# 130
# DBMS_MGD_ID_UTL

The DBMS_MGD_ID_UTL package contains various utility functions and procedures.

These consist of the following utility subprograms:

- A logging utility that sets and gets Java and PL/SQL logging levels.
- A proxy utility consisting of two procedures used to set and unset the host and port of the proxy server.
- A metadata utility consisting of functions and procedures used for managing metadata.

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

This chapter describes each of these utility subprograms and contains the following topics:



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

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

## DBMS_MGD_ID_UTL Security Model

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

DBMS_MGD_ID_UTL is a MGDSYS-owned package. Any DBMS_MGD_ID_UTL 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 DBMS_MGD_ID_UTL package and its subprograms.

EXECUTE privilege is granted to PUBLIC for these ADTs: MGD_ID, MGD_ID_COMPONENT, MGD_ID_COMPONENT_VARRAY, and for this package DBMS_MGD_ID_UTL.

SELECT or READ privilege is granted to PUBLIC for these read-only views: MGD_ID_CATEGORY and MGD_ID_SCHEME and for these metadata views: USER_MGD_ID_CATEGORY and USER_MGD_ID_SCHEME, and for table MGD_ID_XML_VALIDATOR, and for sequence MGD$SEQUENCE_CATEGORY.

INSERT, UPDATE and DELETE privilege is granted to PUBLIC for these metadata views: USER_MGD_ID_CATEGORY and USER_MGD_ID_SCHEME.

Public synonyms, by the same name, are created for these ADTs: MGD_ID, MGD_ID_COMPONENT, MGD_ID_COMPONENT_VARRAY and for this package DBMS_MGD_ID_UTL, as well as for these read-only views: MGD_ID_CATEGORY and MGD_ID_SCHEME and for these metadata views: USER_MGD_ID_CATEGORY and USER_MGD_ID_SCHEME, and for table MGD_ID_XML_VALIDATOR.

# DBMS_MGD_ID_UTL Constants

The DBMS_MGD_ID_UTL package defines several constants for specifying parameter values.

These constants are shown in the following tables.

**Table 130-1 DBMS_MGD_ID_UTL Constants — Installed Category IDs and Names**

| Name | Value |
|---|---|
| EPC_ENCODING_CATEGORY_ID | 1 |
| EPC_ENCODING_CATEGORY_NAME | EPC |

**Table 130-2 DBMS_MGD_ID_UTL Constants — Logging Levels**

| Name | Value |
|---|---|
| LOGGING_LEVEL_OFF | 0 |
| LOGGING_LEVEL_SEVERE | 1 |
| LOGGING_LEVEL_WARNING | 2 |
| LOGGING_LEVEL_INFO | 3 |
| LOGGING_LEVEL_FINE | 4 |
| LOGGING_LEVEL_FINER | 5 |
| LOGGING_LEVEL_FINEST | 6 |
| LOGGING_LEVEL_ALL | 7 |

# DBMS_MGD_ID_UTL Exceptions

The table in this topic lists the DBMS_MGD_ID_UTL exceptions.

**Table 130-3 Exceptions Raised by DBMS_MGD_ID_UTL Package**

| Name | Error Code | Description |
|---|---|---|
| TDTJavaException | -55200 | During the tag data translation, a Java exception was raised. |
| TDTCategoryNotFound | -55201 | The specified category was not found. |
| TDTSchemeNotFound | -55202 | During the tag data translation, the specified scheme was not found. |
| TDTLevelNotFound | -55203 | During the tag data translation, the specified level was not found. |
| TDTOptionNotFound | -55204 | During the tag data translation, the specified option was not found. |
| TDTFieldValidationException | -55205 | During the tag data translation, the validation operation failed on a field. |
| TDTUndefinedField | -55206 | During the tag data translation, an undefined field was detected. |
| TDTRuleEvaluationFailed | -55207 | During the tag data translation, the rule evaluation operation failed. |
| TDTTooManyMatchingLevels | -55208 | During the tag data translation, too many matching levels were found. |

# Summary of DBMS_MGD_ID_UTL Subprograms

This table describes the utility subprograms in the DBMS_MGD_ID_UTL package.

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

**Table 130-4 DBMS_MGD_ID_UTL Package Subprograms**

| Subprogram | Description |
|---|---|
| ADD_SCHEME Procedure | Adds a tag data translation scheme to an existing category |
| CREATE_CATEGORY Function | Creates a new category or a new version of a category |
| EPC_TO_ORACLE_SCHEME Function | Converts the EPCglobal tag data translation (TDT) XML to Oracle tag data translation XML |
| GET_CATEGORY_ID Function | Returns the category ID given the category name and the category version |
| GET_COMPONENTS Function | Returns all relevant separated component names separated by semicolon (';') for the specified scheme |
| GET_ENCODINGS Function | Returns a list of semicolon (';') separated encodings (formats) for the specified scheme |
| GET_JAVA_LOGGING_LEVEL Function | Returns an integer representing the current Java trace logging level |
| GET_PLSQL_LOGGING_LEVEL Function | Returns an integer representing the current PL/SQL trace logging level |
| GET_SCHEME_NAMES Function | Returns a list of semicolon (';') separated scheme names for the specified category |
| GET_TDT_XML Function | Returns the Oracle tag data translation XML for the specified scheme |

**Table 130-4 (Cont.) DBMS_MGD_ID_UTL Package Subprograms**

| Subprogram | Description |
|---|---|
| GET_VALIDATOR Function | Returns the Oracle Database tag data translation schema |
| REFRESH_CATEGORY Function | Refreshes the metadata information on the Java stack for the specified category |
| REMOVE_CATEGORY Procedure | Removes a category including all the related TDT XML if the value of `category_version` parameter is NULL |
| REMOVE_PROXY Procedure | Unsets the host and port of the proxy server |
| REMOVE_SCHEME Procedure | Removes a tag data translation scheme from a category |
| SET_JAVA_LOGGING_LEVEL Procedure | Sets the Java logging level |
| SET_PLSQL_LOGGING_LEVEL Procedure | Sets the PL/SQL tracing logging level |
| SET_PROXY Procedure | Sets the host and port of the proxy server for Internet access |
| VALIDATE_SCHEME Function | Validates the input tag data translation XML against the Oracle tag data translation schema |

# ADD_SCHEME Procedure

This procedure adds a tag data translation scheme to an existing category.

