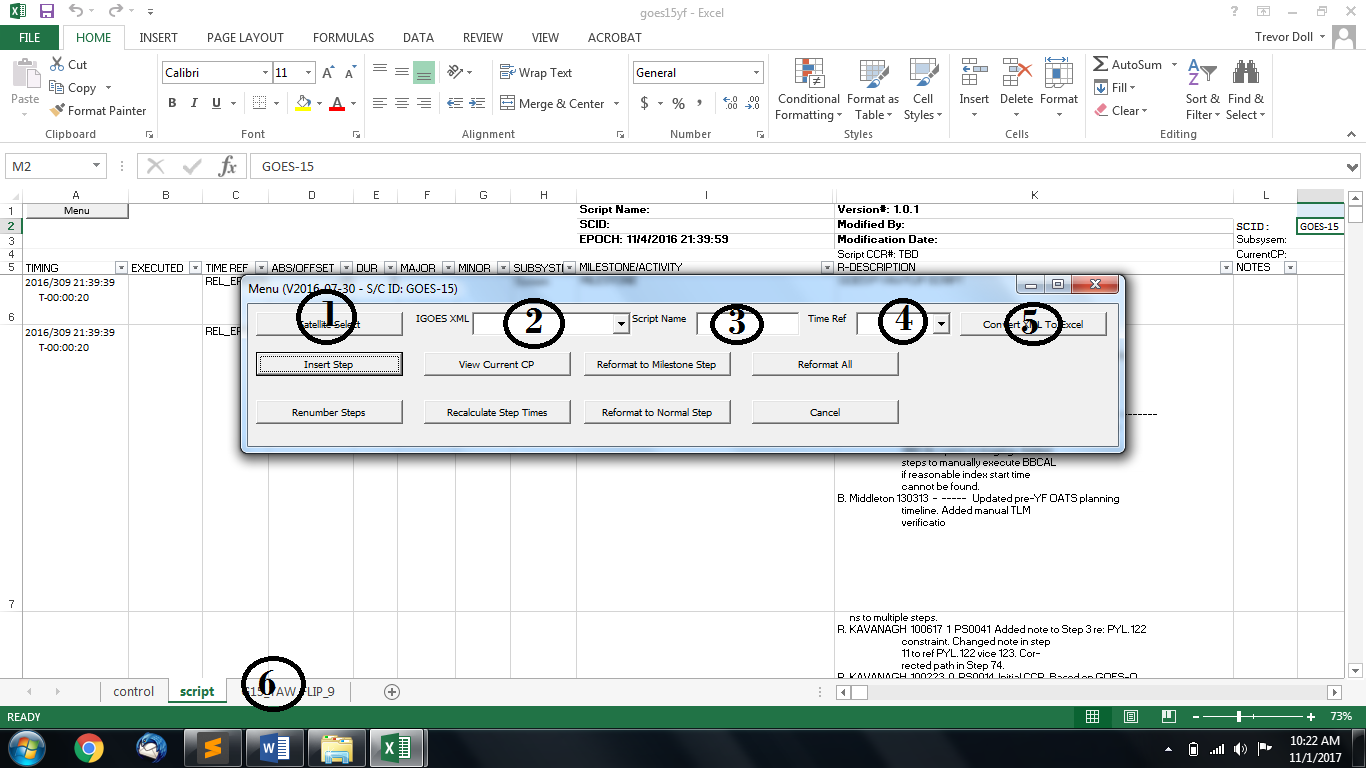
sView Documentation

**To Build SQL Database From Script Tool**

(This is assuming a script was already created and saved using the new NOP Script Tool)



1: Select Satellite

2: Select XML Data of script (exported from IGOES if not already created)

