

Classifying Ministry Data Using Corporate Code Tables - Guide S21

Purpose of Document

This guide explains the standards and conventions both for classifying information using corporate codes, and for the use of standard corporate code tables and code lookups that all applications must follow at the Ministry of Forests and Range.

The guide is intended for ministry business area data custodians, business area staff, systems analysts and systems designers who need to establish classification schemes to achieve standardization and consistency within ministry business information. The *Standard Tables for Code Implementation* portion of the guide may be most applicable to those having some knowledge of the techniques and procedures involved in systems design and programming (see other [Guides](#)).

The source tables holding ministry corporate codes reside on the ministry's Oracle platform. These codes are copied to Oracle application code tables on a regular basis so all ministry applications use codes with consistent values and meaning. This consistency is always important, but is especially important where data is shared between different ministry business areas.

If you have any questions, see the staff in the Data Administration group. All logical and physical names must be approved by Data Administration staff prior to any tables being built. **This includes names for codes which are formally approved at the data model review.**

For an understanding of the importance of ministry corporate codes and the requirements for code usage, see [Data Administration Standards \(Guide S38\)](#).

See also Ministry of Forests and Range System Development Guides:

[S7: Data Modeling Guide for Relational Applications](#)

[S19: Data Naming Standards for Systems Development](#)

Document Status

Although this is a "final" document it is updated periodically, so suggestions for improvement and/or clarification are welcome. Contact any member of the [Data Administration group](#) if you have any comments.

[Back to top](#)

Classifying Ministry Information

This section explains why we classify our data, what we have done in the past, some problems with historical data classifications and how to avoid these problems in the future.

Importance of Information Classification

To get the best value out of the information collected by ministry staff it is often necessary to classify this information. Classifying information, by applying a formal structure to its collection and use, yields many benefits:

- Consistent data capture
- Consistent meaning
- Data Sharing potential
- Data Analysis potential

Where consistent meaning of possible data values is necessary for data sharing, data analysis and data reporting, it is necessary to codify data. If data is left as free-form text it is not possible to consistently capture it, globally understand it, share it or consistently analyse it.

For example, imagine if there were no standard classification scheme for tree species and it was captured as free form text, this is what would happen when you:

- attempt to consistently capture all Douglas Fir stands - they will be captured as F, FIR, Douglas Fir, DF, Fir - Douglas, FD, D. Fir, etc - as many ways as you can think of
- attempt to get consistent understanding that what is being captured is a species of tree - tree species, brush, shrubs, etc will be captured
- attempt to share data consistently with other business areas - they won't always know what DF is, or what kind Fir is and all business areas will capture Douglas Fir differently
- attempt to consistently analyse and report on data - what percentage of the tree stands are Douglas Fir? what would you count?

If data values are defined specifically and the same values are captured and used consistently throughout the ministry (e.g. code **FD** always means Douglas Fir) comprehensive understanding, data sharing, data analysis, data reporting, etc. are possible.

Historical Patterns for Creating Classification Codes

Historically, codes have been often been created to satisfy the needs of a specific business area or a specific business function within the ministry. For example:

- A ministry business area needed to know what species of tree was being harvested so they created a tree species code and values for harvesting. Another business area needed to know what kind of tree species was being

used for re-planting so they created a tree species code and values for silviculture.

- A business area needed to know what types of pests were damaging trees so they created a pest species code and values.

Historically codes have also been created to embed as much meaning in as little space as possible. This is likely the result of having limited space on field forms, punch cards and disks in the past and also, in many cases, likely following the pattern of what has been done in the past.

Problems with Historical Classification Codes

Historical methods of classifying ministry data have caused the ministry some serious problems. Having different codes for the same thing has caused great confusion. For example, a field worker might be determining tree species for harvesting and capture Douglas Fir as FD, and another time be determining tree species for re-planting and have to capture Douglas Fir as DF. There is a pretty good chance the field worker won't notice the code difference and the tree species captured for re-planting will be FD. This kind of error is due to the inconsistency in codes between the two business areas. It gets real complicated when FD in the re-planting tree species means Flowering Dogwood.