### Syntax

```
DBMS_MGD_ID_UTL.ADD_SCHEME (
   category_id IN  VARCHAR2,
   tdt_xml     IN  CLOB);
```

### Parameters

**Table 130-5 ADD_SCHEME Procedure Parameters**

| Parameter | Description |
|---|---|
| `category_id` | Category ID |
| `tdt_xml` | Tag data translation XML |

### Examples

This example performs the following actions:

1. Creates a category.
2. Adds a contractor scheme and an employee scheme to the `MGD_SAMPLE_CATEGORY` category.
3. Validates the `MGD_SAMPLE_CATEGORY` scheme.
4. Tests the tag translation of the contractor scheme and the employee scheme.
5. Removes the contractor scheme.
6. Tests the tag translation of the contractor scheme and this returns the expected exception for the removed contractor scheme.
7. Tests the tag translation of the employee scheme and this returns the expected values.

8. Removes the `MGD_SAMPLE_CATEGORY` category.

```
--contents of add_scheme2.sql
SET LINESIZE 160
-----------------------------------------------------------------------
---CREATE CATEGORY, ADD_SCHEME, REMOVE_SCHEME, REMOVE_CATEGORY-------
-----------------------------------------------------------------------
DECLARE
  amt          NUMBER;
  buf          VARCHAR2(32767);
  pos          NUMBER;
  tdt_xml      CLOB;
  validate_tdtxml VARCHAR2(1042);
  category_id  VARCHAR2(256);
BEGIN
  -- remove the testing category if already existed
  DBMS_MGD_ID_UTL.remove_category('MGD_SAMPLE_CATEGORY', '1.0');
  -- Step 1. Create the testing category 'MGD_SAMPLE_CATEGORY', version 1.0.
  category_id := DBMS_MGD_ID_UTL.CREATE_CATEGORY('MGD_SAMPLE_CATEGORY', '1.0', 'Oracle',
'http://www.example.com/mgd/sample');
  -- Step 2. Add contractor scheme to the category.
  DBMS_LOB.CREATETEMPORARY(tdt_xml, true);
  DBMS_LOB.OPEN(tdt_xml, DBMS_LOB.LOB_READWRITE);

  buf := '<?xml version="1.0" encoding="UTF-8"?>
<TagDataTranslation version="0.04" date="2005-04-18T16:05:00Z"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema"
                    xmlns="oracle.mgd.idcode">
 <scheme name="CONTRACTOR_TAG" optionKey="1" xmlns="">
  <level type="URI" prefixMatch="example.contractor.">
   <option optionKey="1" pattern="example.contractor.([0-9]*).([0-9]*)"
           grammar="''example.contractor.'' contractorID ''.'' divisionID">
    <field seq="1" characterSet="[0-9]*" name="contractorID"/>
    <field seq="2" characterSet="[0-9]*" name="divisionID"/>
   </option>
  </level>
  <level type="BINARY" prefixMatch="11">
   <option optionKey="1" pattern="11([01]{7})([01]{6})"
           grammar="''11'' contractorID divisionID ">
    <field seq="1" characterSet="[01]*" name="contractorID"/>
    <field seq="2" characterSet="[01]*" name="divisionID"/>
   </option>
  </level>
 </scheme>
</TagDataTranslation>';

  amt := length(buf);
  pos := 1;
  DBMS_LOB.WRITE(tdt_xml, amt, pos, buf);
  DBMS_LOB.CLOSE(tdt_xml);

  DBMS_MGD_ID_UTL.ADD_SCHEME(category_id, tdt_xml);

  -- Add the employee scheme to the category.
  DBMS_LOB.CREATETEMPORARY(tdt_xml, true);
  DBMS_LOB.OPEN(tdt_xml, DBMS_LOB.LOB_READWRITE);

  buf := '<?xml version="1.0" encoding="UTF-8"?>
<TagDataTranslation version="0.04" date="2005-04-18T16:05:00Z"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema"
                    xmlns="oracle.mgd.idcode">
 <scheme name="EMPLOYEE_TAG" optionKey="1" xmlns="">
```

```
  <level type="URI" prefixMatch="example.employee.">
   <option optionKey="1" pattern="example.employee.([0-9]*).([0-9]*)"
           grammar="''example.employee.'' employeeID ''.'' divisionID">
    <field seq="1" characterSet="[0-9]*" name="employeeID"/>
    <field seq="2" characterSet="[0-9]*" name="divisionID"/>
   </option>
  </level>
  <level type="BINARY" prefixMatch="01">
   <option optionKey="1" pattern="01([01]{7})([01]{6})"
           grammar="''01'' employeeID divisionID ">
    <field seq="1" characterSet="[01]*" name="employeeID"/>
    <field seq="2" characterSet="[01]*" name="divisionID"/>
   </option>
  </level>
 </scheme>
</TagDataTranslation>';

  amt := length(buf);
  pos := 1;
  DBMS_LOB.WRITE(tdt_xml, amt, pos, buf);
  DBMS_LOB.CLOSE(tdt_xml);
  DBMS_MGD_ID_UTL.ADD_SCHEME(category_id, tdt_xml);

  -- Step 3. Validate the scheme.
  dbms_output.put_line('Validate the MGD_SAMPLE_CATEGORY Scheme');
  validate_tdtxml := DBMS_MGD_ID_UTL.validate_scheme(tdt_xml);
  dbms_output.put_line(validate_tdtxml);
  dbms_output.put_line('Length of scheme xml is: '||DBMS_LOB.GETLENGTH(tdt_xml));

  -- Step 4. Test tag translation of contractor scheme.
  dbms_output.put_line(
    mgd_id.translate('MGD_SAMPLE_CATEGORY', NULL,
                     'example.contractor.123.45',
                     NULL, 'BINARY'));

  dbms_output.put_line(
    mgd_id.translate('MGD_SAMPLE_CATEGORY', NULL,
                     '111111011101101',
                     NULL, 'URI'));

  -- Test tag translation of employee scheme.
  dbms_output.put_line(
    mgd_id.translate('MGD_SAMPLE_CATEGORY', NULL,
                     'example.employee.123.45',
                     NULL, 'BINARY'));

  dbms_output.put_line(
    mgd_id.translate('MGD_SAMPLE_CATEGORY', NULL,
                     '0111110111101101',
                     NULL, 'URI'));

  DBMS_MGD_ID_UTL.REMOVE_SCHEME(category_id, 'CONTRACTOR_TAG');

  -- Step 6. Test tag translation of contractor scheme. Doesn't work any more.
  BEGIN
    dbms_output.put_line(
      mgd_id.translate('MGD_SAMPLE_CATEGORY', NULL,
                       'example.contractor.123.45',
                       NULL, 'BINARY'));

    dbms_output.put_line(
      mgd_id.translate('MGD_SAMPLE_CATEGORY', NULL,
```

```
                    '111111011101101',
                    NULL, 'URI'));
  EXCEPTION
    WHEN others THEN
      dbms_output.put_line('Contractor tag translation failed: '||SQLERRM);
  END;

  -- Step 7. Test tag translation of employee scheme. Still works.
  BEGIN
    dbms_output.put_line(
      mgd_id.translate('MGD_SAMPLE_CATEGORY', NULL,
                    'example.employee.123.45',
                    NULL, 'BINARY'));
    dbms_output.put_line(
      mgd_id.translate('MGD_SAMPLE_CATEGORY', NULL,
                    '011111011101101',
                    NULL, 'URI'));
  EXCEPTION
    WHEN others THEN
      dbms_output.put_line('Employee tag translation failed: '||SQLERRM);
  END;

  -- Step 8. Remove the testing category, which also removes all the associated schemes
  DBMS_MGD_ID_UTL.remove_category('MGD_SAMPLE_CATEGORY', '1.0');
END;
/
SHOW ERRORS;

SQL> @add_scheme3.sql
.
.
.
Validate the MGD_SAMPLE_CATEGORY Scheme
EMPLOYEE_TAG;URI,BINARY;divisionID,employeeID
Length of scheme xml is: 933
111111011101101
example.contractor.123.45
011111011101101
example.employee.123.45
Contractor tag translation failed: ORA-55203: Tag data translation level not found
ORA-06512: at "MGDSYS.DBMS_MGD_ID_UTL", line 54
ORA-06512: at "MGDSYS.MGD_ID", line 242
ORA-29532: Java call terminated by uncaught Java
exception: oracle.mgd.idcode.exceptions.TDTLevelNotFound: Matching level not
found for any configured scheme
011111011101101
example.employee.123.45
.
.
.
```

