# Database Version Control Module SOP

### Overview:

### There is a need to store the upgrade history for a given database instance so it can be safely and easily upgraded when a given version of an application or module is deployed. This Database Version Control Module (VCM) will serve to inform data staff of which database module version is installed on a given database instance and when each database module upgrade was applied to the instance. This module is used to apply the necessary database upgrades in the correct order to deploy a given database module version for an associated application/module.

### Resources:

* [VCM Documentation](DB%20Version%20Control%20Module%20Documentation.docx)
* [Template - Installing or Upgrading the Database](file:///C:\Users\Jesse.Abdul\Documents\Version%20Control\Git\centralized-tools\DB_version_control\docs\Template%20-%20Installing%20or%20Upgrading%20the%20Database.docx)
* Git Repository URL: [git@gitlab.pifsc.gov:centralized-data-tools/centralized-tools.git](mailto:git@gitlab.pifsc.gov:centralized-data-tools/centralized-tools.git) in the DB\_version\_control folder

### Method:

* (Defining DB Upgrades) All DDL and DML commands necessary to define the database structure and data (relevant reference data or application data but not any actual data that is being managed in the database) for a given database upgrade are saved as separate SQL files with standard headings.
* (DB Upgrade Logs) A simple modular database structure is implemented to track database upgrades over time in a given database instance so that all database upgrades applied are clearly identified. Each upgrade applied to a given database adds a record into the database upgrade table (DB\_UPGRADE\_LOGS) to indicate that the upgrade has been applied (the template for database upgrade log entries can be found in DB\_version\_control\_templates.txt under the "Database Upgrade Templates" section). Based on the current version of a given database instance and a desired version the necessary database upgrades can be easily applied in the correct order to upgrade the instance.
* (Defining DB Module Versions) Each database module version is defined as [MAJOR].[MINOR] where [MAJOR] is the major version of the database module and [MINOR] is the minor version of the database module, both the major and minor version values are integers starting from zero that are incremented by one without a leading zero. The particular major and minor versions are up to the discretion of the developer(s).
  + The exception to the rule is that the first version of a database module should be 0.1. Also, when each major version is incremented the minor version should be reset to 0 for that major version (e.g. version 3.0).
* (Determining DB Module Versions) Query the DB\_UPGRADE\_LOGS\_V View to review the upgrade history of all database modules on the given database instance to determine which database upgrade files need to be executed to upgrade a given database instance to a desired database module version.

