315

MGD_ID Package Types

The MGD_ID package provides an extensible framework that supports current radio-frequency ID (RFID) tags with the standard family of EPC bit encodings for the supported encoding types. The MGD_ID Package also supports new and evolving tag encodings that are not included in the current EPC standard (EPC v1.1 specification).

The MGD_ID package contains several predefined types.

> **See Also:**
>
> *Oracle Database Development Guide* for more information.

This chapter contains the following topics:

- Security Model
- Summary of Types
- Summary of MGD_ID Subprograms

The method described in this reference chapter show examples based on the examples shown in the constructor functions.

The examples in this chapter assume that the you have run the following set of commands before running the contents of each script:

```
SQL> connect / as sysdba;
Connected.
SQL> create user mgduser identified by mgduser;
SQL> grant connect, resource to mgduser;
SQL> connect mgduser
Enter password: mgduserpassword
Connected.
SQL> set serveroutput on;
```

MGD_ID Package Types Security Model

You must run the catmgd.sql script to load the DBMS_MGD_ID_UTL package and create the required Identity Code Package schema objects in the MGDSYS schema.

MGD_ID is a MGDSYS-owned object type. Any MGD_ID subprogram called from an anonymous PL/SQL block is run using the privileges of the current user.

A user must be granted connect and resource roles to use the MGD_ID object type and its subprograms.

EXECUTE privilege is granted to PUBLIC for this ADT: MGD_ID.

A public synonym, by the same name, is created for this ADT: MGD_ID.

Summary of Types

This table lists and briefly describes the `MGD_ID` Package object types.

Table 315-1 MGD_ID Package Object Types

Object Type Name	Description
MGD_ID_COMPONENT Object Type	Datatype that specifies the name and value pair attributes that define a component
MGD_ID_COMPONENT_VARRAY Object Type	Datatype that specifies a list of up to 128 components as name-value attribute pairs used in two constructor functions for creating an identity code type object
MGD_ID Object Type	Represents an `MGD_ID` object that specifies the category identifier for the code category for this identity code and its list of components

MGD_ID_COMPONENT Object Type

The `MGD_ID_COMPONENT` type is a datatype that specifies the name and value pair attributes that define a component.

Syntax

```
MGD_ID_COMPONENT as object (name  VARCHAR2(256),
                            value VARCHAR2(1024));
```

Attributes

Table 315-2 MGD_ID_COMPONENT Attributes

Attribute	Description
`name`	Name of component
`value`	Value of the component as a character

Examples

See the MGD_ID Constructor Function for an example.

MGD_ID_COMPONENT_VARRAY Object Type

The `MGD_ID_COMPONENT_VARRAY` type is a datatype that specifies a list of up to 128 components as name-value attribute pairs for use in two constructor functions for creating a product code type object with its list of components.

Syntax

```
MGD_ID_COMPONENT_VARRAY is VARRAY (128) of MGD_ID_COMPONENT;
```

Examples

See the MGD_ID Constructor Function for an example.

MGD_ID Object Type

The MGD_ID type represents an identity code in an RFID application. This type represents RFID tags with standard EPC bit encoding as well as tag encodings that are not included in the EPC standard.

Syntax

```
MGD_ID as object (category_id  VARCHAR2(256),
                  components   MGD_ID_COMPONENT_VARRAY);
```

Attributes

Table 315-3 MGD_ID Object Type Attributes

Attribute	Description
category_id	Category identifier for the code category of this code
components	List of components as name-value attributes

Methods

Table 315-5 describes the methods of the MGD_ID object type.

Table 315-4 MGD_ID Methods

Method	Description
MGD_ID constructor function	Creates an MGD_ID object based on the parameters passed in and returns self as a result
FORMAT function	Returns the string representation of the MGD_ID in the specified format
GET_COMPONENT function	Returns the string value of the specified MGD_ID component
TO_STRING function	Returns the string value of semicolon (;) separated component name value pairs of the MGD_ID object
TRANSLATE function	Returns the result of the conversion of the identifier from one format to the specified format

Examples

See the Summary of MGD_ID Subprograms section and the section about using the Identity Code package in Using the Identity Code Package in *Oracle Database Development Guide* for examples.

