELECT_CATALOG_ROLE role, then any object can be retrieved; otherwise, only the following can be retrieved:
  - Schema objects owned by the current user
  - Public synonyms
  - System privileges granted to the current user or to PUBLIC
  - Grants on objects for which the current user is owner, grantor, or grantee (either explicitly or as PUBLIC).
  - SCHEMA_EXPORT where the name is the current user
  - TABLE_EXPORT where SCHEMA is the current user

  If you request objects that you are not privileged to retrieve, no exception is raised; the object is not retrieved, as if it did not exist.

In stored procedures, functions, and definers-rights packages, roles (such as SELECT_CATALOG_ROLE) are disabled. Therefore, such a PL/SQL program can only fetch metadata for objects in its own schema. If you want to write a PL/SQL program that fetches metadata for objects in a different schema (based on the invoker's possession of SELECT_CATALOG_ROLE), you must make the program invokers-rights.

- For heterogeneous object types, the BEGIN_WITH and BEGIN_AFTER filters allow restart on an object type boundary. Appropriate filter values are returned by the FETCH_XML_CLOB procedure.

  Filters on heterogeneous objects provide default values for filters on object types within the collection. You can override this default for a particular object type by specifying the appropriate filter for the specific object type path. For example, for SCHEMA_EXPORT the NAME filter specifies the schema to be fetched including all the tables in the schema, but you can further restrict this set of tables by supplying a NAME_EXPR filter explicitly for the TABLE object type path. Table 128-19 lists valid object type path names for the major heterogeneous object types along with an explanation of the scope of each path name. (The same information is available in the following catalog views: DATABASE_EXPORT_OBJECTS, SCHEMA_EXPORT_OBJECTS, and TABLE_EXPORT_OBJECTS.) See Table 128-18 for filters defined for each path name. These path names are valid in the INCLUDE_PATH_EXPR and EXCLUDE_PATH_EXPR filters. Path names marked with an asterisk (*) are *only* valid in those filters; they cannot be used as values of the SET_FILTER object_type_path parameter.

**Table 128-19 Object Type Path Names for Heterogeneous Object Types**

| Heterogeneous Type | Path Name (*=valid only in xxx_PATH_EXPR) | Scope |
|---|---|---|
| TABLE_EXPORT | AUDIT_OBJ | Object audits on the selected tables |
| TABLE_EXPORT | COMMENT | Table and column comments for the selected tables |
| TABLE_EXPORT | CONSTRAINT | Constraints (including referential constraints) on the selected tables |
| TABLE_EXPORT | *GRANT | Object grants on the selected tables |
| TABLE_EXPORT | INDEX | Indexes (including domain indexes) on the selected tables |
| TABLE_EXPORT | OBJECT_GRANT | Object grants on the selected tables |
| TABLE_EXPORT | REF_CONSTRAINT | Referential (foreign key) constraints on the selected tables |
| TABLE_EXPORT | STATISTICS | Statistics on the selected tables |
| TABLE_EXPORT | TABLE_DATA | Row data for the selected tables |
| TABLE_EXPORT | TRIGGER | Triggers on the selected tables |
| SCHEMA_EXPORT | ASSOCIATION | Statistics type associations for objects in the selected schemas |
| SCHEMA_EXPORT | AUDIT_OBJ | Audits on all objects in the selected schemas |
| SCHEMA_EXPORT | CLUSTER | Clusters in the selected schemas and their indexes |
| SCHEMA_EXPORT | COMMENT | Comments on all objects in the selected schemas |
| SCHEMA_EXPORT | CONSTRAINT | Constraints (including referential constraints) on all objects in the selected schemas |
| SCHEMA_EXPORT | DB_LINK | Private database links in the selected schemas |
| SCHEMA_EXPORT | DEFAULT_ROLE | Default roles granted to users associated with the selected schemas |
| SCHEMA_EXPORT | DIMENSION | Dimensions in the selected schemas |

**Table 128-19 (Cont.) Object Type Path Names for Heterogeneous Object Types**

| Heterogeneous Type | Path Name (*=valid only in xxx_PATH_EXPR) | Scope |
|---|---|---|
| SCHEMA_EXPORT | FUNCTION | Functions in the selected schemas and their dependent grants and audits |
| SCHEMA_EXPORT | *GRANT | Grants on objects in the selected schemas |
| SCHEMA_EXPORT | INDEX | Indexes (including domain indexes) on tables and clusters in the selected schemas |
| SCHEMA_EXPORT | INDEXTYPE | Indextypes in the selected schemas and their dependent grants and audits |
| SCHEMA_EXPORT | JAVA_SOURCE | Java sources in the selected schemas and their dependent grants and audits |
| SCHEMA_EXPORT | JOB | Jobs in the selected schemas |
| SCHEMA_EXPORT | LIBRARY | External procedure libraries in the selected schemas |
| SCHEMA_EXPORT | MATERIALIZED_VIEW | Materialized views in the selected schemas |
| SCHEMA_EXPORT | MATERIALIZED_VIEW_LOG | Materialized view logs on tables in the selected schemas |
| SCHEMA_EXPORT | OBJECT_GRANT | Grants on objects in the selected schemas |
| SCHEMA_EXPORT | OPERATOR | Operators in the selected schemas and their dependent grants and audits |
| SCHEMA_EXPORT | PACKAGE | Packages (both specification and body) in the selected schemas, and their dependent grants and audits |
| SCHEMA_EXPORT | PACKAGE_BODY | Package bodies in the selected schemas |
| SCHEMA_EXPORT | PACKAGE_SPEC | Package specifications in the selected schemas |
| SCHEMA_EXPORT | PASSWORD_HISTORY | The password history for users associated with the selected schemas |
| SCHEMA_EXPORT | PROCEDURE | Procedures in the selected schemas and their dependent grants and audits |
| SCHEMA_EXPORT | REF_CONSTRAINT | Referential (foreign key) constraints on tables in the selected schemas |
| SCHEMA_EXPORT | REFRESH_GROUP | Refresh groups in the selected schemas |
| SCHEMA_EXPORT | SEQUENCE | Sequences in the selected schemas and their dependent grants and audits |
| SCHEMA_EXPORT | STATISTICS | Statistics on tables and indexes in the selected schemas |
| SCHEMA_EXPORT | SYNONYM | Private synonyms in the selected schemas |
| SCHEMA_EXPORT | TABLE | Tables in the selected schemas and their dependent objects (indexes, constraints, triggers, grants, audits, comments, table data, and so on) |
| SCHEMA_EXPORT | TABLE_DATA | Row data for tables in the selected schemas |
| SCHEMA_EXPORT | TABLESPACE_QUOTA | Tablespace quota granted to users associated with the selected schemas |
| SCHEMA_EXPORT | TRIGGER | Triggers on tables in the selected schemas |
| SCHEMA_EXPORT | XS_SECURITY_CLASS | Oracle Real Application Security (RAS) security classes |
| SCHEMA_EXPORT | XS_DATA_SECURITY | Oracle Real Application Security (RAS) data security policies |