### Installing the DB Version Control Module in a Given Code Repository:

* (Version Control Folder Structure) Implement a standard folder structure (shown in [Figure 1](#figure_1)) depending on the type of development project. There are two templates in the DB\_version\_control\docs\repository\_templates folder that will define the folder structure for the "database root" directory (either SQL or Shared\_SQL) and provide template/readme files.
  + The "SQL" directory is for projects that have a dedicated database model that is not shared between multiple applications/modules (e.g. DB Version Control). This SQL folder is stored in a given module's folder since it is module-specific.
    - Copy the SQL template folder into the appropriate application/module directory
  + The "Shared\_SQL" directory is for projects that have multiple applications/modules in one repository that use the same database module (e.g. PIFSC Data Set Database with bulk download module, URL verification module, and APEX data management application). This Shared\_SQL folder is stored at the root of the repository since it is shared between multiple applications/modules.
    - Copy the Shared\_SQL template folder into the project repository directory
  + Both "database root" directories contains a README.txt file that outlines the policies defined in this SOP.
  + (Code Commit and DB Upgrade Templates) The SQL/shared\_SQL "database root" directory contains a DB\_version\_control\_templates.txt file that provides some standard templates for code commit messages and database upgrade file comments. These can be modified to be project-specific so they can be easily copied and pasted into commit messages and database module upgrade files. Each line in the template file that starts with "\*\*" is informational and will not be actually added to the commit message or database upgrade files. The first section (Code Commit Templates) contains templates for code commits and the second section (Database Upgrade Templates) contains templates for database upgrade files. Placeholders (indicated by brackets) will be replaced with actual content as indicated in the informational content.
    - \*\*Note: For ease of use the database module upgrade template content has been included in the DDL/DML template files.
* (DB Upgrade Files) The "upgrades" folder contains each individual database module upgrade file that is necessary to apply a given database module upgrade. The naming convention is [DATABASE NAME]\_DDL\_DML\_upgrade\_v[MAJOR].[MINOR].sql where [DATABASE NAME] is the name of the database module with underscores instead of spaces. For example PIFSC\_data\_set\_DB\_DDL\_DML\_upgrade\_v0.5.sql) is the fifth minor version of the PIFSC Data Set DB module.
  + Requirements: Each database module upgrade must be able to be executed on a blank schema in order (major/minor version) from the lowest to highest version to deploy a given version of the database module. Executing a given [DATABASE NAME]\_DDL\_DML\_upgrade\_v[MAJOR].[MINOR].sql version on a database module of the previous version will upgrade the database module version to the given [MAJOR].[MINOR] version (e.g. executing [DATABASE NAME]\_DDL\_DML\_upgrade\_v1.14.sql will upgrade an instance of the v1.13 database module to v1.14). Each upgrade file will contain a SQL statement to define and describe the given database module upgrade in the Database Version Control Module. The template for the headings used in the upgrade files can be found in DB\_version\_control\_templates.txt under the "Database Upgrade Templates" section.
* (Combined DB Definition File)The "database root" directory contains a combined DDL/DML file ([DATABASE NAME]\_combined\_DDL\_DML.sql) that will deploy the given database module version on a blank database instance.
  + The combined DDL/DML file will be updated each time a new database module upgrade has been developed to add a reference to the new database upgrade file so that the combined file will run each database upgrade file in the proper sequence to generate the necessary objects and reference/application data for a new database module deployment of the given version. The template for the headings used in the combined file can be found in DB\_version\_control\_templates.txt under the "Database Upgrade Templates" section.
* (Standard DB Installation/Upgrade Documentation) The "Installing or Upgrading the Database Documentation.docx" is a standard document that provides information for installing/upgrading a given database module that should be linked from the given application's/module's main technical document so the database module upgrade process is clearly defined.
  + When installing the DB version control module the "[Template - Installing or Upgrading the Database Documentation.docx](Template%20-%20Installing%20or%20Upgrading%20the%20Database%20Documentation.docx)" should be copied to a given module's code repository's directory structure (and renamed) and all placeholders should be replaced with their actual values based on the guidance in the document comments.