# CREATE_CATEGORY Function

This function creates a new category or a new version of a category.

### Syntax

```
DBMS_MGD_ID_UTL.CREATE_CATEGORY (
   category_name    IN  VARCHAR2,
   category_version IN  VARCHAR2,
   agency           IN  VARCHAR2,
```

```
  URI              IN  VARCHAR2)
 RETURN VARCHAR2;
```

### Parameters

**Table 130-6 CREATE_CATEGORY Function Parameters**

| Parameter | Description |
|---|---|
| category_name | Name of category |
| category_version | Category version |
| agency | Organization that owns the category. For example, EPCglobal owns the category EPC. |
| URI | URI that provides additional information about the category |

### Usage Notes

The return value is the category ID.

### Examples

See the ADD_SCHEME Procedure for an example of creating the MGD_SAMPLE_CATEGORY category.

## EPC_TO_ORACLE_SCHEME Function

This function converts the EPCglobal tag data translation (TDT) XML to Oracle Database tag data translation XML.

### Syntax

```
DBMS_MGD_ID_UTL.EPC_TO_ORACLE_SCHEME (
   xml_scheme IN CLOB)
 RETURN CLOB;
```

### Parameters

**Table 130-7 EPC_TO_ORACLE_SCHEME Function Parameters**

| Parameter | Description |
|---|---|
| xml_scheme | Name of EPC tag scheme to be converted |

### Usage Notes

The return value is the contents of the CLOB containing the Oracle Datanase tag data translation XML.