**Table 128-19 (Cont.) Object Type Path Names for Heterogeneous Object Types**

| Heterogeneous Type | Path Name (*=valid only in xxx_PATH_EXPR) | Scope |
|---|---|---|
| SCHEMA_EXPORT | XS_ACL | Oracle Real Application Security (RAS) access control lists (ACLs) |
| SCHEMA_EXPORT | TYPE | Types (both specification and body) in the selected schemas, and their dependent grants and audits |
| SCHEMA_EXPORT | TYPE_BODY | Type bodies in the selected schemas |
| SCHEMA_EXPORT | TYPE_SPEC | Type specifications in the selected schemas |
| SCHEMA_EXPORT | USER | User definitions for users associated with the selected schemas |
| SCHEMA_EXPORT | VIEW | Views in the selected schemas and their dependent objects (grants, constraints, comments, audits) |
| DATABASE_EXPORT | ASSOCIATION | Statistics type associations for objects in the database |
| DATABASE_EXPORT | AUDIT | Audits of SQL statements |
| DATABASE_EXPORT | AUDIT_OBJ | Audits on all objects in the database |
| DATABASE_EXPORT | CLUSTER | Clusters and their indexes |
| DATABASE_EXPORT | COMMENT | Comments on all objects |
| DATABASE_EXPORT | CONSTRAINT | Constraints (including referential constraints) |
| DATABASE_EXPORT | CONTEXT | Application contexts |
| DATABASE_EXPORT | DB_LINK | Private and public database links |
| DATABASE_EXPORT | DEFAULT_ROLE | Default roles granted to users in the database |
| DATABASE_EXPORT | DIMENSION | Dimensions in the database |
| DATABASE_EXPORT | DIRECTORY | Directory objects in the database |
| DATABASE_EXPORT | FGA_POLICY | Fine-grained audit policies |
| DATABASE_EXPORT | FUNCTION | Functions |
| DATABASE_EXPORT | * GRANT | Object and system grants |
| DATABASE_EXPORT | INDEX | Indexes (including domain indexes) on tables and clusters |
| DATABASE_EXPORT | INDEXTYPE | Indextypes and their dependent grants and audits |
| DATABASE_EXPORT | JAVA_SOURCE | Java sources and their dependent grants and audits |
| DATABASE_EXPORT | JOB | Jobs |
| DATABASE_EXPORT | LIBRARY | External procedure libraries |
| DATABASE_EXPORT | MATERIALIZED_VIEW | Materialized views |
| DATABASE_EXPORT | MATERIALIZED_VIEW_LOG | Materialized view logs |
| DATABASE_EXPORT | OBJECT_GRANT | All object grants in the database |
| DATABASE_EXPORT | OPERATOR | Operators and their dependent grants and audits |
| DATABASE_EXPORT | PACKAGE | Packages (both specification and body) and their dependent grants and audits |
| DATABASE_EXPORT | PACKAGE_BODY | Package bodies |
| DATABASE_EXPORT | PACKAGE_SPEC | Package specifications |
| DATABASE_EXPORT | PASSWORD_HISTORY | Password histories for database users |

**Table 128-19 (Cont.) Object Type Path Names for Heterogeneous Object Types**

| Heterogeneous Type | Path Name (*=valid only in xxx_PATH_EXPR) | Scope |
|---|---|---|
| DATABASE_EXPORT | *PASSWORD_VERIFY_FUNCTION | The password complexity verification function |
| DATABASE_EXPORT | PROCEDURE | Procedures and their dependent grants and objects |
| DATABASE_EXPORT | PROFILE | Profiles |
| DATABASE_EXPORT | PROXY | Proxy authentications |
| DATABASE_EXPORT | REF_CONSTRAINT | Referential (foreign key) constraints on tables in the database |
| DATABASE_EXPORT | REFRESH_GROUP | Refresh groups |
| DATABASE_EXPORT | *RESOURCE_ COST | Resource cost information |
| DATABASE_EXPORT | RLS_CONTEXT | Fine-grained access-control driving contexts |
| DATABASE_EXPORT | RLS_GROUP | Fine-grained access-control policy groups |
| DATABASE_EXPORT | RLS_POLICY | Fine-grained access-control policies |
| DATABASE_EXPORT | ROLE | Roles |
| DATABASE_EXPORT | ROLE_GRANT | Role grants to users in the database |
| DATABASE_EXPORT | ROLLBACK_SEGMENT | Rollback segments |
| DATABASE_EXPORT | *SCHEMA (named object) | Database schemas including for each schema all related and dependent objects: user definitions and their attributes (default roles, role grants, tablespace quotas, and so on), objects in the schema (tables, view, packages, types, and so on), and their dependent objects (grants, audits, indexes, constraints, and so on). The NAME and NAME_EXPR filters can be used with this object type path name to designate the database schemas to be fetched. |
| DATABASE_EXPORT | SEQUENCE | Sequences |
| DATABASE_EXPORT | STATISTICS | Statistics on tables and indexes |
| DATABASE_EXPORT | SYNONYM | Public and private synonyms |
| DATABASE_EXPORT | SYSTEM_GRANT | System privilege grants |
| DATABASE_EXPORT | TABLE | Tables and their dependent objects (indexes, constraints, triggers, grants, audits, comments, table data, and so on) |
| DATABASE_EXPORT | TABLE_DATA | Row data for all tables |
| DATABASE_EXPORT | TABLESPACE | Tablespace definitions |
| DATABASE_EXPORT | TABLESPACE_QUOTA | Tablespace quota granted to users in the database |
| DATABASE_EXPORT | TRIGGER | Triggers on the database, on schemas, and on schema objects |
| DATABASE_EXPORT | XS_USER | Oracle Real Application Security (RAS) users |
| DATABASE_EXPORT | XS_ROLE | Oracle Real Application Security (RAS) roles |
| DATABASE_EXPORT | XS_SECURITY_CLASS | Oracle Real Application Security (RAS) security classes |
| DATABASE_EXPORT | XS_DATA_SECURITY | Oracle Real Application Security (RAS) data security policies |
| DATABASE_EXPORT | XS_ACL | Oracle Real Application Security (RAS) access control lists (ACLs) |
| DATABASE_EXPORT | XS_NAMESPACE | Oracle Real Application Security (RAS) namespaces |
| DATABASE_EXPORT | TRUSTED_DB_LINK | Trusted links |