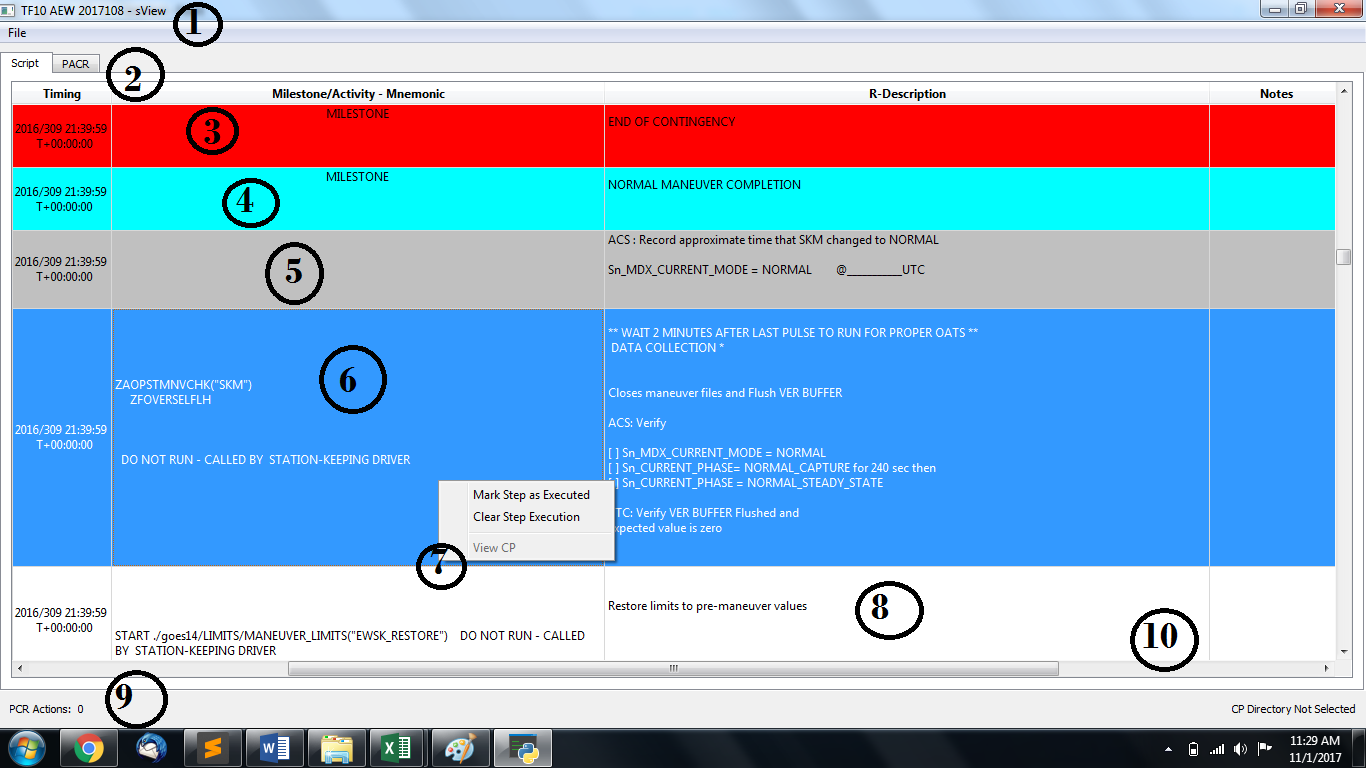
3: Name the script appropriately

4: Select Mode for Time Ref

5: Click ‘Convert XML to Excel’

6: Outputted Excel sheet (this is what we will use)

**sView Operations**



NOTE: Initial startup of sView will take about 30 seconds. This is attributed to sView not being in a local drive but residing in \\goesimage01

To load the newly created SQL file:

**File->Open Script**

An excel book or a database can be opened. If an Excel book is opened, it will look in /*path/to/excels/Maneuvers* to determine whether a database of the selected Excel was already created today. If it was, it will automatically connect to that database. Otherwise, it will convert the Excel to a database and then connect to the newly created database that is located in */Maneuvers*.

The created database file will have the naming convention *ManeuverName\_DOY.db*

The Script and PACR tables will then populate with the data from the created database.