### Examples

The following example converts standard EPCglobal Tag Data Translation (TDT) files into Oracle Database TDT files:

```
--Contents of MGD_ID_DOC2.sql
----------------------------
-- EPC_TO_ORACLE_SCHEME   --
----------------------------
```

```
call DBMS_MGD_ID_UTL.set_proxy('www-proxy.example.com', '80');

BEGIN
 DBMS_JAVA.set_output(1000000);
 DBMS_OUTPUT.ENABLE(1000000);
 DBMS_MGD_ID_UTL.set_java_logging_level(DBMS_MGD_ID_UTL.LOGGING_LEVEL_SEVERE);
END;
/

DECLARE
  epcScheme                 CLOB;
  oracleScheme              CLOB;
  amt                       NUMBER;
  buf                       VARCHAR2(32767);
  pos                       NUMBER;
  seq                       BINARY_INTEGER;
  validate_epcscheme        VARCHAR2(256);
  validate_oraclescheme     VARCHAR2(256);
BEGIN

  DBMS_LOB.CREATETEMPORARY(epcScheme, true);
  DBMS_LOB.OPEN(epcScheme, DBMS_LOB.LOB_READWRITE);

  buf := '<?xml version="1.0" encoding="UTF-8"?>
<epcTagDataTranslation version="0.04" date="2005-04-18T16:05:00Z"
                       epcTDSVersion="1.1r1.27"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema"
                       xsi:noNamespaceSchemaLocation="EpcTagDataTranslation.xsd">
  <scheme name="GID-96" optionKey="1" tagLength="96">
    <level type="BINARY" prefixMatch="00110101"
           requiredFormattingParameters="taglength">
      <option optionKey="1" pattern="00110101([01]{28})([01]{24})([01]{36})"
              grammar="''00110101'' generalmanager objectclass serial">
        <field seq="1" decimalMinimum="0" decimalMaximum="268435455"
               characterSet="[01]*" bitLength="28" name="generalmanager"/>
        <field seq="2" decimalMinimum="0" decimalMaximum="16777215"
               characterSet="[01]*" bitLength="24" name="objectclass"/>
        <field seq="3" decimalMinimum="0" decimalMaximum="68719476735"
               characterSet="[01]*" bitLength="36" name="serial"/>
      </option>
    </level>
    <level type="TAG_ENCODING" prefixMatch="urn:epc:tag:gid-96"
           requiredFormattingParameters="taglength">
      <option optionKey="1"
              pattern="urn:epc:tag:gid-96:([0-9]*)\.([0-9]*)\.([0-9]*)"
              grammar="''urn:epc:tag:gid-96:'' generalmanager ''.'' objectclass ''.'' serial">
        <field seq="1" decimalMinimum="0" decimalMaximum="268435455"
               characterSet="[0-9]*" name="generalmanager"/>
        <field seq="2" decimalMinimum="0" decimalMaximum="16777215"
               characterSet="[0-9]*" name="objectclass"/>
        <field seq="3" decimalMinimum="0" decimalMaximum="68719476735"
               characterSet="[0-9]*" name="serial"/>
      </option>
    </level>
    <level type="PURE_IDENTITY" prefixMatch="urn:epc:id:gid">
      <option optionKey="1"
              pattern="urn:epc:id:gid:([0-9]*)\.([0-9]*)\.([0-9]*)"
              grammar="''urn:epc:id:gid:'' generalmanager ''.'' objectclass ''.'' serial">
        <field seq="1" decimalMinimum="0" decimalMaximum="268435455"
               characterSet="[0-9]*" name="generalmanager"/>
        <field seq="2" decimalMinimum="0" decimalMaximum="16777215"
               characterSet="[0-9]*" name="objectclass"/>
```

```
        <field seq="3" decimalMinimum="0" decimalMaximum="68719476735"
               characterSet="[0-9]*" name="serial"/>
      </option>
    </level>
    <level type="LEGACY" prefixMatch="generalmanager=">
      <option optionKey="1"
              pattern="generalmanager=([0-9]*);objectclass=([0-9]*);serial=([0-9]*)"
              grammar="''generalmanager=''generalmanager'';objectclass=''objectclass '';serial=''
serial">
        <field seq="1" decimalMinimum="0" decimalMaximum="268435455"
               characterSet="[0-9]*" name="generalmanager"/>
        <field seq="2" decimalMinimum="0" decimalMaximum="16777215"
               characterSet="[0-9]*" name="objectclass"/>
        <field seq="3" decimalMinimum="0" decimalMaximum="68719476735"
               characterSet="[0-9]*" name="serial"/>
      </option>
    </level>
  </scheme>
</epcTagDataTranslation>';
  amt := length(buf);
  pos := 1;
  DBMS_LOB.WRITE(epcScheme, amt, pos, buf);
  DBMS_LOB.CLOSE(epcScheme);
    oracleScheme := DBMS_MGD_ID_UTL.epc_to_oracle_scheme(epcScheme);
  dbms_output.put_line('Length of oracle scheme xml is: '||DBMS_LOB.GETLENGTH(oracleScheme));
  dbms_output.put_line(DBMS_LOB.SUBSTR(oracleScheme, DBMS_LOB.GETLENGTH(oracleScheme), 1));
  dbms_output.put_line(' ');
  dbms_output.put_line('Validate the Oracle Scheme');
  validate_oraclescheme := DBMS_MGD_ID_UTL.validate_scheme(oracleScheme);
  dbms_output.put_line('Validation result: '||validate_oraclescheme);
END;
/
SHOW ERRORS;

SQL> @mgd_id_doc2.sql
PL/SQL procedure successfully completed.

Length of oracle scheme xml is: 2475
<?xml version = '1.0' encoding = 'UTF-8'?>
<TagDataTranslation version="0.04"
date="2005-04-18T16:05:00Z" xmlns:xsi="http://www.w3.org/2001/XMLSchema"
xmlns="oracle.mgd.idcode"><scheme name="GID-96" optionKey="1" xmlns=""><level
type="BINARY" prefixMatch="00110101" requiredFormattingParameters=""><option
optionKey="1" pattern="00110101([01]{28})([01]{24})([01]{36})"
grammar="'00110101' generalmanager objectclass serial"><field seq="1"
decimalMinimum="0" decimalMaximum="268435455" characterSet="[01]*"
bitLength="28" name="generalmanager"/><field seq="2" decimalMinimum="0"
decimalMaximum="16777215" characterSet="[01]*" bitLength="24"
name="objectclass"/><field seq="3" decimalMinimum="0"
decimalMaximum="68719476735" characterSet="[01]*" bitLength="36"
name="serial"/></option></level><level type="TAG_ENCODING"
prefixMatch="urn:epc:tag:gid-96" requiredFormattingParameters=""><option
optionKey="1" pattern="urn:epc:tag:gid-96:([0-9]*)\.([0-9]*)\.([0-9]*)"
grammar="'urn:epc:tag:gid-96:' generalmanager '.' objectclass '.' serial"><field
seq="1" decimalMinimum="0" decimalMaximum="268435455" characterSet="[0-9]*"
name="generalmanager"/><field seq="2" decimalMinimum="0"
decimalMaximum="16777215" characterSet="[0-9]*" name="objectclass"/><field
seq="3" decimalMinimum="0" decimalMaximum="68719476735" characterSet="[0-9]*"
name="serial"/></option></level><level type="PURE_IDENTITY"
prefixMatch="urn:epc:id:gid"><option optionKey="1"
pattern="urn:epc:id:gid:([0-9]*)\.([0-9]*)\.([0-9]*)" grammar="'urn:epc:id:gid:'
generalmanager '.' objectclass '.' serial"><field seq="1" decimalMinimum="0"
```

```
decimalMaximum="268435455" characterSet="[0-9]*" name="generalmanager"/><field
seq="2" decimalMinimum="0" decimalMaximum="16777215" characterSet="[0-9]*"
name="objectclass"/><field seq="3" decimalMinimum="0"
decimalMaximum="68719476735" characterSet="[0-9]*"
name="serial"/></option></level><level type="LEGACY"
prefixMatch="generalmanager="><option optionKey="1"
pattern="generalmanager=([0-9]*);objectclass=([0-9]*);serial=([0-9]*)"
grammar="'generalmanager='generalmanager';objectclass='objectclass ';serial='
serial"><field seq="1" decimalMinimum="0" decimalMaximum="268435455"
characterSet="[0-9]*" name="generalmanager"/><field seq="2" decimalMinimum="0"
decimalMaximum="16777215" characterSet="[0-9]*" name="objectclass"/><field
seq="3" decimalMinimum="0" decimalMaximum="68719476735" characterSet="[0-9]*"
name="serial"/></option></level></scheme></TagDataTranslation>
Validate the Oracle Scheme
Validation result:
GID-96;LEGACY,TAG_ENCODING,PURE_IDENTITY,BINARY;objectclass,generalmanager,serial,

PL/SQL procedure successfully completed.
.
.
.
```