**Table 128-19 (Cont.) Object Type Path Names for Heterogeneous Object Types**

| Heterogeneous Type | Path Name (*=valid only in xxx_PATH_EXPR) | Scope |
|---|---|---|
| DATABASE_EXPORT | TYPE | Types (both specification and body) and their dependent grants and audits |
| DATABASE_EXPORT | TYPE_BODY | Type bodies |
| DATABASE_EXPORT | TYPE_SPEC | Type specifications |
| DATABASE_EXPORT | USER | User definitions |
| DATABASE_EXPORT | VIEW | Views |

### Exceptions

- INVALID_ARGVAL. A NULL or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- INVALID_OPERATION. SET_FILTER was called after the first call to FETCH_xxx for the OPEN context. After the first call to FETCH_xxx is made, no further calls to SET_FILTER are permitted.
- INCONSISTENT_ARGS. The arguments are inconsistent. Possible inconsistencies include the following:
  - The filter name is not valid for the object type associated with the OPEN context.
  - The filter name is not valid for the object_type_path.
  - The object_type_path is not part of the collection designated by handle .
  - The filter value is the wrong datatype.

## SET_PARSE_ITEM Procedure

This procedure is used for both retrieval and submission. This procedure enables output parsing and specifies an object attribute to be parsed and returned.

> **See Also:**
>
> For more information about related subprograms:
>
> - Subprograms for Retrieving Multiple Objects From the Database
> - Subprograms for Submitting XML to the Database

### Syntax

The following syntax applies when SET_PARSE_ITEM is used for object retrieval:

```
DBMS_METADATA.SET_PARSE_ITEM (
   handle       IN  NUMBER,
   name         IN  VARCHAR2,
   object_type  IN  VARCHAR2 DEFAULT NULL);
```

The following syntax applies when SET_PARSE_ITEM is used for XML submission:

```
DBMS_METADATA.SET_PARSE_ITEM (
   handle     IN NUMBER,
   name        IN VARCHAR2);
```

### Parameters

**Table 128-20 SET_PARSE_ITEM Procedure Parameters**

| Parameter | Description |
|---|---|
| handle | The handle returned from OPEN (or OPENW). |
| name | The name of the object attribute to be parsed and returned. See Table 128-21 for the attribute object type, name, and meaning. |
| object_type | Designates the object type to which the parse item applies (this is an object type name, not a path name). By default, the parse item applies to the object type of the OPEN handle. When the OPEN handle designates a heterogeneous object type, behavior can be either of the following:<br>• If object_type is omitted, then the parse item applies to all object types within the heterogeneous collection<br>.<br>• If object_type is specified, then the parse item only applies to that specific object type within the collection<br>.<br>This parameter only applies when SET_PARSE_ITEM is used for object retrieval. |

Table 128-21 describes the object type, name, and meaning of the items available in the SET_PARSE_ITEM procedure.

Because new items are occasionally added, you can query the DBMS_METADATA_PARSE_ITEMS view to see a complete list of valid parse items or to find valid parse items for a specific object type.

**Table 128-21 SET_PARSE_ITEM: Parse Items**

| Object Type | Name | Meaning |
|---|---|---|
| All objects | VERB | If FETCH_XML_CLOB is called, no value is returned.<br>If FETCH_DDL is called, then for every row in the sys.ku$_ddls nested table returned by FETCH_DDL the verb in the corresponding ddlText is returned. If the ddlText is a SQL DDL statement, then the SQL verb (for example, CREATE, GRANT, AUDIT) is returned. If the ddlText is a procedure call (for example, DBMS_AQADM.CREATE_QUEUE_TABLE()) then the package.procedure-name is returned. |
| All objects | OBJECT_TYPE | If FETCH_XML_CLOB is called, an object type name from Table 128-12 is returned.<br>If FETCH_DDL is called and the ddlText is a SQL DDL statement whose verb is CREATE or ALTER, the object type as used in the DDL statement is returned (for example, TABLE, PACKAGE_BODY, and so on). Otherwise, an object type name from Table 128-12 is returned. |