Summary of MGD_ID Subprograms

This table describes the subprograms in the MGD_ID object type.

All the values and names passed to the procedures defined in the MGD_ID object type are case insensitive unless otherwise mentioned. To preserve the case, enclose the values with double quotation marks.

Table 315-5 MGD_ID Object Type Subprograms

Subprogram	Description
MGD_ID Constructor Function	Creates an MGD_ID object based on the parameters passed in and returns self as a result
FORMAT Function	Returns the string representation of the MGD_ID object in the specified format
GET_COMPONENT Function	Returns the string value of the specified MGD_ID component
TO_STRING Function	Returns the string value of semicolon (;) separated component name value pairs of the MGD_ID object
TRANSLATE Function	Returns the result of the conversion of the identifier from one format to the specified format

MGD_ID Constructor Function

This constructor function constructs an identity code type object, MGD_ID. The constructor function is overloaded. The different functionality of each form of syntax is presented along with the definitions.

Syntax

Constructs an MGD_ID object type based on the category ID and a list of components.

```
MGD_ID (
   category_id      IN VARCHAR2,
   components       IN MGD_ID_COMPONENT_VARRAY)
 RETURN SELF AS RESULT DETERMINISTIC;
```

Constructs an MGD_ID object type based on the category ID, the identifier string, and the list of additional parameters required to create it.

```
MGD_ID (
   category_id        VARCHAR2,
   identifier         VARCHAR2,
   parameter_list      VARCHAR2)
 RETURN SELF AS RESULT DETERMINISTIC;
```

Constructs an MGD_ID object type based on the category name, category version, and a list of components.

```
MGD_ID (
   category_name      VARCHAR2,
   category_version   VARCHAR2,
   components         MGD_ID_COMPONENT_VARRAY)
 RETURN SELF AS RESULT DETERMINISTIC;
```

Constructs an MGD_ID object type based on the category name, category version, the identifier string, and the list of additional parameters required to create it.

```
MGD_ID (
   category_name      VARCHAR2,
   category_version   VARCHAR2,
   identifier         VARCHAR2,
   parameter_list      VARCHAR2)
 RETURN SELF AS RESULT DETERMINISTIC;
```

Parameters

Table 315-6 MGD_ID Constructor Function Parameters

Parameter	Description
`category_id`	Category identifier
`components`	List of component name value pairs
`category_name`	Category name, such as `EPC`
`category_version`	Category version. If `NULL`, the latest version for the specified category name will be used.
`identifier`	Identifier string in any format of an encoding scheme in the specified category. For example, for SGTIN-96 encoding, the identifier can be in the format of `BINARY`, `PURE_IDENTITY`, `TAG_ENCODING`, or `LEGACY`. Express this identifier as a string according to the appropriate grammar or pattern in the tag data translation (TDT) markup file. For example, a binary string consisting of characters `0` and `1`, a URI (either tag-encoding or pure-identity formats), or a serialized legacy code expressed as a string format for input, such as `gtin=00037000302414;serial=10419703` for a SGTIN coding scheme.
`parameter_list`	List of additional parameters required to create the object in the representation. The list is expressed as a parameter string containing key-value pairs, separated by the semicolon (;) as a delimiter between key-value pairs. For example, for a GTIN code, the parameter string would look as follows: `filter=3;companyprefixlength=7;taglength=96`

Usage Notes

- Use `MGD_ID_UTL.EPC_ENCODING_CATEGORY_ID` as `category_id`.
- If the category is not already registered, an error is raised.
- If the `bit_length` parameter is `NULL`, the `bit_length` is 8* the length of `bit_encoding`.
- If the component list does not contain all required components, an exception `MGD_ID_UTL.e_LackComponent` will be thrown.

Examples

The following examples construct identity code type objects.