# GET_CATEGORY_ID Function

This function returns the category ID for a given category name and category version.

### Syntax

```
DBMS_MGD_ID_UTL.GET_CATEGORY_ID (
   category_name     IN  VARCHAR2,
   category_version  IN  VARCHAR2)
 RETURN VARCHAR2;
```

### Parameters

**Table 130-8 GET_CATEGORY_ID Function Parameters**

| Parameter | Description |
|---|---|
| `category_name` | Name of category |
| `category_version` | Category version |

### Usage Notes

- If the value of `category_version` is NULL, then the ID of the latest version of the specified category is returned.
- The return value is the category ID for the specified category name.

### Examples

The following example returns a category ID given a category name and its version:

```
-- Contents of get_category1.sql file
SELECT DBMS_MGD_ID_UTL.get_category_id('EPC', NULL) FROM DUAL;

SQL> @get_category1.sql
.
.
.
```

```
DBMS_MGD_ID_UTL.GET_CATEGORY_ID('EPC',NULL)-----------------------------------------------------------------
-----------------1
.
.
.
```

# GET_COMPONENTS Function

This function returns all relevant separated component names separated by semicolon (;) for the specified scheme.

### Syntax

```
DBMS_MGD_ID_UTL.GET_COMPONENTS (
   category_id IN VARCHAR2,
   scheme_name IN VARCHAR2)
 RETURN VARCHAR2;
```

### Parameters

**Table 130-9 GET_COMPONENTS Function Parameters**

| Parameter | Description |
|---|---|
| category_id | Category ID |
| scheme_name | Name of scheme |

### Usage Notes

The return value contains the component names separated by a semicolon (;) for the specified scheme.

### Examples

The following example gets the components:

```
--Contents of get_components.sql
DECLARE
  id             mgd_id;
  getcomps       VARCHAR2(1000);
  getencodings   VARCHAR2(1000);
  getschemenames VARCHAR2(1000);
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));
 getcomps := DBMS_MGD_ID_UTL.get_components(1,'SGTIN-64');
 dbms_output.put_line('Component names are: ' || getcomps);
 getencodings := DBMS_MGD_ID_UTL.get_encodings(1,'SGTIN-64');
 dbms_output.put_line('Encodings are: ' || getencodings);
 getschemenames := DBMS_MGD_ID_UTL.get_scheme_names(1);
 dbms_output.put_line('Scheme names are: ' || getschemenames);
END;
/
SHOW ERRORS;

SQL> @get_components.sql
.
.
.
Component names are:
```

```
filter,gtin,companyprefixlength,companyprefix,companyprefixindex,itemref,serial
Encodings are: ONS_HOSTNAME,LEGACY,TAG_ENCODING,PURE_IDENTITY,BINARY
Scheme names are:
GIAI-64,GIAI-96,GID-96,GRAI-64,GRAI-96,SGLN-64,SGLN-96,SGTIN-64,SGTIN-96,SSCC-64
,SSCC-96,USDOD-64,USDOD-96
PL/SQL procedure successfully completed.
.
.
.
```

# GET_ENCODINGS Function

This function returns a list of semicolon (;) separated encodings (formats) for the specified scheme.

### Syntax

```
DBMS_MGD_ID_UTL.GET_ENCODINGS (
   category_id IN VARCHAR2,
   scheme_name IN VARCHAR2)
 RETURN VARCHAR2;
```

### Parameters

**Table 130-10 GET_ENCODINGS Function Parameters**

| Parameter | Description |
|---|---|
| category_id | Category ID |
| scheme_name | Name of scheme |

### Usage Notes

The return value contains the encodings separated by a semicolon (;) for the specified scheme.

### Examples

See the GET_COMPONENTS Function for an example.

# GET_JAVA_LOGGING_LEVEL Function

This function returns an integer representing the current trace logging level.

### Syntax

```
DBMS_MGD_ID_UTL.GET_JAVA_LOGGING_LEVEL
 RETURN INTEGER;
```

### Usage Notes

The return value is the integer value denoting the current Java logging level.

### Examples

The following example gets the Java logging level.

```
--Contents of getjavaloggінglevel.sql
DECLARE
```

```
     loglevel   NUMBER;
BEGIN
  DBMS_MGD_ID_UTL.set_java_logging_level(DBMS_MGD_ID_UTL.LOGGING_LEVEL_OFF);
  loglevel := DBMS_MGD_ID_UTL.get_java_logging_level();
  dbms_output.put_line('Java logging level = ' ||loglevel);
END;
/
SHOW ERRORS;

SQL> @getjavalogginglevel.sql
.
.
.
Java logging level = 0
PL/SQL procedure successfully completed.
.
.
.
```

# GET_PLSQL_LOGGING_LEVEL Function

This function returns an integer representing the current PL/SQL trace logging level.