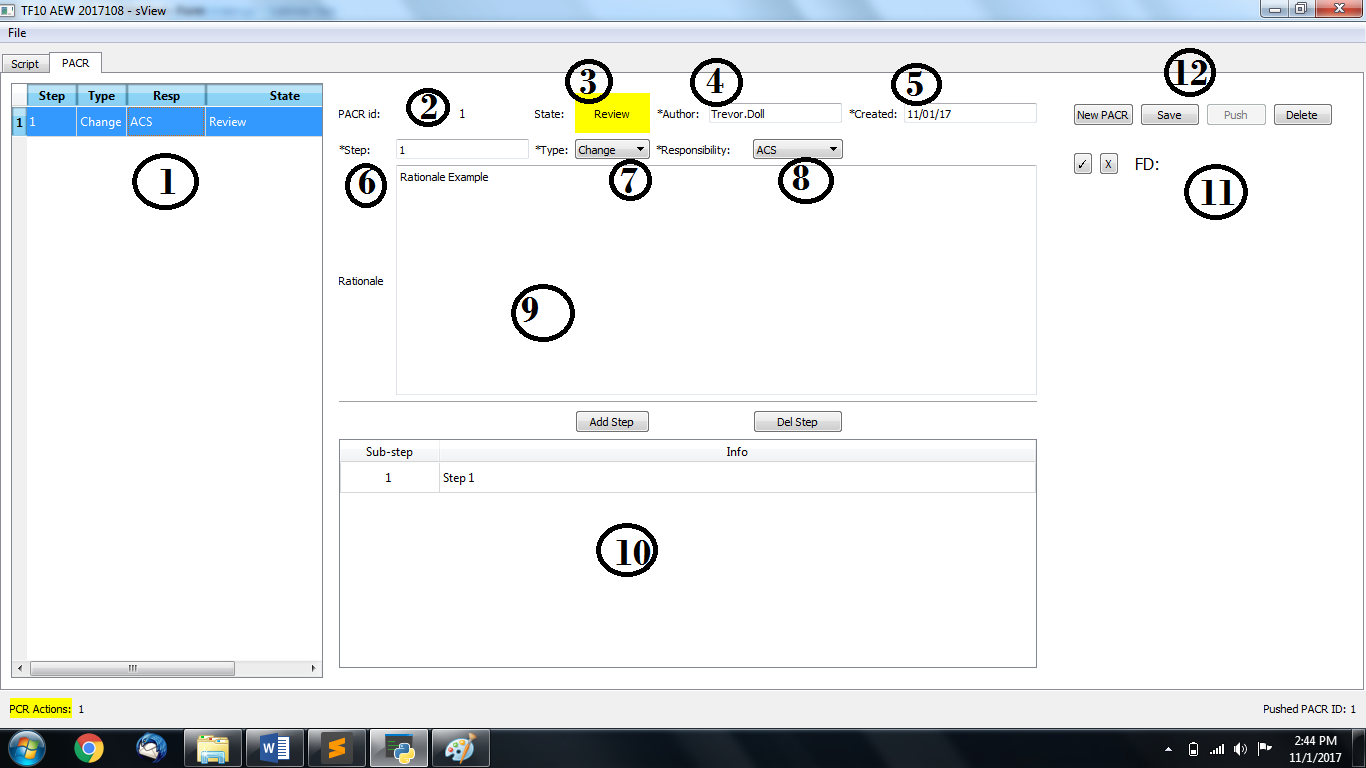
Some things to know before explaining each point:

* **ANY** change to the Script Table, or the PACR table, will result in a real time update to the database and will be seen across machines, also closing the program itself writes any changes to the database
* The column widths on any of the tables can be adjusted, they’re initially set for a non-full screen window, so if full screen it is recommended to adjust the column widths

Script Tab

1. The title of the window will show the current DB that it is connected to. If no connection it will show ‘[script]’
2. The Script and PACR tabs. The Script Tab shows the script table whereas the PACR tab shows the PACR information (duh)
3. A RED row is indicative an ABORT section. It will be red at the beginning of the ABORT Contingency section and at the END OF CONTINGENCY
4. A CYAN (?) row is a MILESTONE and will be listed as such
5. A GREY row is an ‘Executed’ row, this can be toggled by Right Clicking the row (see # 7)
6. A BLUE row is a selected row. **If a row is selected and it contains a CP, the CP will be added to the clipboard. If you select a row whilst holding Ctrl, it will copy the CP parameters**
7. The Context Menu:
   1. Mark Step as Executed
      1. Greys out a row indicating that the step has been executed
   2. Clear Step Execution
      1. Re-Enables the row
   3. View CP:
      1. To view a CP, the CP Directory has to be set. This can be done in **File->Settings**. If that is set, the ‘View CP’ button will be un-greyed. If a row is right clicked and it contains a CP, clicking ‘View CP’ will open up a window and display the contents of the CP.
   4. Quick Note:
      1. Creates a note in the Note column
8. A WHITE row is neither selected nor Executed.
9. PACR Actions counts the number of PACRS in ‘REVIEW’ stage. If more than 0, it will be yellow.
10. Major program commands are listed in the right corner, examples command are ‘Connected to DB XXX’, ‘CP DIR NOT Selected’, etc.