Construct an `MGD_ID` object (SGTIN-64) passing in the category ID and a list of components.

```
--Contents of constructor11.sql
call DBMS_MGD_ID_UTL.set_proxy('www-proxy.example.com', '80');
call DBMS_MGD_ID_UTL.refresh_category('1');
select MGD_ID('1',
              MGD_ID_COMPONENT_VARRAY(
               MGD_ID_COMPONENT('companyprefix','0037000'),
               MGD_ID_COMPONENT('itemref','030241'),
               MGD_ID_COMPONENT('serial','1041970'),
               MGD_ID_COMPONENT('schemes','SGTIN-64')
              )
             ) from dual;
call DBMS_MGD_ID_UTL.remove_proxy();
```

```
SQL> @constructor11.sql
.
.
.
MGD_ID('1', MGD_ID_COMPONENT_VARRAY(MGD_ID_COMPONENT('companyprefix', '0037000'),
                                    MGD_ID_COMPONENT('itemref', '030241'),
                                    MGD_ID_COMPONENT('serial', '1041970'),
                                    MGD_ID_COMPONENT('schemes', 'SGTIN-64')))
.
.
.
```

Constructs an `MGD_ID` object (SGTIN-64) passing in the category ID, the tag identifier, and the list of additional parameters that may be required to create it.

```
--Contents of constructor22.sql
call DBMS_MGD_ID_UTL.set_proxy('www-proxy.example.com', '80');
call DBMS_MGD_ID_UTL.refresh_category('1');
select MGD_ID('1',
              'urn:epc:id:sgtin:0037000.030241.1041970',
              'filter=3;scheme=SGTIN-64') from dual;
call DBMS_MGD_ID_UTL.remove_proxy();

SQL> @constructor22.sql
.
.
.
MGD_ID('1', MGD_ID_COMPONENT_VARRAY(MGD_ID_COMPONENT('filter', '3'),
                                    MGD_ID_COMPONENT('schemes', 'SGTIN-64'),
                                    MGD_ID_COMPONENT('companyprefixlength', '7'),
                                    MGD_ID_COMPONENT('companyprefix', '0037000'),
                                    MGD_ID_COMPONENT('scheme', 'SGTIN-64'),
                                    MGD_ID_COMPONENT('serial', '1041970'),
                                    MGD_ID_COMPONENT('itemref', '030241')))
.
.
.
```

Construct an `MGD_ID` object (SGTIN-64) passing in the category name, category version (if `NULL`, then the latest version will be used), and a list of components.

```
--Contents of constructor33.sql
call DBMS_MGD_ID_UTL.set_proxy('www-proxy.example.com', '80');
call DBMS_MGD_ID_UTL.refresh_category(DBMS_MGD_ID_UTL.get_category_id('EPC', NULL));
select MGD_ID('EPC', NULL,
              MGD_ID_COMPONENT_VARRAY(
               MGD_ID_COMPONENT('companyprefix','0037000'),
               MGD_ID_COMPONENT('itemref','030241'),
               MGD_ID_COMPONENT('serial','1041970'),
               MGD_ID_COMPONENT('schemes','SGTIN-64')
              )
             ) from dual;
call DBMS_MGD_ID_UTL.remove_proxy();

SQL> @constructor33.sql
.
.
.
MGD_ID('1', MGD_ID_COMPONENT_VARRAY(MGD_ID_COMPONENT('companyprefix', '0037000'),
                                    MGD_ID_COMPONENT('itemref', '030241'),
                                    MGD_ID_COMPONENT('serial', '1041970'),
                                    MGD_ID_COMPONENT('schemes', 'SGTIN-64')))
```

```
.
.
.
```

Constructs an `MGD_ID` object (SGTIN-64) passing in the category name and category version, the tag identifier, and the list of additional parameters that may be required to create it.

```
--Contents of constructor44.sql
call DBMS_MGD_ID_UTL.set_proxy('www-proxy.example.com', '80');
call DBMS_MGD_ID_UTL.refresh_category(DBMS_MGD_ID_UTL.get_category_id('EPC', NULL));
select MGD_ID('EPC', NULL,
              'urn:epc:id:sgtin:0037000.030241.1041970',
              'filter=3;scheme=SGTIN-64') from dual;
call DBMS_MGD_ID_UTL.remove_proxy();

SQL> @constructor4.sql
.
.
.
MGD_ID('1', MGD_ID_COMPONENT_VARRAY(MGD_ID_COMPONENT('filter', '3'),
                                    MGD_ID_COMPONENT('schemes', 'SGTIN-64'),
                                    MGD_ID_COMPONENT('companyprefixlength', '7'),
                                    MGD_ID_COMPONENT('companyprefix', '0037000'),
                                    MGD_ID_COMPONENT('scheme', 'SGTIN-64'),
                                    MGD_ID_COMPONENT('serial', '1041970'),
                                    MGD_ID_COMPONENT('itemref', '030241')))
.
.
.
```

FORMAT Function

This function returns the string representation of the `MGD_ID` object in the specified format.