### DB Version Control Module Procedure:

* (Project Database Upgrades) Copy and rename the [DATABASE NAME]\_DDL\_DML\_upgrade\_v0.1.sql with the appropriate values for the template placeholders to use as the current project database upgrade file.
  + Update the content of the current database upgrade file to replace all template placeholders with the appropriate values for the database upgrade (excluding the "INSERT INTO DB\_UPGRADE\_LOGS" SQL statement, which will be done at the end of the process).
  + Enter all DDL/DML commands necessary to upgrade the database in the current database upgrade file.
    - If you are implementing the Database Version Control Module on an existing database for the first time then copy all of the DDL/DML commands necessary to create the database (excluding the data managed in the database) on a blank database into the first DDL/DML upgrade file:
      * Install the DB Version Control Module at the top of the first DDL/DML upgrade file as normal.
      * Export the database DDL/DML for the existing database and copy it after the DB Version Control Module installation script in the first DDL/DML upgrade file.
      * This resultant database upgrade file will serve to create the database on a blank database schema.
        + In order to actually upgrade the existing database schema execute the DB version control installation script and also run the upgrade file DML for the DB\_UPGRADE\_LOG entry to indicate that the initial DDL/DML upgrade was applied to the existing database schema. The SOP for testing database upgrades should be followed as normal.
  + (External Module Installations/Upgrades) When installing a new external database module or upgrading an existing external database module (e.g. DB version control module) on an existing project's database (e.g. PIFSC Data Set Database) add the external database module upgrade code to the current project database upgrade file, you can also optionally add the individual external database module upgrade code files to the repository (e.g. SQL\upgrades\external\_modules) and reference them using a relative path ("@@" syntax) in the current project database upgrade file.
    - Add a comment in the project database upgrade file to indicate that the given external database module is being installed/upgraded directly above the actual external database module upgrade code. Indent the external database module's DDL/DML code so it is apparent that all indented code following the comment is part of the external database module installation/upgrade.
    - If the external database module is being upgraded then copy/reference all of the code in the given external database module's [DATABASE NAME]\_DDL\_DML\_upgrade\_v[MAJOR].[MINOR].sql file(s) necessary to make the upgrade into the current project database upgrade file.
    - If the external database module is being installed for the first time then copy/reference all of the code in the given external database module's [DATABASE NAME]\_DDL\_DML\_upgrade\_v[MAJOR].[MINOR].sql file(s) necessary to install the specific version of the database module (e.g. v 0.7) into the current project database upgrade file.
    - If you are installing/upgrading multiple external modules refer to the module documentation to determine if there are any dependencies that require a given module to be installed/upgraded before another one (e.g. DB Version Control Module must be installed before the Authorization Application Module).
      * Copy/Reference all of the code in the corresponding external database modules' [DATABASE NAME]\_DDL\_DML\_upgrade\_v[MAJOR].[MINOR].sql file(s) into the project database upgrade file in the appropriate order based on the module dependencies.
      * \*\*Note: To streamline new installations of multiple database modules you can use the DB Module Packager (DMP) project ([git@gitlab.pifsc.gov:centralized-data-tools/db-module-packager.git](mailto:git@gitlab.pifsc.gov:centralized-data-tools/db-module-packager.git)) to install multiple modules using the DMP files.
  + Update the current project database upgrade file to change the template placeholder values in the "INSERT INTO DB\_UPGRADE\_LOGS" SQL statement to appropriate values for the given database upgrade. It is important to define a descriptive value for [UPGRADE DESCRIPTION] so it is clear what was done in a given database module upgrade.
  + (Combined DB Definition File) Add a reference to the new database upgrade file to the bottom of the project database's [DATABASE NAME]\_combined\_DDL\_DML.sql file.
* (Testing Database Upgrades) Test that the current database upgrade works as expected (The approach listed below requires a dedicated comparison database schema)
  + \*\*Note: the [SQL scripts/drop\_all\_objects.sql](../../SQL%20scripts/drop_all_objects.sql) script can be used to drop all objects in a given schema and can be used for initializing the comparison database but be very careful when using this script to ensure it is not used on the development, test, or production databases or work/data may be lost.
  + In the blank comparison database schema execute the combined DDL/DML script from the previous database module version and then execute the pending database upgrade file. Use a database diff tool to confirm that the current development database and the comparison database have equivalent objects.
    - Confirm there are no database errors that result from executing the DDL/DML files.
  + In the blank comparison database schema execute the updated combined DDL/DML script from the pending database module upgrade. Use a database diff tool to confirm that the current development database and the comparison database have equivalent objects.
    - Confirm there are no database errors that result from executing the combined DDL/DML file.
* (Version Control Code Commits) When application/database code is committed to a version control system clearly document each version of the database module and associated application(s) to clearly identify which database module version is required for a given application/module version. The template for code commits can be found in DB\_version\_control\_templates.txt under the "Code Commit Templates" section.
  + \*\*Note: Not all application code updates will require a corresponding database upgrade and vice versa.
  + (Git only) Tag the revision with a standard DB module version tag to indicate that the database module was upgraded in the given code commit according to the version control template
    - (Git only) Define a tag message for the DB module upgrade according to the version control template.
  + (Git only) When installing/upgrading an external DB module tag the revision with a standard DB module version tag to indicate the version of each of the external DB modules that were installed or upgraded to in the given code commit according to the version control template.
    - (Git only) Define a tag message for the highest version each external DB module that was installed/upgraded according to the version control template. For example if versions 0.1, 0.2, and 0.3 of a given external module are installed then tag the repository with only the version 0.3 tag

### Figure 1: Standard DB Version Control Module folder structure