Another historical problem with classification of data is embedding multiple, often hierarchical, meaning within one code. There are many examples of this kind of classification structure within the ministry, and just about everywhere else information has been classified. Embedding meaning can cause real problems. Often the embedded meaning has to be parsed out of the code, or worse we run out of values to continue the embedded meaning when new codes are added. Running out of values means we either add non-consistent values which are misleading, or we re-assign the values of the entire code causing (often massive and complex) data conversion, or we re-structure the code data expanding its field size to add levels to the hierarchy and cause expensive application maintenance.

Ways to Avoid Historical Problems

We still must live with many codes created in the past which cause these and other problems, or we must resolve the problems (often at great expense). There are some things we can do now and in the future to avoid or lessen code problems.

Identify specifically what it is you are dealing with and want to capture consistent, standard data values for. Ensure it reflects base data which the business area is responsible for. Create code values for the business data items using the guidelines below:

- Create meaningful codes where possible. A code value of 1 for Proposed and 2 for Completed is not very meaningful. Better to go with PRO for Proposed and COM for Completed; something that is intuitive.
- Make your code long enough to handle potential future values without making them meaningless. One or two character codes are often too restrictive. For example, you have a code that is two characters, one of the values is OG for Other Government. You want to add a code for Oil and Gas Commission. The

intuitive code of OG is not available. A three character code would allow a value of OGC for Oil and Gas Commission.

- Make your codes as simple as possible. Some coding schemes are so complex that you literally have to take a course to understand them. Simple codes with good, meaningful and easily understood descriptions will make your data much more useable and shareable.
- Ensure each code value is unique across all business areas that need to record and share data the code identifies. Most shared codes should be unique anyway as there is a business area, and a data custodian, responsible for the data the code identifies. This won't always be the case though, as in the past data custodian responsibilities were not identified and codes were created without much thought about sharing.
- If you need to group the codes you have identified, use a separate code to designate the grouping. Do not embed multiple meanings within codes. When multiple meanings (so called smart codes) are embedded, the codes created tend to be very narrow in their business usefulness, very inflexible when business changes or enhancements are necessary and very susceptible to the slightest business change causing expensive maintenance.

The ministry has a number of standard code tables in place for implementing classification codes.

[Back to top](#)

Standard Tables for Code Implementation

The MoFR Data Administration group maintains classification codes for different business areas throughout the ministry. When ministry business area staff need codes for use within their applications they must request these codes be added to the corporate code dictionary. Within MoFR, codes are managed corporately on behalf of a data custodian. Before adding new code values Data Administration staff determine if a code already exists and can be reused.

Application Code Tables

In the Oracle environment, individual 'application code tables' must be used for code validation. The source for all application code tables is the corporate code dictionary. Application developers are responsible for the design and creation (DDL submission) of Application Code tables. Individual application code tables are loaded from codes within the corporate code dictionary by data administration staff.

It is strongly recommended that projects use application code tables to enforce validations and build in referential integrity.

Use

In the Oracle environment, Application Code Tables must be used for code validation. Corporate code values from the source corporate code dictionary are used to populate and update the Application Code Table values on an as needed basis. The Data Administration section manages the update of Application Code Tables on behalf of Data Custodians.

Application Code Tables are based on a standard MoFR design. Application developers are responsible for creating the physical structure (DDL) of the Application Code Tables, using the standard design, through the ministry data modelling process.

It is strongly recommended that projects use Application Code Tables to enforce validations and support referential integrity.

Process

The business area defines appropriate codes matching the business needs. All requests for new codes are submitted to [Data Administration staff](#) through the appropriate [Data Standards Manager](#). See Appendix A for a sample format for proposing Application Code Table values.

