Automated Build Process

Contents

[Versions 2](#_Toc63249516)

[Command Line Arguments 4](#_Toc63249517)

[Example of use 5](#_Toc63249518)

[What is Happening When You Execute Clone.bat 5](#_Toc63249519)

[When executed the process will first validate the required command line parameters. 5](#_Toc63249520)

[This will clones SDS (Regarding REPO versions see Restructure of SDS REPO). When the process runs it will download the SDS REPO automatically using SSH. 5](#_Toc63249521)

# Versions

|  |  |  |
| --- | --- | --- |
| Version | Author | Comment |
| 1.0 | Craig Allum | Initial Version Of Document |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

This document explains the Automated Build Process that links into the Restructured SDS REPO and pulls a release and then can apply the updates on that release to a known schema or create a new schema and install all the objects that make up that release.

This document should be read in hand with Restructure of SDS REPO.doc which is located in the documentation REPO.

# Command Line Arguments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NAME | Description | Usage | Parameter Order | Required |
| SysUsrName | SYS User Name For System | SYS Username | 1 | Y |
| SysPassword | SYS Password For System | SYS Password | 2 | Y |
| ServiceName | Target Database | TNS Service Name e.g. DEV19 | 3 | Y |
| NewSchemaUsrName | New / Existing Schema Name | When NewOrUpgrade is N then provide the new schema name.  When NewOrUpgrade is U then provide the schema name which we want to upgrade.  SDSMAIN and SDS\_MAIN are blocked from being provided. | 4 | Y |
| NewSchemaPassword | New/Exisiting Schema Password | Password for the new schema or the existing schema | 5 | Y |
| BuildConfiguration | BuildConfiguration Not In Use | Just Enter a Value as it is not in use currently, but will be used as a way of defining what build data to introduce as part of the build. | 6 | Y |
| NewOrUpgrade | N/U - (N)New Or (U)Upgrade | (N)ew will create a New Schema.  (U)pdate will process update an existing schema | 7 | Y |
| Clearup | Y/N – (Y) Yes Or (N) No | (Y) will destroy the build folder on the local machine. (N) will not destroy the build folder. | 8 | Y |
| SchemaToDrop | Schema To Drop | Optional parameter, provided when we need to drop a schema, SDSMAIN and SDS\_MAIN are blocked from being created or dropped. | 9 | N |
| GitHash | This is The Version Number of the Release, not the CommitHash in Git | If not provided it will take the latest changes to the Master Branch. If a Version is provided e.g. v1.0.1 it will take that version of the Master branch | 10 | N |

# Example of use

D:\Users\Craig.Allum\Documents\BuildScripts\Clone.bat SYS syspassword DEV19 TESTSCHEMA testschemapassword build1 N TESTSCHEMA

# Under The Bonnet

# When executed the process will first validate the required command line parameters, refer to the command line arguments for the required items.

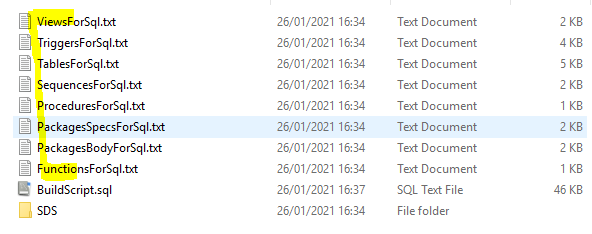
# If all required items are provided then we have fork the process.