Syntax

```
FORMAT (parameter_list  IN VARCHAR2,
        output_format   IN VARCHAR2)
 RETURN VARCHAR2 DETERMINISTIC;
```

Parameters

Table 315-7 FORMAT Function Parameters

Parameter	Description
`parameter_list`	List of additional parameters required to create the object in the representation. The list is expressed as a parameter string containing key-value pairs, separated by the semicolon (;) as a delimiter between key-value pairs. For example, for a GTIN code, the parameter string would look as follows: `filter=3;companyprefixlength=7;taglength=96`

Table 315-7 (Cont.) FORMAT Function Parameters

Parameter	Description
output_format	One of the supported output formats into which an MGD_ID component is formatted: • BINARY • LEGACY • TAG_ENCODING • PURE_IDENTITY • ONS_HOSTNAME

Examples

See the example for the GET_COMPONENT Function.

GET_COMPONENT Function

This function returns the value of the specified MGD_ID component.

Syntax

```
GET_COMPONENT (
   component_name  IN  VARCHAR2)
 RETURN VARCHAR2 DETERMINISTIC;
```

Parameters

Table 315-8 GET_COMPONENT Function Parameter

Parameter	Description
component_name	Name of component

Usage Notes

- If the code is an invalid code, meaning its structure is not defined in the metadata table, an error is raised.
- If the code is valid, but it does not contain the required component, NULL is returned.

Examples

The following example returns the general manager, object class, and serial number components for this GID-96 identity component:

```
--Contents of get_components.sql file
call DBMS_MGD_ID_UTL.set_proxy('www-proxy.example.com', '80');
DECLARE
id  MGD_ID;
BEGIN
 DBMS_MGD_ID_UTL.set_java_logging_level(DBMS_MGD_ID_UTL.LOGGING_LEVEL_OFF);
 DBMS_MGD_ID_UTL.refresh_category(DBMS_MGD_ID_UTL.get_category_id('EPC', NULL));
    ---------------
    --PURE_IDENTIT
    ---------------
    dbms_output.put_line('..Testing constructor with pure identity');
```

```
    ------------------------------------------------------------------------
    -- PURE_IDENTITY representation can be translated to BINARY and
    -- TAG_ENCODING ONLY when BOTH scheme and filer are provided.
    ------------------------------------------------------------------------
    id := MGD_ID('EPC', NULL, 'urn:epc:id:sgtin:0037000.030241.1041970', 'scheme=SGTIN-64;filter=3');
    dbms_output.put_line(id.to_string);
    dbms_output.put_line('filter         = ' || id.get_component('filter'));
    dbms_output.put_line('company prefix = ' || id.get_component('companyprefix'));
    dbms_output.put_line('itemref        = ' || id.get_component('itemref'));
    dbms_output.put_line('serial         = ' || id.get_component('serial'));
    dbms_output.put_line('BINARY format  = ' || id.format(NULL, 'BINARY'));
    dbms_output.put_line('PURE_IDENTITY format  = ' || id.format(NULL, 'PURE_IDENTITY'));
    dbms_output.put_line('TAG_ENCODING format   = ' || id.format(NULL, 'TAG_ENCODING'));
END;
/
SHOW ERRORS;
call DBMS_MGD_ID_UTL.remove_proxy();
SQL> @get_component.sql
.
.
.
..Testing constructor with pure identity
category_id =1;filter = 3;schemes = SGTIN-64;companyprefixlength =
7;companyprefix = 0037000;scheme = SGTIN-64;serial = 1041970;itemref = 030241
filter         = 3
company prefix = 0037000
itemref        = 030241
serial         = 1041970
BINARY format  =1001100000000000001000001110110001000010000011111110011000110010
PURE_IDENTITY format  = urn:epc:id:sgtin:0037000.030241.1041970
TAG_ENCODING format   = urn:epc:tag:sgtin-64:3.0037000.030241.1041970
PL/SQL procedure successfully completed.
.
.
.
```

TO_STRING Function

This function returns the semicolon (;) separated component name value pairs of the `MGD_ID` object.