Both the Data Standards Manager and Data Administration staff must review and approve all new code values. Once approved, Data Administration staff will manage the update process to populate Application Code Tables.

Detailed Design Standards for Application Code Table structure:

- The name of an Application Code Table should reflect the business content of the table. It is preferable to spell out abbreviated words.
- The table name must end with "_CODE".
- The Application Code Table name may be up to 30 characters.
- The first column of the Application Code Table is the SAME name as the Application Code Table. This column forms the primary key for the Application Code Table.
- When deciding on the length of the code primary key, the length should be that of the longest code value.
- The column order for the attributes must be as shown below.
- All columns must be mandatory.

Example

```
CREATE TABLE TIMBER_FILE_STATUS_CODE
( TIMBER_FILE_STATUS_CODE          VARCHAR2(5)          NOT NULL
, DESCRIPTION                      VARCHAR2(120)       NOT NULL
, EFFECTIVE_DATE                  DATE                NOT NULL
, EXPIRY_DATE                    DATE                NOT NULL
, UPDATE_TIMESTAMP                DATE                NOT NULL
) ;
```

The effective/expiry dates allow for historical codes to remain on the list for lookup but be restricted for data capture. The default for EFFECTIVE_DATE is the date the code is added and the default for EXPIRY_DATE is 9999-12-31. The structure is such that there can only be one valid date range for each code value (DESCRIPTION).

The UPDATE_TIMESTAMP is used by the Database Administration group in the physical update process that maintains the Application Code Tables.

Table Definition

Column Name	Column Number	Column Type	Length
XXXXXXXXXXXXXXXXXXXXXXXXX_CODE* ^[1]	1	VARCHAR2	varies (usually 1 to 3, but may be longer)
DESCRIPTION	2	VARCHAR2	120
EFFECTIVE_DATE	3	DATE	
EXPIRY_DATE	4	DATE	
UPDATE_TIMESTAMP	5	DATE	

(* indicates primary key)

[1] The column name MUST be the SAME AS the Application Code Table name

Standard Suffix: _CODE

[Back to top](#)

Cross Reference Tables

Cross Reference tables allows cross-referencing of codes. These are needed for situations where selection of a value from one code determines the values which are valid from a second code. For example, selection of a code for part-time temporary employment could guide the list of employment benefit codes available. This would result in a valid combination of two or more codes (concatenated keys). Concatenated keys (to code tables) cannot be supported, and should be implemented as a cross reference table.

As well, hierarchies of codes should be implemented in cross reference tables. For the most part, hierarchies can be split out. E.g. a two-level hierarchy, can be split out into two code lists, plus one cross reference table (XREF).

Projects maintain their own cross reference tables with the application's data tables. Insert scripts are to be submitted by the project for populating these cross reference tables.

Org Unit Table

Org_Unit table holds organizational unit codes and descriptions for all ministry offices and is used by all ministry applications whenever data entry validation or name lookup of ministry organizational units is needed.

Table Definition

Column Name	Table Name	Column Number	Column Type	Length
ORG_UNIT_NO ^[*]	ORG_UNIT	1	NUMBER	10
ORG_UNIT_CODE	ORG_UNIT	2	VARCHAR2	6
ORG_UNIT_NAME	ORG_UNIT	3	VARCHAR2	100
LOCATION_CODE	LOCATION ^[**]	4	VARCHAR2	3
AREA_CODE	ORG_UNIT	5	VARCHAR2	3
TELEPHONE_NO	ORG_UNIT	6	VARCHAR2	7
ORG_LEVEL_CODE	ORG_UNIT	7	VARCHAR2	1
OFFICE_NAME_CODE	ORG_UNIT	8	VARCHAR2	2
ROLLUP_REGION_NO	ORG_UNIT	9	NUMBER	10
ROLLUP_REGION_CODE	ORG_UNIT	10	VARCHAR2	6
ROLLUP_DIST_NO	ORG_UNIT	11	NUMBER	10
ROLLUP_DIST_CODE	ORG_UNIT	12	VARCHAR2	6
EFFECTIVE_DATE	ORG_UNIT	13	DATE	
EFFECTIVE_DATE	ORG_UNIT	14	DATE	
UPDATE_TIMESTAMP	ORG_UNIT	15	DATE	