**Table 128-21 (Cont.) SET_PARSE_ITEM: Parse Items**

| Object Type | Name | Meaning |
|---|---|---|
| Schema objects | SCHEMA | The object schema is returned. If the object is not a schema object, no value is returned. |
| Named objects | NAME | The object name is returned. If the object is not a named object, no value is returned. |
| TABLE, TABLE_DATA, INDEX | TABLESPACE | The name of the object's tablespace or, if the object is a partitioned table, the default tablespace is returned. For a TABLE_DATA object, this is always the tablespace where the rows are stored. |
| TRIGGER | ENABLE | If the trigger is enabled, ENABLE is returned. If the trigger is disabled, DISABLE is returned. |
| OBJECT_GRANT, TABLESPACE_QUOTA | GRANTOR | The grantor is returned. |
| Dependent objects (including domain index secondary tables) | BASE_OBJECT_NAME | The name of the base object is returned. If the object is not a dependent object, no value is returned. |
| Dependent objects (including domain index secondary tables) | BASE_OBJECT_SCHEMA | The schema of the base object is returned. If the object is not a dependent object, no value is returned. |
| Dependent objects (including domain index secondary tables) | BASE_OBJECT_TYPE | The object type of the base object is returned. If the object is not a dependent object, no value is returned. |
| Granted objects | GRANTEE | The grantee is returned. If the object is not a granted object, no value is returned. |

### Usage Notes

These notes apply when using SET_PARSE_ITEM to retrieve objects.

By default, the FETCH_xxx routines return an object's metadata as XML or creation DDL. By calling SET_PARSE_ITEM you can request that individual attributes of the object be returned as well.

You can call SET_PARSE_ITEM multiple times to ask for multiple items to be parsed and returned. Parsed items are returned in the sys.ku$_parsed_items nested table.

For TABLE_DATA objects, the following parse item return values are of interest:

| If Object Is | NAME, SCHEMA | BASE_OBJECT_NAME, BASE_OBJECT_SCHEMA |
|---|---|---|
| nonpartitioned table | table name, schema | table name, schema |
| table partition | partition name, schema | table name, schema |
| nested table | storage table name, schema | name and schema of top-level table (*not* the parent nested table) |

Tables are not usually thought of as dependent objects. However, secondary tables for domain indexes are dependent on the domain indexes. Consequently, the BASE_OBJECT_NAME,

BASE_OBJECT_SCHEMA and BASE_OBJECT_TYPE parse items for secondary TABLE objects return the name, schema, and type of the domain index.

> **See Also:**
>
> - "FETCH_xxx Functions and Procedures"
> - *Oracle Database Utilities* for more information about using the metadata APIs.

By default, the CONVERT and PUT procedures simply transform an object's XML metadata to DDL. By calling SET_PARSE_ITEM you can request that individual attributes of the object be returned as well.

### Exceptions

- INVALID_ARGVAL. A NULL or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- INVALID_OPERATION. SET_PARSE_ITEM was called after the first call to FETCH_xxx for the OPEN context. After the first call to FETCH_xxx is made, no further calls to SET_PARSE_ITEM are permitted.
- INCONSISTENT_ARGS. The attribute name is not valid for the object type associated with the OPEN context.

## SET_TRANSFORM_PARAM and SET_REMAP_PARAM Procedures

These procedures are used for both retrieval and submission. SET_TRANSFORM_PARAM and SET_REMAP_PARAM specify parameters to the XSLT stylesheet identified by transform_handle.

### Purpose

Use these parameters to modify or customize the output of the transform.

> **See Also:**
>
> For more information about related subprograms:
>
> - Subprograms for Retrieving Multiple Objects From the Database
> - Subprograms for Submitting XML to the Database

### Syntax

```
DBMS_METADATA.SET_TRANSFORM_PARAM (
   transform_handle   IN NUMBER,
   name               IN VARCHAR2,
   value              IN VARCHAR2,
   object_type        IN VARCHAR2 DEFAULT NULL);

DBMS_METADATA.SET_TRANSFORM_PARAM (
   transform_handle   IN NUMBER,
   name               IN VARCHAR2,
   value              IN BOOLEAN DEFAULT TRUE,
```

```
   object_type         IN VARCHAR2 DEFAULT NULL);

DBMS_METADATA.SET_TRANSFORM_PARAM (
   transform_handle    IN NUMBER,
   name                IN VARCHAR2,
   value               IN NUMBER,
   object_type         IN VARCHAR2 DEFAULT NULL);

DBMS_METADATA.SET_REMAP_PARAM (
   transform_handle    IN NUMBER,
   name                IN VARCHAR2,
   old_value           IN VARCHAR2,
   new_value           IN VARCHAR2,
   object_type         IN VARCHAR2 DEFAULT NULL);
```

### Parameters

Table 128-22 describes the parameters for the SET_TRANSFORM_PARAM and SET_REMAP_PARAM procedures.

Because new parameters are occasionally added, Oracle suggests that you query the DBMS_METADATA_TRANSFORM_PARAMS view to see all the valid transform parameters for each transform or to find valid transform parameters for specific object types.

**Table 128-22 SET_TRANSFORM_PARAM and SET_REMAP_PARAM Parameters**

| Parameters | Description |
|---|---|
| name | The name of the transform parameter.<br>For descriptions of the parameters available for each transform on the SET_TRANSFORM_PARAM procedure, see the following:<br>Table 128-23 - DDL transform<br>Table 128-24 - MODIFY transform<br>Table 128-26 - SXML transform<br>Table 128-27 - MODIFYSXML transform<br>Table 128-28 - SXMLDDL transform<br>For descriptions of the parameters available for the MODIFY transform on the SET_REMAP_PARAM procedure, see Table 128-25.<br>For descriptions of the parameters available for the ALTERXML transform, see Table 128-4. |
| new_value | The new value for the remapping. This parameter is valid only for SET_REMAP_PARAM. |
| object_type | Designates the object type to which the transform or remap parameter applies. By default, it applies to the same object type as the transform. In cases where the transform applies to all object types within a heterogeneous collection, the following apply:<br>• If object_type is omitted, the parameter applies to all applicable object types within the heterogeneous collection.<br>• If object_type is specified, the parameter only applies to that object type.<br>This allows a caller who has added a transform to a heterogeneous collection to specify different transform parameters for different object types within the collection. |
| old_value | The old value for the remapping. This parameter is valid only for SET_REMAP_PARAM. |

**Table 128-22 (Cont.) SET_TRANSFORM_PARAM and SET_REMAP_PARAM Parameters**

| Parameters | Description |
|---|---|
| transform_handle | Either (1) the handle returned from ADD_TRANSFORM, or (2) the enumerated constant SESSION_TRANSFORM that designates the DDL transform for the whole session.<br>Note that the handle returned by OPEN is not a valid transform handle.<br>For SET_REMAP_PARAM, the transform handle must designate the MODIFY transform. |
| value | The value of the transform. This parameter is valid only for SET_TRANSFORM_PARAM. |

Table 128-23 describes the object type, name, datatype, and meaning of the parameters for the DDL transform in the SET_TRANSFORM_PARAM procedure.