## NEW

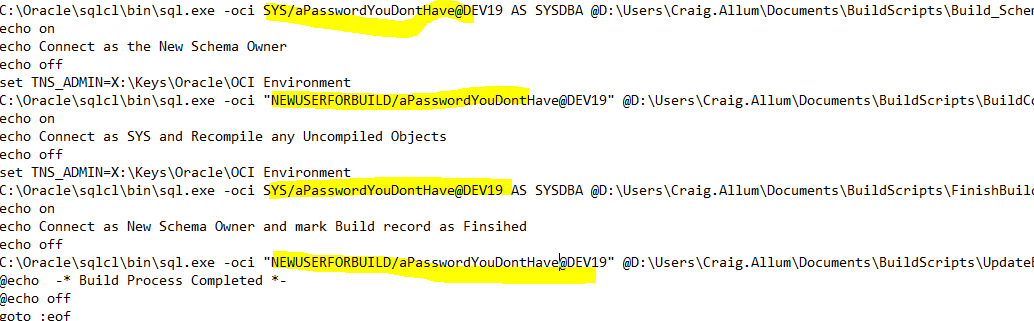
# The process will clone the SDS REPO (see Restructure of SDS REPO.doc for details of how this REPO operates). This Clone will either be the latest content of the Master Branch or it will be a version of the software tagged in the Master branch.

Assumption has been made that you have access to the REPO with a SSH connection.

Having downloaded the files it will set up other files ready for SQLcl to process to process, these files are all created currently in the Build folder if you have marked the Clearup parameter to Y these will be destroyed at the end of the process, otherwise they will remain in place.

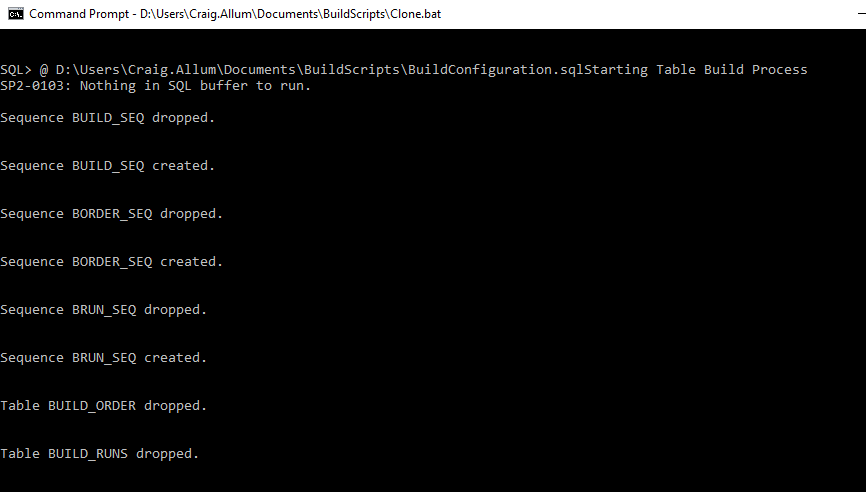
The first set of files generated the \*ForSql.txt

When demonstrating this process everything the build script are all located in D:\Users\Craig.Allum\Documents\BuildScripts. So when deployed a change is needed so that it can live in a different location or the Craig.Allum become relative to the logged in user, but the focus was to get this bet working and prove it works before messing around with the switches and run parameters. Having got all the nessary setup file generated the process will then login automatically as SYS



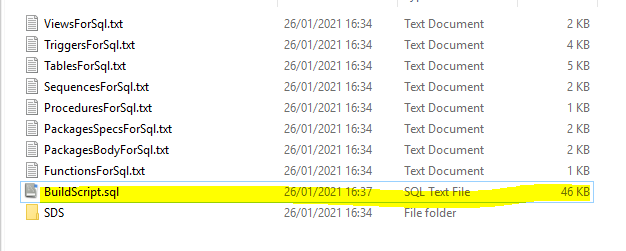
Once logged in it will start to kill connections to the new schema. This could take a while as it wait to see that no sessions are present on that schema before trying to create a new user, as it could be the same user being created. As soon as no session are found it will then potentially drop the schema if stated in the SchemaToDrop parameter. Once that is done it will try to create a new schema from the NewSchemaUsrName, NewSchemaPassword parameters.

Having created the new schema the next script will automatically login as the new schema user, having logged in it will build the auto build tables which are used for this process.

