DacPac Data/Code Standard

# Objective:

This document is designed to provide information for adding/changing data using the DacPac. There are also some coding standards included. It is assumed that you are using Visual Studio and have the Git repository already configured.

## File groups\Files

Will not be deployed via Dacpacs.  The following needs to be done **before** the dacpac deployment.

**Developer** needs to:

* Add new File\Filegroups to the solution
* DSO tracker submitted to add to PPD stacks
* PSO tracker submitted to add to prod (many times this can be done at the same time as PPD, and does not need to wait on release to prod)

**PSO**(If PSO is adding files to address large file sizes or production issues in prod):

* Create PSO tracker for the files they added
* Create a DSO tracker to get files added to sln
* Also request the filegroup\files get added to the lower stacks

Partitions

* We purposefully set the publish profiles to ignore partitions.  If you do not follow the steps below, the partition fn\scheme will not get added.
* If adding a **new** partitioned table:
* -Add to sln
* -**Create a post deploy script** to add partitions FN\Scheme to newly created table
* Adding to **existing** table:
* -Add to sln
* -Create DSO tracker and work with DSO on time and approach to implement new partition on table with data.
* -Create PSO tracker and work with PSO on time and approach to implement new partition on table with data.

## Object Script

* There are tasks that are not allowed in the DacPac we handle these in the pre-deployment
  + Dropping of objects will be renamed instead
    - Update the source control with the new name
    - Create a pre-deployment script renaming the object
      * Use the prefix ToBeDelete\_objectname
  + There will be a job that DSO/PSO will use to determine which objects get dropped

## Data Script

* Scripts need to be re-runnable
  + USE IF CHECKS
  + MERGE STATEMENTS
  + SENSIBLE WHERE CLAUSES
* Data scripts must be saved in a folder under the \Scripts root folder
  + There will be folders for PermanentScripts(scripts that must run regardless of release), ProductionDefect, Releases
    - If you do not see a release folder that matches the work needed
      * create the ReleaseFolder following the tag in Rational
    - Ex. \Release\_2020\_Clover\Myscript.sql
    - If you need to organize the scripts in a folder it MUST be saved in one of the constant folders (PermanentScripts, ProductionDefect, Releases)
      * Ex. \Release\_2020\_Clover\Cart\Myscript.sql
  + The path in the post deployment file must include the exact path
  + The name of the script will be removed from the PreDeployment and PostDeployment scripts after release to production
* Name the script the with the action followed by the table name that is being changed
  + Example: DELETE\_d\_land\_use\_modifier.sql – multiple devs working on same table use comments and sections in one file
    - Action Names
      * DELETE
      * INSERT (use this for MERGE statements as well)
      * UPDATE
      * If any other action is needed use table name only
    - Grouping of related tables in one script is okay
    - Name should reflect that it is grouping multiple tables (ex. Group\_ as a prefix then name most parent table)
  + Use the names to check for duplicate scripts
* You can use one driver script that calls multiple scripts
* All scripts MUST have a terminating ‘GO’ statement
* All scripts MUST include a ‘Print’ statement that prints the name of the script in the postdeployment.sql and preDeployment.sql
  + Print ‘cart.d\_answer\_type.sql’
  + :r 01\_cart.d\_answer\_type.sql
* Do NOT use ‘BEGIN TRANSACTION’ and ‘COMMIT’
  + The DacPac deploys in a transaction already
  + Use Try Catch instead if you need to track commits based on error
* Do NOT write ‘USE’ statements
  + The DacPac deploys using the project and know what database to deploy
* Use 3 part names only if needed
  + Should be used sparingly
  + We should try to find another solution to avoid coupling the databases
* Use server checks for environment specific data
  + Do NOT use the \_stack variable
    - The stack variable does not always get set
    - Code has a chance to not run at all
  + SELECT @@SERVERNAME
  + Use this list for the server names
    - NRCtBoxicpCID3A\NRCtBoxicpCID3A
    - NRCdBoxicpSAN3A\NRCdBoxicpSAN3A (SAN-INT)
    - NRCdBoxiMGSAN3A\NRCdBoxiMGSAN3A (SAN-TEST2)
    - NRCdBoxiSCSAN3A\NRCdBoxiSCSAN3A (SAN-UAT/LOAD)
    - NRCtBoxiCpInt1A\NRCtBoxiCpInt1A
    - NRCtBoxiCpQat2A\NRCtBoxiCpQat2A
    - NRCPBoxixxuat1c\NRCPBoxixxuat1c
    - NRCtBoxiCpLod1A\NRCtBoxiCpLod1A
    - NRCtBoxiCpQat1A\NRCtBoxiCpQat1A
    - NRCPCP\NRCPCP
  + Example:

IF @@SERVERNAME IN (‘NRCtBoxiCpQat1A\NRCtBoxiCpQat1A’, ‘NRCtBoxiCpQat2A\NRCtBoxiCpQat2A’)

BEGIN

--Insert Code here

END

## Merge Statement

* Do NOT use DELETE IF NOT IN source statements
  + Write a separate DELETE script for data that needs to be removed
* Do NOT disable CONSTRAINTS
  + The developer will be required to find all dependencies
  + Re-enabling CONSTRAINTS can cause issues

## Domain Data

* The DacPac will not manage domain data
  + The source of record will be production
* Scripts will not include all data in existing tables
  + One exception is new tables not in prod will have more/all data
  + Should only be the data that needs to be added or deleted

## Example

This is a script that follows the standards

PRINT N'd\_comparison\_operator.sql'

:r d\_comparison\_operator.sql

Actual script:

IF @@SERVERNAME IN

('NRCtBoxiCpLod1A\NRCtBoxiCpLod1A','NRCdBoxicpSAN3A\NRCdBoxicpSAN3A') –-on an as needed basis

BEGIN

DECLARE @d\_comparison\_operator AS TABLE

(

[comparison\_operator\_id] [int] NOT NULL,

[comparison\_operator\_name] [varchar](50) NOT NULL

)

INSERT @d\_comparison\_operator ([comparison\_operator\_id], [comparison\_operator\_name]) VALUES (1, N'Equal =')

INSERT @d\_comparison\_operator ([comparison\_operator\_id], [comparison\_operator\_name]) VALUES (2, N'Not Equal !=')

INSERT @d\_comparison\_operator ([comparison\_operator\_id], [comparison\_operator\_name]) VALUES (3, N'Less Than <')

INSERT @d\_comparison\_operator ([comparison\_operator\_id], [comparison\_operator\_name]) VALUES (4, N'Less Than or Equal To <=')

INSERT @d\_comparison\_operator ([comparison\_operator\_id], [comparison\_operator\_name]) VALUES (5, N'Greater Than >')

INSERT @d\_comparison\_operator ([comparison\_operator\_id], [comparison\_operator\_name]) VALUES (6, N'Greater Than or Equal To >=')

INSERT @d\_comparison\_operator ([comparison\_operator\_id], [comparison\_operator\_name]) VALUES (7, N'Range X <= val <= Y')

MERGE INTO [cart].[d\_comparison\_operator] AS TGT

USING @d\_comparison\_operator AS SRC

ON TGT.[comparison\_operator\_id] = SRC.[comparison\_operator\_id]

WHEN MATCHED THEN

UPDATE SET

TGT.[comparison\_operator\_name] = SRC.[comparison\_operator\_name]

WHEN NOT MATCHED THEN

INSERT(

[comparison\_operator\_id],

[comparison\_operator\_name])

VALUES (

SRC.[comparison\_operator\_id],

SRC.[comparison\_operator\_name]);

END

GO