**Table 128-23 SET_TRANSFORM_PARAM: Transform Parameters for the DDL Transform**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| USER, TABLE, CLUSTER, VIEW, MATERIALIZED_VIEW PROCEDURE, FUNCTION, PACKAGE, TYPE, TRIGGER | COLLATION_CLAUSE | Text | There are three possible values:<br>• NEVER — Collation clauses are never generated.<br>• ALWAYS — Collation clauses are always generated.<br>• NON_DEFAULT — Collation clauses other than USING_NLS_COMP are generated. |
| TABLE | OMIT_ENCRYPTION_CLAUSE | BOOLEAN | If set to Y, directs Data Pump to suppress column encryption clauses. Columns encrypted in the source database are not encrypted in imported tables. Defaults to N.<br>If set to N, directs Data Pump to create column encryption clauses, as in the source database. |
| TABLE | DWCS_CVT_IOTS | BOOLEAN | If set to Y, directs Data Pump to transform Index Organized tables to heap organized tables by suppressing the ORGANIZATION INDEX clause when creating the table. Defaults to N.<br>If set to N, the generated DDL retains the table characteristics of the source object. |
| TABLE, CONSTRAINT | DWCS_CVT_CONSTRAINTS | BOOLEAN | If set to Y, directs Data Pump to create pk/fk/uk constraints as disabled. Defaults to N.<br>If set to N, directs Data Pump to create pk/fk/uk constraints based on the source database status. |

**Table 128-23 (Cont.) SET_TRANSFORM_PARAM: Transform Parameters for the DDL Transform**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| TABLE, CONSTRAINT | CONSTRAINT_USE_DEFAULT_INDEX | BOOLEAN | This transform parameter affects the generation of index relating the pk/uk constraint. If set to Y, forces the name of an index automatically created to enforce the constraint to be identical to the constraint name. Defaults to N.<br>If set to N, the index is created as it was named on the source database. |
| TABLE, CONSTRAINT | CONSTRAINT_NAME_FROM_INDEX | BOOLEAN | This transform parameter affects the generation of pk/uk constraints which reference user created indexes. If set to Y, forces the name of the constraint to match the name of the index. Defaults to N.<br>If set to N, the constraint is created as it was named on the source database. |
| TABLE | INCLUDE_SHARDING_CLAUSES | BOOLEAN | This transform parameter enables sharding keywords for the transform to facilitate creating sharded tables, sequences, tablespaces and tablespace set. Options: [TRUE \| FALSE]<br>Default: FALSE.<br>When set to TRUE, sharding syntax keywords are available:<br>SHARDED keyword<br>DUPLICATED keyword<br>PARTITION BY keyword, with the keyword options CONSISTENT HASH and PARTITIONS AUTO<br>When the INCLUDE_SHARDING_CLAUSES parameter is set to FALSE in CREATE TABLE, the DDL will contain PARTITION BY RANGE and not include the PARTITIONS AUTO clause. |
| All objects | PRETTY | BOOLEAN | If TRUE, format the output with indentation and line feeds. Defaults to TRUE. |
| All objects | SQLTERMINATOR | BOOLEAN | If TRUE, append a SQL terminator (; or /) to each DDL statement. Defaults to FALSE. |
| TABLE | CONSTRAINTS | BOOLEAN | If TRUE, include all non-referential table constraints in the DDL. If FALSE, omit them. Defaults to TRUE. |
| TABLE | REF_CONSTRAINTS | BOOLEAN | If TRUE, include all referential constraints (foreign keys) in the DDL. If FALSE, omit them. Defaults to TRUE. |

**Table 128-23 (Cont.) SET_TRANSFORM_PARAM: Transform Parameters for the DDL Transform**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| TABLE | CONSTRAINTS_AS_ALTER | BOOLEAN | If TRUE, include table constraints as separate ALTER TABLE (and, if necessary, CREATE INDEX) statements. If FALSE, specify table constraints as part of the CREATE TABLE statement. Defaults to FALSE. Requires that CONSTRAINTS be TRUE.<br>**Note**: The CONSTRAINTS_AS_ALTER parameter has no effect when an index and a constraint are created for a table in two separate DDL statements. |
| TABLE, TYPE | OID | BOOLEAN | If TRUE, include the Object ID (OID) clause in the DDL. If FALSE, omit it. Defaults to FALSE. |
| TABLE | SIZE_BYTE_KEYWORD | BOOLEAN | If TRUE, include the BYTE keyword as part of the size specification of CHAR and VARCHAR2 columns that use byte semantics. If FALSE, omit the keyword. Defaults to FALSE. |
| TABLE, INDEX | PARTITIONING | BOOLEAN | If TRUE, include partitioning clauses in the DDL. If FALSE, omit them. Defaults to TRUE. |
| INDEX, CONSTRAINT, ROLLBACK_SEGMENT, CLUSTER, TABLE, TABLESPACE | SEGMENT_ATTRIBUTES | BOOLEAN | If TRUE, include segment attributes clauses (physical attributes, storage attributes, tablespace, logging) in the DDL. If FALSE, omit them. Defaults to TRUE. |
| INDEX, CONSTRAINT, ROLLBACK_SEGMENT, CLUSTER,TABLE | STORAGE | BOOLEAN | If TRUE, include storage clauses in the DDL. If FALSE, omit them. (Ignored if SEGMENT_ATTRIBUTES is FALSE.) Defaults to TRUE. |
| INDEX, CONSTRAINT, ROLLBACK_SEGMENT, CLUSTER,TABLE | TABLESPACE | BOOLEAN | If TRUE, include tablespace clauses in the DDL. If FALSE, omit them. (Ignored if SEGMENT_ATTRIBUTES is FALSE.) Defaults to TRUE. |
| TYPE, PACKAGE | SPECIFICATION | BOOLEAN | If TRUE, include the type or package specification in the DDL. If FALSE, omit it. Defaults to TRUE. |
| TYPE, PACKAGE | BODY | BOOLEAN | If TRUE, include the type body or package body in the DDL. If FALSE, omit it. Defaults to TRUE. |
| VIEW | FORCE | BOOLEAN | If TRUE, use the FORCE keyword in the CREATE VIEW statement. If FALSE, do not use the FORCE keyword in the CREATE VIEW statement. Defaults to TRUE. |
| OUTLINE | INSERT | BOOLEAN | If TRUE, include the INSERT statements into the OL$ dictionary tables that will create the outline and its hints. If FALSE, omit a CREATE OUTLINE statement. Defaults to FALSE.<br>Note: This object type is being deprecated. |