PACR Tab



1. The actual PACR list. Clicking on the PACR record populates the fields with the PACR info, regardless of PACR state. \*If PACR info doesn’t look updated, re-click on record to re-populate
2. PACR ID-Automatically created, simply the length of the PACR list
3. Status – Status of PACR, ‘Dev’ is GREY, ‘Review’ is YELLOW, ‘Approved’ is GREEN.
4. Author – PACR author taken from the machines current user, can be modified
5. Created – Date created
6. Step – Step number to edit. If Adding/Removing there cannot be duplicate step numbers in PACR list, but if Changing and already Approved PACR, a duplicate number is allowed. Step numbers are also limited to number of steps in the script
7. Type – ‘Add’, ‘Remove’, ‘Change’. The action occurs at the defined step number. If Add, the PACR is inserted into the script at the defined step number and all the subsequent steps are pushed down.
8. Responsibility – Subsystem responsibility

**All items with an asterisk (\*) have to be completed in order to ‘Push’ a PACR for approval**

1. Rationale – Explanation of the change that will occur with the PACR
2. Sub-step – If PACR will include commanding or multiple steps add them here. The first step starts at the defined Step Number and the rest are added afterwards. (If only a note is required for a PACR, filling this out is not necessary)
3. FD – Flight Director Approval, if the PACR was pushed into ‘Review’ the FD can either approve it or reject it with the buttons. If approved, it’s implemented into the script, else it is pushed back into the ‘Dev’ state
4. PACR Actions – ‘Save’ a PACR to work on later, ‘Push’ a PACR into ‘Review’, if the PACR was re-saved in the ‘Review’ stage, it’s pushed back into the ‘Dev’ state. ‘Push’ it again to put it into ‘Review’. ‘Approved’ is implemented into the script, no further changes can occur to the PACR. NOTE: Only the user who created the PACR can ‘Save’ and ‘Push’ the PACR. Anyone can approve it.

**Backend Documentation**

Requirements for developing and maintaining sView:

* Python >3.5 (3.6 was used to develop it, but it will run with >3.5)
* PyQt5
  + In terminal navigate to your python installation path and type in *pip3 install –U pyqt5*
  + Qt Designer comes with PyQt5, so if you want to change the front-end Qt Designer is located in the PyQt5 installation directory in the *lib* folder in the Python installation directory
* Pandas
  + To install on a Windows machine go to
    - <https://www.lfd.uci.edu/~gohlke/pythonlibs/#pandas>
    - Download the version that’s matching your Python version and your system architecture
    - Once download is completed open CMD Prompt in the downloaded directory and *pip3 install* whichever pandas .whl you just downloaded
* PyInstaller
  + *pip3 install –U pyinstaller*

All source code files are pretty well commented and hopefully easy enough to follow

To build and distribute sView:

You need to edit *main.spec*. This is the configuration file that tells pyInstaller where everything is and how to bundle sView and its contents.

* Change all the hardcoded directories to your specific directories, anything with a hardcoded directory is labeled *###########Hardcoded Directory Inside###########*

After editing *main.spec* attempt to build sView buy going into the command terminal, navigating to where *main.spec* and running *pyinstaller –clean main.spec*

If there are no errors, sView was successfully bundled to the directory you chose in *main.spec*. To transfer sView to wherever it is being placed. As of writing this, it resides in [\\goesimage01\emoss\sView](file:///\\goesimage01\emoss\sView)

As long as it resides on [\\goesimage01](file:///\\goesimage01), only a few people can actually push updates to sView, the System Admins and Nathan Sanders. Although the sView folder grants all privledges to all users, the admins of the image drive are the only ones that can re-write the .dlls that will be required when updating sView.