### Syntax

```
DBMS_MGD_ID_UTL.GET_PLSQL_LOGGING_LEVEL
 RETURN INTEGER;

PRAGMA restrict_references(get_plsql_logging_level, WNDS);
```

### Usage Notes

The return value is the integer value denoting the current PL/SQL logging level.

### Examples

The following example gets the PL/SQL logging level.

```
--Contents of getplsqllogginglevel.sql
DECLARE
     loglevel   NUMBER;
BEGIN
  DBMS_MGD_ID_UTL.set_plsql_logging_level(0);
  loglevel := DBMS_MGD_ID_UTL.get_plsql_logging_level();
  dbms_output.put_line('PL/SQL logging level = ' ||loglevel);
END;
/
SHOW ERRORS;

SQL> @getplsqllogginglevel.sql
.
.
.
PL/SQL logging level = 0
PL/SQL procedure successfully completed.
.
.
.
```

# GET_SCHEME_NAMES Function

This function returns a list of semicolon (;) separated scheme names for the specified category.

### Syntax

```
DBMS_MGD_ID_UTL.GET_SCHEME_NAMES (
   category_id IN VARCHAR2)
 RETURN VARCHAR2;
```

### Parameters

**Table 130-11 GET_SCHEME_NAMES Function Parameters**

| Parameter | Description |
|---|---|
| category_id | Category ID |

### Usage Notes

The return value contains the scheme names for the specified category ID.

### Examples

See the GET_COMPONENTS Function for an example.

# GET_TDT_XML Function

This function returns the Oracle Database tag data translation XML for the specified scheme.

### Syntax

```
DBMS_MGD_ID_UTL.GET_TDT_XML (
   category_id  IN VARCHAR2,
   scheme_name  IN VARCHAR2)
 RETURN CLOB;
```

### Parameters

**Table 130-12 GET_TDT_XML Function Parameters**

| Parameter | Description |
|---|---|
| category_id | Category ID |
| scheme_name | Name of scheme |

### Usage Notes

The return value contains the Oracle Database tag data translation XML for the specified scheme.

### Examples

The following example gets the Oracle Database TDT XML for the specified scheme:

```
--Contents of get_tdtxml.sql
DECLARE
  gettdtxml      CLOB;

BEGIN
  gettdtxml := DBMS_MGD_ID_UTL.get_tdt_xml(1,'SGTIN-64');
  dbms_output.put_line('Length of tdt XML is '||DBMS_LOB.GETLENGTH(gettdtxml));
  dbms_output.put_line(DBMS_LOB.SUBSTR(gettdtxml, DBMS_LOB.GETLENGTH(gettdtxml), 1));
END;
/
SHOW ERRORS;

SQL> @get_tdtxml.sql
.
.
.
Length of tdt XML is 22884
<?xml version = '1.0' encoding = "UTF-8"?>
<TagDataTranslation version="0.04"
date="2005-04-18T16:05:00Z" xmlns:xsi="http://www.w3.org/2001/XMLSchema"
xmlns="oracle.mgd.idcode"><scheme name="SGTIN-64"
optionKey="companyprefixlength" xmlns="">
    <level type="BINARY"
prefixMatch="10" requiredFormattingParameters="filter">
      <option
optionKey="12" pattern="10([01]{3})([01]{14})([01]{20})([01]{25})" grammar="'10'
filter companyprefixindex itemref serial">
        <field seq="1"
decimalMinimum="0" decimalMaximum="7" characterSet="[01]*" bitLength="3"
length="1" padChar="0" padDir="LEFT" name="filter"/>
        <field seq="2"
decimalMinimum="0" decimalMaximum="16383" characterSet="[01]*" bitLength="14"
name="companyprefixindex"/>
        <field seq="3" decimalMinimum="0"
decimalMaximum="9" characterSet="[01]*" bitLength="20" length="1" padChar="0"
padDir="LEFT" name="itemref"/>
        <field seq="4" decimalMinimum="0"
decimalMaximum="33554431" characterSet="[01]*" bitLength="25" name="serial"/>
.
.
.
        <field seq="1" decimalMinimum="0" decimalMaximum="9999999" characterSet="[0-9]*"
length="7" padChar="0" padDir="LEFT" name="itemref"/>
        <field seq="2" decimalMinimum="0" decimalMaximum="999999" characterSet="[0-9]*" length="6"
padChar="0" padDir="LEFT" name="companyprefix"/>
      </option>
    </level>

</scheme></TagDataTranslation>
PL/SQL procedure successfully completed.
.
.
.
```

# GET_VALIDATOR Function

This function returns the Oracle Database tag data translation schema.

### Syntax

```
DBMS_MGD_ID_UTL.GET_VALIDATOR
 RETURN CLOB;
```

### Usage Notes

The return value contains the Oracle Database tag data translation schema.