**Table 128-23 (Cont.) SET_TRANSFORM_PARAM: Transform Parameters for the DDL Transform**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| All objects | DEFAULT | BOOLEAN | Calling SET_TRANSFORM_PARAM with this parameter set to TRUE has the effect of resetting all parameters for the transform to their default values. Setting this FALSE has no effect. There is no default. |
| All objects | INHERIT | BOOLEAN | If TRUE, inherits session-level parameters. Defaults to FALSE. If an application calls ADD_TRANSFORM to add the DDL transform, then by default the only transform parameters that apply are those explicitly set for that transform handle. This has no effect if the transform handle is the session transform handle. |
| ROLE | REVOKE_FROM | Text | The name of a user from whom the role must be revoked. If this is a non-null string and if the CREATE ROLE statement grants you the role, a REVOKE statement is included in the DDL after the CREATE ROLE statement.<br>Note: When you issue a CREATE ROLE statement, Oracle may grant you the role. You can use this transform parameter to undo the grant.<br>Defaults to null string. |
| TABLESPACE | REUSE | BOOLEAN | If TRUE, include the REUSE parameter for data files in a tablespace to indicate that existing files can be reused. If FALSE, omit the REUSE parameter.<br>Defaults to FALSE. |
| CLUSTER, INDEX, ROLLBACK_SEGMENT, TABLE, TABLESPACE | PCTSPACE | NUMBER | A number representing the percentage by which space allocation for the object type is to be modified. The value is the number of one-hundredths of the current allocation. For example, 100 means 100%.<br>If the object type is TABLESPACE, the following size values are affected:<br>- in file specifications, the value of SIZE<br>- MINIMUM EXTENT<br>- EXTENT MANAGEMENT LOCAL UNIFORM SIZE<br>For other object types, INITIAL and NEXT are affected. |

**Table 128-23 (Cont.) SET_TRANSFORM_PARAM: Transform Parameters for the DDL Transform**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| TABLE | LOB_STORAGE | Text | Specifies the storage type to use for LOB segments. The options are as follows:<br>• SECUREFILE - LOB storage is returned as SECUREFILE<br>• BASICFILE - LOB storage is returned as BASICFILE<br>• DEFAULT - The keyword (SECUREFILE or BASICFILE) is omitted in the LOB STORE AS clause.<br>• NO_CHANGE - LOB segments are created with the same storage they had in the source database. This is the default.<br>Specifying this transform changes the LOB storage for all tables in the job, including tables that provide storage for materialized views. |
| TABLE | TABLE_COMPRESSION_CLAUSE | Text | Specifies a table compression clause (for example, COMPRESS BASIC) to use when the table is created.<br>Specify NONE to omit the table compression clause. The table will have the default compression for the tablespace.<br>Specifying this transform changes the compression type for all tables in the job, including tables that provide storage for materialized views. |

Table 128-24 describes the object type, name, datatype, and meaning of the parameters for the MODIFY transform in the SET_TRANSFORM_PARAM procedure.

**Table 128-24 SET_TRANSFORM_PARAM: Transform Parameters for the MODIFY Transform**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| All objects | OBJECT_ROW | NUMBER | A number designating the object row for an object. The object in the document that corresponds to this number will be copied to the output document.<br>This parameter is additive.<br>By default, all objects are copied to the output document. |

Table 128-25 describes the object type, name, datatype, and meaning of the parameters for the MODIFY transform in the SET_REMAP_PARAM procedure.

**Table 128-25 SET_REMAP_PARAM: Transform Parameters for the MODIFY Transform**

| Object Type | Name | Datatype | Meaning |
|---|---|---|---|
| LIBRARY, TABLESPACE, DIRECTORY | REMAP_DATAFILE | Text | Objects in the document will have their filespecs renamed as follows: any filespec matching old_value will be changed to new_value. Filespecs should *not* be enclosed in quotes.<br>This parameter is additive.<br>By default, filespecs are not renamed. |
| Named objects and all objects dependent on named objects | REMAP_NAME | Text | Any named object in the document whose name matches old_value will have its name changed to new_value.<br>Any dependent object whose base object name matches old_value will have its base schema name changed to new_value.<br>This parameter is additive.<br>By default, names are not remapped. |
| Schema Objects, Dependent Objects, Granted Objects, USER | REMAP_SCHEMA | Text | Any schema object in the document whose name matches old_value will have its schema name changed to new_value.<br>This parameter is additive.<br>By default, schemas are not remapped.<br>NOTE: The mapping may not be 100 percent complete because there are certain schema references that Import is not capable of finding. For example, Import will not find schema references embedded within the body of definitions of triggers, types, views, procedures, and packages. |
| TABLE, CLUSTER, CONSTRAINT, INDEX, ROLLBACK_SEGMENT, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG, TABLESPACE_QUOTA | REMAP_TABLESPACE | Text | Objects in the document will have their tablespaces renamed as follows: any tablespace name matching old_value will be changed to new_value.<br>This parameter is additive.<br>By default, tablespaces are not remapped.<br>Starting with Oracle Database 21c, you can use the % wildcard in place of the source tablespaces. The wildcard is applied to DDL for the object types USER, TABLE, MVIEW, MVIEW_LOG, MVIEW_ZONEMAP and CLUSTERS.<br>TRANSFORM=TABLESPACE:N is mutually exclusive with REMAP_TABLESPACE. |