(* indicates primary key)

(** indicates foreign key from related table)

Use

This table provides a lookup facility for ministry offices and addresses throughout the province. It has been designed to allow use of a fairly intuitive coding scheme for identifying ministry organizational units, and also to try to shield the ministry applications from changes within the organization. Shielding the ministry from organizational change comes from storing only the ORG_UNIT_NO. This way when there is a change in ORG_UNIT_CODE or ORG_UNIT_NAME it can be changed in only one place in ORG_UNIT_TABLE. To realize the benefits of this design:

- **only ORG_UNIT_NO is stored at the database level;** never ORG_UNIT_CODE
- **only ORG_UNIT_CODE and ORG_UNIT_NAME are presented to users;** never ORG_UNIT_NO
- ORG_UNIT_NO is a **meaningless number** (a "surrogate key") and **meaning is never attached to its values;** e.g., there are no significant ranges of numbers, or any other significance of ORG_UNIT_NO values that programmers can use to "take short cuts"
- ORG_UNIT_CODE is **never hard-coded in program code**

- take the effective and expiry dates into consideration: organizational units which are not yet effective would not normally be used; organizational units which are expired would normally be used for display purposes only

The reason that only org unit number is stored is to isolate ministry application from organizational changes. The advantage to this approach is that if our org unit code or name changes in the future for a particular organizational unit (which it OFTEN does as we change office names relatively frequently), we won't have to go back and convert all our historical data, we just have to change the code or name in the ORG_UNIT_TABLE. For example, in the summer of 1989 McBride Forest District [DMB] changed its name to Robson Valley Forest District [DRV]. No data conversion was required because of the ORG_UNIT_NO / ORG_UNIT_CODE lookup process.

ORG_LEVEL_CODE and OFFICE_NAME_CODE have been included to allow for database searching as these codes are also embedded in the ORG_UNIT_CODE.

ROLLUP attributes have been included to allow easy identification of the region a district reports to.

EFFECTIVE_DATE and EXPIRY_DATE have been added to allow organizational units which no longer exist, to be maintained for historical purposes.

Historical org unit codes are also maintained within org unit history and audit tables. For information on these tables, please contact [Data Administration staff](#).

[Back to top](#)

Appendix A - Application Code Table Spreadsheet

The following format can be used for submitting proposed Application Code table code values to DA for review. Three examples are provided:

1) In the first example, Appraisal_Category_Code, the project is proposing a brand new application code table, with new code values.

2) In the second example, Appraisal_Culvert_Type_Code, the project will create a new subset from an existing list of codes. If additional code values were needed, these could also be added as new code values to the master list (with Data Custodian approval) and to the new subset.

3) In the third Point_Of_Origin_Code, example, the project will use an existing code list and subset as is. In this case it is not necessary to list all of the code values.

Application Code Table (physical table name)	Code ID	Code Description	Code Source (provide existing code list source name or indicate new)	Code Subset Source (provide existing subset name or indicate new)
---	---------	------------------	---	--

APPRAISAL_CATEGORY_CODE	N	New	new	new
	R	Reappraisal		
	A			
APPRAISAL_CULVERT_TYPE_CODE	W	Wooden	CULVERT_TYPE_CODE	new
	M	Metal		
	T	Tabular		
POINT_OF_ORIGIN_CODE	AGAM	Agamemnon	BC_LOCATION_CODE	PT_OF_ORIGIN_ST
	ALAR	Alice Arm		
	(Continued)			

Where a new code list or new subset is required, DA will assign a new name.

[Back to top](#)

[Back to Data Administration Section](#)