### Examples

This example returns the Oracle Database TDT schema.

```
--Contents of get_validator.sql
DECLARE
  getvalidator              CLOB;
BEGIN
  getvalidator := DBMS_MGD_ID_UTL.get_validator;
  dbms_output.put_line('Length of validated oracle scheme xml is '||DBMS_LOB.GETLENGTH(getvalidator));
  dbms_output.put_line(DBMS_LOB.SUBSTR(getvalidator, DBMS_LOB.GETLENGTH(getvalidator), 1));
END;
/
SHOW ERRORS;

SQL> @get_validator.sql
.
.
.
Length of validated oracle scheme xml is 5780
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="oracle.mgd.idcode"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:tdt="oracle.mgd.idcode" elementFormDefault="unqualified"

attributeFormDefault="unqualified" version="1.0">
  <xsd:annotation>

<xsd:documentation>
      <![CDATA[
<epcglobal:copyright>Copyright ?2004
Epcglobal Inc., All
Rights
Reserved.</epcglobal:copyright>
<epcglobal:disclaimer>EPCglobal Inc., its
members, officers, directors,
employees, or agents shall not be liable for any
injury, loss, damages,
financial or otherwise, arising from, related to, or
caused by the use of this
document. The use of said document shall constitute
your express consent to
the foregoing
exculpation.</epcglobal:disclaimer>
<epcglobal:specification>Tag Data
Translation (TDT) version
```

```
1.0</epcglobal:specification>
]>

</xsd:documentation>
  </xsd:annotation>
  <xsd:simpleType
name="LevelTypeList">
    <xsd:restriction base="xsd:string">

</xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="TagLengthList"

<xsd:restriction base="xsd:string">
    </xsd:restriction>
  </xsd:simpleType>

<xsd:simpleType name="SchemeNameList">
    <xsd:restriction base="xsd:string">

</xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType
name="InputFormatList">
    <xsd:restriction base="xsd:string">

<xsd:enumeration value="BINARY"/>
      <xsd:enumeration value="STRING"/>

</xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="ModeList">

<xsd:restriction base="xsd:string">
      <xsd:enumeration value="EXTRACT"/>

<xsd:enumeration value="FORMAT"/>
    </xsd:restriction>
  </xsd:simpleType>

<xsd:simpleType name="CompactionMethodList">
    <xsd:restriction
base="xsd:string">
      <xsd:enumeration value="32-bit"/>
<xsd:enumeration value="16-bit"/>
      <xsd:enumeration value="8-bit"/>

<xsd:enumeration value="7-bit"/>      <xsd:enumeration value="6-bit"/>

<xsd:enumeration value="5-bit"/>
    </xsd:restriction>
  </xsd:simpleType>

<xsd:simpleType name="PadDirectionList">
    <xsd:restriction
base="xsd:string">
      <xsd:enumeration value="LEFT"/>
      <xsd:enumeration
value="RIGHT"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType
```

```
name="Field">
    <xsd:attribute name="seq" type="xsd:integer" use="required"/>

<xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute
name="bitLength" type="xsd:integer"/>
    <xsd:attribute name="characterSet"
type="xsd:string" use="required"/>
    <xsd:attribute name="compaction"
type="tdt:CompactionMethodList"/>
    <xsd:attribute name="compression"
type="xsd:string"/>
    <xsd:attribute name="padChar" type="xsd:string"/>

<xsd:attribute name="padDir" type="tdt:PadDirectionList"/>
    <xsd:attribute
name="decimalMinimum" type="xsd:long"/>
    <xsd:attribute name="decimalMaximum"
type="xsd:long"/>
    <xsd:attribute name="length" type="xsd:integer"/>

</xsd:complexType>
  <xsd:complexType name="Option">
    <xsd:sequence>

<xsd:element name="field" type="tdt:Field" maxOccurs="unbounded"/>

</xsd:sequence>
    <xsd:attribute name="optionKey" type="xsd:string"
use="required"/>
    <xsd:attribute name="pattern" type="xsd:string"/>

<xsd:attribute name="grammar" type="xsd:string" use="required"/>

</xsd:complexType>
  <xsd:complexType name="Rule">
    <xsd:attribute
name="type" type="tdt:ModeList" use="required"/>
    <xsd:attribute
name="inputFormat" type="tdt:InputFormatList"

use="required"/>
    <xsd:attribute name="seq" type="xsd:integer"
use="required"/>
    <xsd:attribute name="newFieldName" type="xsd:string"
use="required"/>
    <xsd:attribute name="characterSet" type="xsd:string"
use="required"/>
    <xsd:attribute name="padChar" type="xsd:string"/>

<xsd:attribute name="padDir" type="tdt:PadDirectionList"/>
    <xsd:attribute
name="decimalMinimum" type="xsd:long"/>
    <xsd:attribute name="decimalMaximum"
type="xsd:long"/>
    <xsd:attribute name="length" type="xsd:string"/>

<xsd:attribute name="function" type="xsd:string" use="required"/>

<xsd:attribute name="tableURI" type="xsd:string"/>
    <xsd:attribute
name="tableParams" type="xsd:string"/>
```

```
    <xsd:attribute name="tableXPath"
type="xsd:string"/>
    <xsd:attribute name="tableSQL" type="xsd:string"/>

</xsd:complexType>
  <xsd:complexType name="Level">
    <xsd:sequence>
<xsd:element name="option" type="tdt:Option" minOccurs="1"

maxOccurs="unbounded"/>
      <xsd:element name="rule" type="tdt:Rule"
minOccurs="0"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
<xsd:attribute name="type" type="tdt:LevelTypeList" use="required"/>
<xsd:attribute name="prefixMatch" type="xsd:string" use="optional"/>
<xsd:attribute name="requiredParsingParameters" type="xsd:string"/>
<xsd:attribute name="requiredFormattingParameters" type="xsd:string"/>

</xsd:complexType>
  <xsd:complexType name="Scheme">
    <xsd:sequence>
<xsd:element name="level" type="tdt:Level" minOccurs="1" maxOccurs="5"/>

</xsd:sequence>
    <xsd:attribute name="name" type="tdt:SchemeNameList"
use="required"/>
    <xsd:attribute name="optionKey" type="xsd:string"
use="required"/>
    <xsd:attribute name="tagLength" type="tdt:TagLengthList"
use="optional"/>
  </xsd:complexType>
  <xsd:complexType
name="TagDataTranslation">
    <xsd:sequence>
      <xsd:element name="scheme"
type="tdt:Scheme" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute
name="version" type="xsd:string" use="required"/>
    <xsd:attribute name="date"
type="xsd:dateTime" use="required"/>
  </xsd:complexType>
  <xsd:element
name="TagDataTranslation" type="tdt:TagDataTranslation"/>
</xsd:schema>

PL/SQL procedure successfully completed.
.
.
.
```

# REFRESH_CATEGORY Function

This function refreshes the metadata information on the Java stack for the specified category.

This function must be called before using `MGD_ID` functions.