**Table 128-26 SET_TRANSFORM_PARAM: Transform Parameters for the SXML Transform**

| Object type | Name | Datatype | Meaning |
|---|---|---|---|
| USER, TABLE, CLUSTER, VIEW, MATERIALIZED_VIEW PROCEDURE, FUNCTION, PACKAGE, TYPE, TRIGGER | COLLATION_CLAUSE | Text | There are three possible values:<br>• NEVER — Collation clauses are never generated.<br>• ALWAYS — Collation clauses are always generated.<br>• NON_DEFAULT — Collation clauses other than USING_NLS_COMP are generated. |

**Table 128-26 (Cont.) SET_TRANSFORM_PARAM: Transform Parameters for the SXML Transform**

| Object type | Name | Datatype | Meaning |
|---|---|---|---|
| TABLE, TYPE | OID | Boolean | If TRUE, include the Oracle Internet Directory (OID) clause in the SXML. If FALSE, omit it. Defaults to FALSE. |
| TABLE, INDEX, CLUSTER, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG. | STORAGE | Boolean | If TRUE, include storage clauses in the SXML. If FALSE, omit them. (Ignored if SEGMENT_ATTRIBUTES is FALSE.) Defaults to TRUE. |
| TABLE, INDEX, CLUSTER, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG. | TABLESPACE | Boolean | If TRUE, include tablespace clauses in the SXML. If FALSE, omit them. (Ignored if SEGMENT_ATTRIBUTES is FALSE.) Defaults to TRUE. |
| TABLE | REF_CONSTRAINTS | Boolean | If TRUE, include all referential constraints (foreign keys) in the SXML. If FALSE, omit them. Defaults to TRUE. |
| TABLE, INDEX, MATERIALIZED_VIEW | PHYSICAL_PROPERTIES | Boolean | If TRUE, include segment attributes clauses (physical attributes, storage attributes, tablespace, logging) in the SXML. If FALSE, omit them. Defaults to TRUE. |
| INDEX, CONSTRAINT, ROLLBACK_SEGMENT, CLUSTER, TABLE, TABLESPACE | SEGMENT_ATTRIBUTES | Boolean | If TRUE, include segment attributes clauses (physical attributes, storage attributes, tablespace, logging) in the SXML. If FALSE, omit them. Defaults to TRUE. |
| TABLE, INDEX | PARTITIONING | Boolean | If TRUE, include partitioning clauses in the SXML. If FALSE, omit them. Defaults to TRUE. |
| TABLE | CONSTRAINTS | Boolean | If TRUE, include all non-referential table constraints in the SXML. If FALSE, omit them. Defaults to TRUE. |

**Table 128-27 SET_TRANSFORM_PARAM: Transform Parameters for the MODIFYSXML Transform**

| Object type | Name | Datatype | Meaning |
|---|---|---|---|
| TABLE, INDEX, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG | STORAGE | Boolean | If TRUE, include storage clauses in the output SXML. If FALSE, omit them. (Ignored if SEGMENT_ATTRIBUTES is FALSE.) Defaults to TRUE. |
| TABLE, INDEX, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG | TABLESPACE | Boolean | If TRUE, include tablespace clauses in the output SXML. If FALSE, omit them. (Ignored if SEGMENT_ATTRIBUTES is FALSE.) Defaults to TRUE. |

**Table 128-27 (Cont.) SET_TRANSFORM_PARAM: Transform Parameters for the MODIFYSXML Transform**

| Object type | Name | Datatype | Meaning |
|---|---|---|---|
| TABLE | REF_CONSTRAINTS | Boolean | If TRUE, include all referential constraints (foreign keys) in the output SXML. If FALSE, omit them. Defaults to TRUE. |
| TABLE, INDEX, VIEW, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG | REMAP_NAME | Text | Any NAME element in the document that matches old_value will be changed to new_value.<br>This does not apply to column names. (See REMAP_COLUMN_NAME). |
| TABLE, INDEX, VIEW, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG | REMAP_SCHEMA | Text | Any SCHEMA element in the document matching old_value will be changed to new_value. |
| TABLE, INDEX, VIEW, MATERIALIZED_VIEW | REMAP_COLUMN_NAME | Text | Any column in the document whose name matches old_value will have its name changed to new_value. |
| TABLE, INDEX, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG | SEGMENT_ATTRIBUTES | Boolean | If TRUE, include segment attributes clauses (physical attributes, storage attributes, tablespace, logging) in the output SXML. If FALSE, omit them. Defaults to TRUE. |
| TABLE | CONSTRAINTS | Boolean | If TRUE, include all non-referential table constraints in the output SXML. If FALSE, omit them. Defaults to TRUE. |

**Table 128-28 SET_TRANSFORM_PARAM: Transform Parameters for the SXMLDDL Transform**

| Object type | Name | Datatype | Meaning |
|---|---|---|---|
| USER, TABLE, CLUSTER, VIEW, MATERIALIZED_VIEW PROCEDURE, FUNCTION, PACKAGE, TYPE, TRIGGER | COLLATION_CLAUSE | Text | There are three possible values:<br>• NEVER — Collation clauses are never generated.<br>• ALWAYS — Collation clauses are always generated.<br>• NON_DEFAULT — Collation clauses other than USING_NLS_COMP are generated. |
| TABLE | OID | Boolean | If TRUE, include OIDs in the DDL. If FALSE, omit them. (Ignored if SEGMENT_ATTRIBUTES is FALSE.) Defaults to TRUE. |
| TABLE, INDEX, CLUSTER, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG. | TABLESPACE | Boolean | If TRUE, include tablespace clauses in the DDL. If FALSE, omit them. (Ignored if SEGMENT_ATTRIBUTES is FALSE.) Defaults to TRUE. |