Syntax

```
TO_STRING
 RETURN VARCHAR2;
```

Examples

The following example converts the `MGD_ID` object into a string value:

```
-- Contents of tostring3.sql file
call DBMS_MGD_ID_UTL.set_proxy('www-proxy.example.com', '80');
DECLARE
id          MGD_ID;
BEGIN
 DBMS_MGD_ID_UTL.refresh_category(DBMS_MGD_ID_UTL.get_category_id('EPC', NULL));
 dbms_output.put_line('..Testing to_string');
   id := mgd_id('EPC', NULL, 'urn:epc:id:gid:0037000.30241.1041970', 'scheme=GID-96');
      DBMS_OUTPUT.PUT_LINE('mgd_id object as a string');
      DBMS_OUTPUT.PUT_LINE(id.to_string);
END;
```

```
/
SHOW ERRORS;
call DBMS_MGD_ID_UTL.remove_proxy();
connect / as sysdba;
drop user mgduser cascade;

SQL> @tostring3.sql
.
.
.
..Testing to_string
mgd_id object as a string
category_id =1;schemes = GID-96;objectclass = 30241;generalmanager =
0037000;scheme = GID-96;1 = 1;serial = 1041970
PL/SQL procedure successfully completed.
.
.
```

TRANSLATE Function

This static function translates between different representations directly without first constructing an `MGD_ID` object.

This method is overloaded. The different functionality of each form of syntax is presented along with the definitions.

Syntax

Converts the identifier in one format to another given the category name, the tag identifier, the parameter list, and the output format.

```
TRANSLATE (
   category_name    IN VARCHAR2,
   identifier       IN VARCHAR2,
   parameter_list   IN VARCHAR2,
   output_format    IN VARCHAR2)
 RETURN VARCHAR2 DETERMINISTIC;
```

Converts the identifier in one format to another given the category name, category version, the tag identifier, the parameter list, and the output format.

```
TRANSLATE (
   category_name    IN VARCHAR2,
   category_version IN VARCHAR2,
   identifier       IN VARCHAR2,
   parameter_list   IN VARCHAR2,
   output_format    IN VARCHAR2)
 RETURN VARCHAR2 DETERMINISTIC;
```

Parameters

Table 315-9 TRANSLATE Function Parameters

Parameter	Description
`category_name`	Name of category
`category_version`	Category version. If `NULL`, the latest version of the specified category name will be used.

Table 315-9 (Cont.) TRANSLATE Function Parameters

Parameter	Description
identifier	EPC identifier, expressed as a string in accordance with one of the grammars or patterns in the TDT markup file. For example, a binary string consisting of characters 0 and 1, a URI (either tag-encoding or pure-identity formats), or a serialized legacy code expressed as a string format for input, such as gtin=00037000302414;serial=10419703 for a SGTIN coding scheme.
parameter_list	List of additional parameters required to create the object in the representation. The list is expressed as a parameter string containing key-value pairs, separated by the semicolon (;) as a delimiter between key-value pairs. For example, for a GTIN code, the parameter string would look as follows: filter=3;companyprefixlength=7;taglength=96
output_format	One of the supported output formats into which an MGD_ID component shall be converted: • BINARY • LEGACY • TAG_ENCODING • PURE_IDENTITY • ONS_HOSTNAME

Usage Notes

When converting from a pure identity representation to a binary representation, the filter value must be supplied as a value using the parameter_list parameter.