### Syntax

```
DBMS_MGD_ID_UTL.REFRESH_CATEGORY (
   category_id  IN VARCHAR2);
```

### Parameters

**Table 130-13 REFRESH_CATEGORY Function Parameters**

| Parameter | Description |
|---|---|
| category_id | Category ID |

### Examples

The following example refreshes the metadata information for the EPC category ID.

```
--Contents of tostring3.sql
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));
 dbms_output.put_line('..Testing to_string');
 DBMS_OUTPUT.PUT_LINE('test 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();

SQL> @tostring3.sql
..Testing to_string
test 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.
```

# REMOVE_CATEGORY Procedure

This procedure removes a category including all the related TDT XML.

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

### Syntax

Removes a category based on the specified category ID.

```
DBMS_MGD_ID_UTL.REMOVE_CATEGORY (
   category_id  IN VARCHAR2);
```

Removes a category based on the specified category name and category version.

```
DBMS_MGD_ID_UTL.REMOVE_CATEGORY (
   category_name    IN VARCHAR2,
   category_version IN VARCHAR2);
```

### Parameters

**Table 130-14 REMOVE_CATEGORY Procedure Parameters**

| Parameter | Description |
|---|---|
| `category_id` | Category ID |
| `category_name` | Name of category |
| `category_version` | Category version |

### Usage Notes

If the value of `category_version` is NULL, all versions for the specified category will be removed.

### Examples

See the ADD_SCHEME Procedure for an example of removing a category.

## REMOVE_PROXY Procedure

This procedure unsets the host and port of the proxy server.

### Syntax

```
DBMS_MGD_ID_UTL.REMOVE_PROXY;
```

### Examples

See the REFRESH_CATEGORY Function for an example.

## REMOVE_SCHEME Procedure

This procedure removes a tag data translation scheme from a category.

### Syntax

```
DBMS_MGD_ID_UTL.REMOVE_SCHEME (
   category_id  IN  VARCHAR2,
   scheme_name  IN  VARCHAR2);
```

### Parameters

**Table 130-15 REMOVE_SCHEME Procedure Parameters**

| Parameter | Description |
|---|---|
| `category_id` | Category ID |
| `scheme_name` | Name of scheme |

### Examples

See the ADD_SCHEME Procedure for an example of removing a scheme.

# SET_JAVA_LOGGING_LEVEL Procedure

This procedure sets the Java trace logging level.

### Syntax

```
DBMS_MGD_ID_UTL.SET_JAVA_LOGGING_LEVEL (
   logginglevel IN  INTEGER);
```

### Parameters

**Table 130-16 SET_JAVA_LOGGING_LEVEL Procedure Parameters**

| Parameter | Description |
|---|---|
| `logginglevel` | Logging level. The Java logging level can be one of the following values in descending order:<br>• `LOGGING_LEVEL_OFF CONSTANT INTEGER := 0`<br>• `LOGGING_LEVEL_SEVERE CONSTANT INTEGER := 1`<br>• `LOGGING_LEVEL_WARNING CONSTANT INTEGER := 2`<br>• `LOGGING_LEVEL_INFO CONSTANT INTEGER := 3`<br>• `LOGGING_LEVEL_FINE CONSTANT INTEGER := 4`<br>• `LOGGING_LEVEL_FINER CONSTANT INTEGER := 5`<br>• `LOGGING_LEVEL_FINEST CONSTANT INTEGER := 6`<br>• `LOGGING_LEVEL_ALL CONSTANT INTEGER := 7` |

### Examples

See the GET_JAVA_LOGGING_LEVEL Function for an example.

# SET_PLSQL_LOGGING_LEVEL Procedure

This procedure sets the PL/SQL trace logging level.

### Syntax

```
DBMS_MGD_ID_UTL.SET_PLSQL_LOGGING_LEVEL (
   level IN INTEGER);

PRAGMA restrict_references(set_plsql_logging_level, WNDS);
```

### Parameters

**Table 130-17 SET_PLSQL_LOGGING_LEVEL Procedure Parameters**

| Parameter | Description |
|---|---|
| level | Logging level. The PL/SQL logging level can be one of the following values in descending order:<br>• LOGGING_LEVEL_OFF CONSTANT INTEGER := 0<br>• LOGGING_LEVEL_SEVERE CONSTANT INTEGER := 1<br>• LOGGING_LEVEL_WARNING CONSTANT INTEGER := 2<br>• LOGGING_LEVEL_INFO CONSTANT INTEGER := 3<br>• LOGGING_LEVEL_FINE CONSTANT INTEGER := 4<br>• LOGGING_LEVEL_FINER CONSTANT INTEGER := 5<br>• LOGGING_LEVEL_FINEST CONSTANT INTEGER := 6<br>• LOGGING_LEVEL_ALL CONSTANT INTEGER := 7 |

### Examples

See the GET_PLSQL_LOGGING_LEVEL Function for an example.

## SET_PROXY Procedure

This procedure sets the host and port of the proxy server for Internet access.

This procedure must be called if the database server accesses the Internet using a proxy server. Internet access is necessary because some rules need to look up the Object Naming Service (ONS) table to get the company prefix index.

You do not need to call this procedure does if you are only using schemes that do not contain any rules requiring Internet access.

### Syntax

```
DBMS_MGD_ID_UTL.SET_PROXY (
   proxt_host   IN  VARCHAR2,
   proxy_port   IN  VARCHAR2);
```

### Parameters

**Table 130-18 SET_PROXY Procedure Parameters**

| Parameter | Description |
|---|---|
| proxy_host | Name of host |
| proxy_port | Host port number |

### Examples

See the REFRESH_CATEGORY Function for an example.

# VALIDATE_SCHEME Function

This function validates the input tag data translation XML against the Oracle Database tag data translation schema.

### Syntax

```
DBMS_MGD_ID_UTL.VALIDATE_SCHEME (
   xml_scheme IN CLOB)
 RETURN VARCHAR2;
```

### Parameters

**Table 130-19 VALIDATE_SCHEME Function Parameters**

| Parameter | Description |
|---|---|
| xml_scheme | Scheme to be validated. |

### Usage Notes

The return value contains the components names for the specified scheme.

### Examples

See the ADD_SCHEME Procedure or the EPC_TO_ORACLE_SCHEME Function for an example.