**Table 128-28 (Cont.) SET_TRANSFORM_PARAM: Transform Parameters for the SXMLDDL Transform**

| Object type | Name | Datatype | Meaning |
|---|---|---|---|
| TABLE, INDEX, CLUSTER, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG | STORAGE | Boolean | If TRUE, include storage clauses in the DDL. If FALSE, omit them. (Ignored if SEGMENT_ATTRIBUTES is FALSE.) Defaults to TRUE. |
| TABLE | REF_CONSTRAINTS | Boolean | If TRUE, include all referential constraints (foreign keys) in the DDL. If FALSE, omit them. Defaults to TRUE. |
| INDEX | PRESERVE_LOCAL | Boolean | If PARTITIONING is FALSE and PRESERVE_LOCAL is TRUE and the index is a locally partitioned index, include the LOCAL keyword in the DDL. Defaults to FALSE. |
| TABLE, INDEX, CLUSTER, MATERIALIZED_VIEW, MATERIALIZED_VIEW_LOG | SEGMENT_ATTRIBUTES | Boolean | If TRUE, include segment attributes clauses (physical attributes, storage attributes, tablespace, logging) in the DDL. If FALSE, omit them. Defaults to TRUE. |
| TABLESPACE | REUSE | Boolean | If TRUE, include the REUSE parameter for data files in a tablespace to indicate that existing files can be reused. If FALSE, omit the REUSE parameter.<br>Defaults to FALSE. |
| TABLE, INDEX | PARTITIONING | Boolean | If TRUE, include partitioning clauses in the DDL. If FALSE, omit them. Defaults to TRUE. |
| TABLE | CONSTRAINTS | Boolean | If TRUE, include all non-referential table constraints in the output SXML. If FALSE, omit them. Defaults to TRUE. |

### Exceptions

- INVALID_ARGVAL. A NULL or invalid value was supplied for an input parameter. The error message text identifies the parameter.
- INVALID_OPERATION. Either SET_TRANSFORM_PARAM or SET_REMAP_PARAM was called after the first call to FETCH_xxx for the OPEN context. After the first call to FETCH_xxx is made, no further calls to SET_TRANSFORM_PARAM or SET_REMAP_PARAM are permitted.
- INCONSISTENT_ARGS. The arguments are inconsistent. This can mean the following:
  - The transform parameter name is not valid for the object type associated with the OPEN context or for the transform associated with the transform handle.
  - The transform applies to all object types in a heterogeneous collection, but object_type is not part of the collection.
- TRANSFORM=TABLESPACE:N is mutually exclusive with REMAP_TABLESPACE.
- The following restrictions apply to use of REMAP_TABLESPACE transform parameter:
  - The target tablespace must be a permanent tablespace, and it must exist before the import.
  - The target tablespace cannot be a temporary tablespace.
  - The % wildcard cannot be used with multiple REMAP_TABLESPACE parameters.

- The REMAP_TABLESPACE parameter and TRANSFORM=TABLESPACE:N transform parameter are mutually exclusive.

### Usage Notes

XSLT allows parameters to be passed to stylesheets. You call SET_TRANSFORM_PARAM or SET_REMAP_PARAM to specify the value of a parameter that you want to be passed to the stylesheet, identified by transform_handle.

Normally, if you call SET_TRANSFORM_PARAMETER multiple times for the same parameter name, each call overrides the prior call. For example, the following sequence sets the STORAGE transform parameter to TRUE, because TRUE is the last value specified for SET_TRANSFORM_PARAM:

```
SET_TRANSFORM_PARAM(tr_handle,'STORAGE',false);
SET_TRANSFORM_PARAM(tr_handle,'STORAGE',true);
```

However, some transform parameters are additive. This means that all specified parameter values are applied to the document, not just the last one. For example, the OBJECT_ROW parameter to the MODIFY transform is additive. If you specify the following, then both specified rows are copied to the output document:

```
SET_TRANSFORM_PARAM(tr_handle,'OBJECT_ROW',5);
SET_TRANSFORM_PARAM(tr_handle,'OBJECT_ROW',8);
```

The REMAP_TABLESPACE parameter is also additive. If you specify the following, then tablespaces TBS1 and TBS3 are changed to TBS2 and TBS4, respectively:

```
SET_REMAP_PARAM(tr_handle,'REMAP_TABLESPACE','TBS1','TBS2');
SET_REMAP_PARAM(tr_handle,'REMAP_TABLESPACE','TBS3','TBS4');
```

The order in which the transformations are performed is undefined. For example, if you specify the following, then the result is undefined:

```
SET_REMAP_PARAM(tr_handle,'REMAP_TABLESPACE','TBS1','TBS2');
SET_REMAP_PARAM(tr_handle,'REMAP_TABLESPACE','TBS2','TBS3');
```

> **Note:**
>
> The number of remap parameters that can be specified for a MODIFY transform is limited to ten. That is, you can specify up to ten REMAP_DATAFILE parameters, up to ten REMAP_SCHEMA parameters and so on. Additional instances are ignored. To work around this limit, you can perform another DBMS_METADATA.ADD_TRANSFORM and specify additional remap parameters.

The GET_DDL, GET_DEPENDENT_DDL, and GET_GRANTED_DDL functions allow the casual browser to extract the creation DDL for an object. So that you can specify transform parameters, this package defines an enumerated constant SESSION_TRANSFORM as the handle of the DDL transform at the session level. You can call SET_TRANSFORM_PARAM using DBMS_METADATA.SESSION_TRANSFORM as the transform handle to set transform parameters for the whole session. GET_DDL, GET_DEPENDENT_DDL, and GET GRANTED_DDL inherit these parameters when they invoke the DDL transform.

> **Note:**
>
> The enumerated constant must be prefixed with the package name `DBMS_METADATA.SESSION_TRANSFORM`.