Examples

The following examples translates one GID-96 representation into another:

```
-- Contents of translate1.sql file
call DBMS_MGD_ID_UTL.set_proxy('www-proxy.example.com', '80');
DECLARE
id          MGD_ID;
BEGIN
 DBMS_MGD_ID_UTL.refresh_category(DBMS_MGD_ID_UTL.get_category_id('EPC', NULL));
  dbms_output.put_line('Category ID is EPC, Identifier is BINARY, Output format is BINARY');
  dbms_output.put_line(
    mgd_id.translate('EPC',
NULL,'00110101000000000000100100001000100000000000011101100010000100000000000000000011111110011000110010'
, NULL, 'BINARY'));
  dbms_output.put_line('Category ID is EPC, Identifier is BINARY, Output format is PURE_IDENTITY');
  dbms_output.put_line(
    mgd_id.translate('EPC',
NULL,'00110101000000000000100100001000100000000000011101100010000100000000000000000011111110011000110010'
, NULL, 'PURE_IDENTITY'));
dbms_output.put_line('Category ID is EPC, Identifier is BINARY, Output format is TAG_ENCODING');
  dbms_output.put_line(
    mgd_id.translate('EPC',
NULL,'00110101000000000000100100001000100000000000011101100010000100000000000000000011111110011000110010'
, NULL, 'TAG_ENCODING'));
  dbms_output.put_line('Category ID is EPC, Identifier is TAG_ENCODING, Output format is BINARY');
  dbms_output.put_line(
    mgd_id.translate('EPC', NULL,
                  'urn:epc:tag:gid-96:0037000.30241.1041970',
```

```
                    NULL, 'BINARY'));
  dbms_output.put_line('Category ID is EPC, Identifier is TAG_ENCODING, Output format is
PURE_IDENTITY');
  dbms_output.put_line(
    mgd_id.translate('EPC', NULL,
                    'urn:epc:tag:gid-96:0037000.30241.1041970',
                    NULL, 'PURE_IDENTITY'));
dbms_output.put_line('Category ID is EPC, Identifier is TAG_ENCODING, Output format is TAG_ENCODING');
  dbms_output.put_line(
    mgd_id.translate('EPC', NULL,
                    'urn:epc:tag:gid-96:0037000.30241.1041970',
                    NULL, 'TAG_ENCODING'));
  dbms_output.put_line('Category ID is EPC, Identifier is PURE_IDENTITY, Output format is BINARY');
  dbms_output.put_line(
    mgd_id.translate('EPC', NULL,
                    'urn:epc:id:gid:0037000.30241.1041970',
                    NULL, 'BINARY'));
  dbms_output.put_line('Category ID is EPC, Identifier is PURE_IDENTITY, Output format is
PURE_IDENTITY');
  dbms_output.put_line(
    mgd_id.translate('EPC', NULL,
                    'urn:epc:id:gid:0037000.30241.1041970',
                    NULL, 'PURE_IDENTITY'));
  dbms_output.put_line('Category ID is EPC, Identifier is PURE_IDENTITY, Output format is TAG_ENCODING');
  dbms_output.put_line(
    mgd_id.translate('EPC', NULL,
                    'urn:epc:id:gid:0037000.30241.1041970',
                    NULL, 'TAG_ENCODING'));
END;
/
SHOW ERRORS;
call DBMS_MGD_ID_UTL.remove_proxy();

SQL> @translate1.sql
.
.
.
Category ID is EPC, Identifier is BINARY, Output format is BINARY
00110101000000000000100100001000100000000000011101100010000100000000000000000011111110011000110010
Category ID is EPC, Identifier is BINARY, Output format is PURE_IDENTITY
urn:epc:id:gid:37000.30241.1041970
Category ID is EPC, Identifier is BINARY, Output format is TAG_ENCODING
urn:epc:tag:gid-96:37000.30241.1041970
Category ID is EPC, Identifier is TAG_ENCODING, Output format is BINARY
00110101000000000000100100001000100000000000011101100010000100000000000000000011111110011000110010
Category ID is EPC, Identifier is TAG_ENCODING, Output format is PURE_IDENTITY
urn:epc:id:gid:0037000.30241.1041970
Category ID is EPC, Identifier is TAG_ENCODING, Output format is TAG_ENCODING
urn:epc:tag:gid-96:0037000.30241.1041970
Category ID is EPC, Identifier is PURE_IDENTITY, Output format is BINARY
00110101000000000000100100001000100000000000011101100010000100000000000000000011111110011000110010
Category ID is EPC, Identifier is PURE_IDENTITY, Output format is PURE_IDENTITY
urn:epc:id:gid:0037000.30241.1041970
Category ID is EPC, Identifier is PURE_IDENTITY, Output format is TAG_ENCODING
urn:epc:tag:gid-96:0037000.30241.1041970
PL/SQL procedure successfully completed.
.
.
.
```