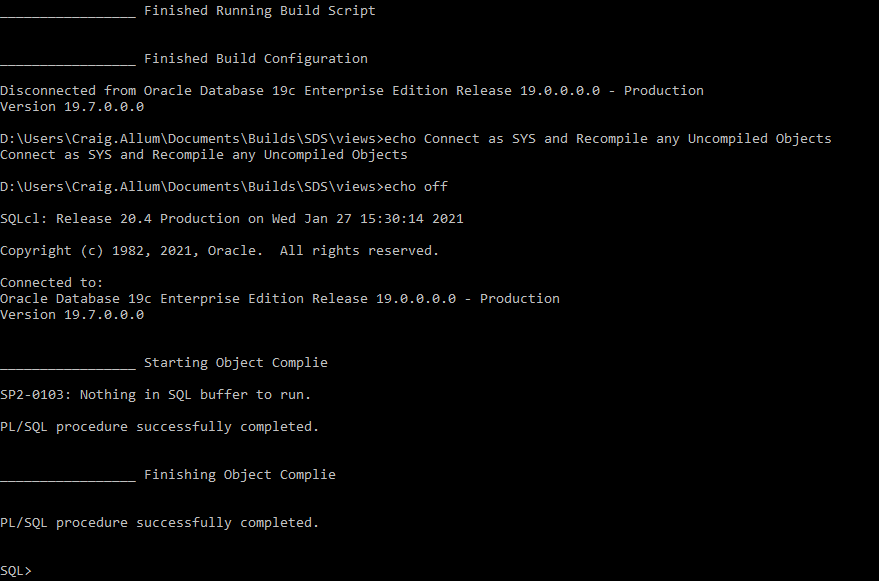


We then populate the configuration data automatically and then we give all the object an initial build order. We then have a section in the script that allows for exceptions to apply for objects so we can adjust the build order as we see fit, currently nothing in here.

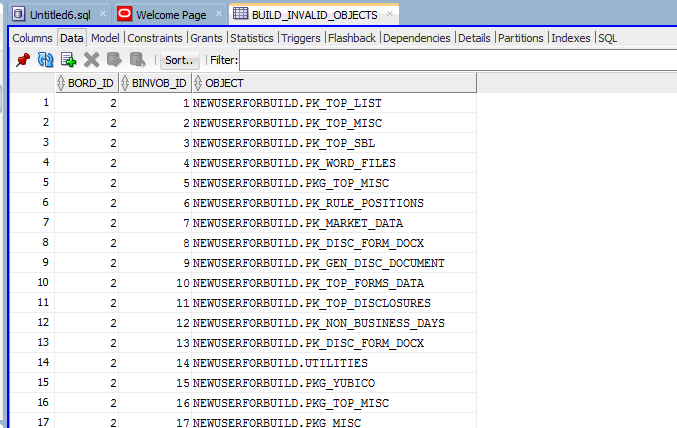
Once the final build order has been set in the build tables we have a process that will come along and build the final script which will follow that order of objects and then it will execute against the new schema.



This will take a while but when it has completed the process will disconnect from the new schema and then it will re connect as SYS and it will try to resolve all the invalid object by recompiling them within this new schema.



If anything is not valid after this it will be present in BUILD\_INVALID\_OBJECTS, which acts like an exception report.



In an ideal world this will be empty but it gives us a start on resolving the issues and possible issue with the build script.

## UPDATE

When the NewOrUpgrade is U instead of N it means we want to update an existing schema. This means it will follow the same steps as NEW above but will the following differences:

We don’t drop schemas when running in update mode.

When processing the scripts for the tables, instead of pointing at the table creation scripts we instead pick up the table fix scripts that will contain ALTER TABLE commands in the table\_fixes folder held in the REPO, thus we don’t drop any tables.

When processing the scripts for the sequences, instead of pointing at the sequence creation scripts we instead pick up the sequence fix scripts that will contain a script that will record the sequence current position, drop the sequence, recreate the sequence with whatever change was needed and then it will be reset to the sequence value that was on the original recorded sequence.

As for the other objects, they have been created in the REPO with CREATE OR REPLACE scripts so will be applied as is.