Improvements to Python Program for Backup/Restore of EPICS High Voltage Process Variables

Tyler Lemon, Mary Ann Antonioli, Peter Bonneau, Pablo Campero, Brian Eng, George Jacobs, Mindy Leffel, Marc McMullen, and Amrit Yegneswaran
Physics Division, Thomas Jefferson National Accelerator Facility, Newport News, VA 23606
July 25, 2019

Hall C uses an EPICS/CSS-BOY high voltage controls and monitoring system. This note discusses changes made to remove unneeded dependencies from the Python program (HV B/R) for the high voltage backup/restore of Hall C's EPICS high voltage process variables (PVs) [1].

Several issues were discovered with the first version of HV B/R.

- Pyepics was required to be installed in any environment where HV B/R is executed.
- Pre-defined file paths in HV B/R required manual update if HV B/R is moved.
- Using previously generated backup files as templates for new backup files meant that if, between backups the physical channel used for a detector changes, the new backup would be incorrect because it was based on the last, out-of-date backup.
- HV B/R required manual changes to the code to account for physical wiring changes in the high voltage system.

To remedy the above issues, HV B/R was rewritten to use only standard Python modules and take advantage of Jython (Java implementation of Python) packages included in CSS.

Figure 1 shows how these issues were addressed.

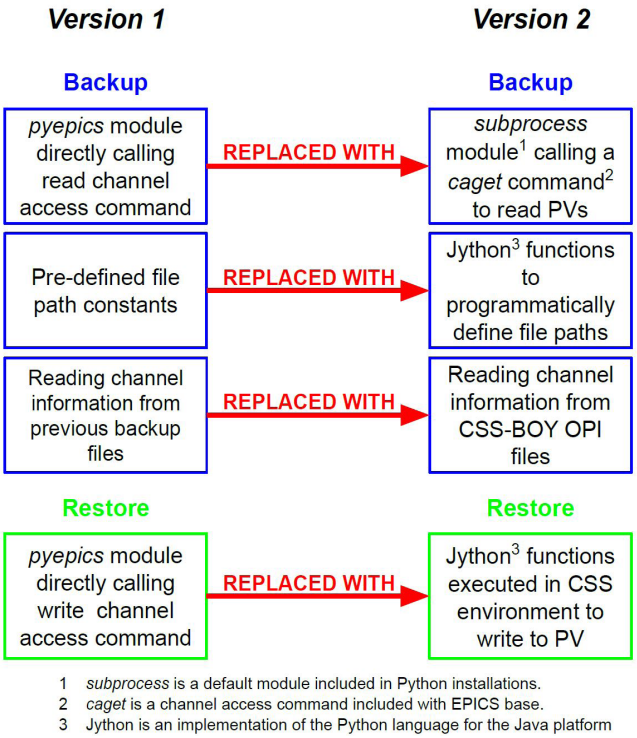

FIG. 1. Changes to version 1 of HV B/R program.

Figure 2 shows the main structure of HV B/R. The standard Python module named subprocess calls the EPICS channel-access command caget (included with all EPICS base installations) for backup operations, Fig. 3, and Jython functions for restore operations, Fig. 4.

The downside of switching to subprocess for backup operations is that the backup takes ~30 seconds (as opposed to only ~3 seconds with pyepics). During testing, Hall C found this difference in execution time to not be an issue.

All pre-defined paths were replaced with a Jython function that returns the location of the script relative to the CSS workspace it is executed in.

The new HV B/R was changed to look at the list view CSS-BOY screens in the CSS workspace to get up-to-date channel information and PVs.

In summary, after testing and use of HV B/R by Hall C, limitations and issues were found. To remedy these issues, pyepics was replaced with subprocess and Jython functions, all pre-defined file paths were replaced with programmatically defined paths, and the reliance on the last backup file as a template was removed by reading PV and channel information directly from list view CSS-BOY high voltage screens. This new version of HV B/R will be included in all future releases of Hall C EPICS High Voltage CSS monitoring and controls screens.

[1] T. Lemon, et al. *Python Program for Backup and Restore of Hall C EPICS High Voltage Process Variables*, DSG Note 2019-20, 2019.

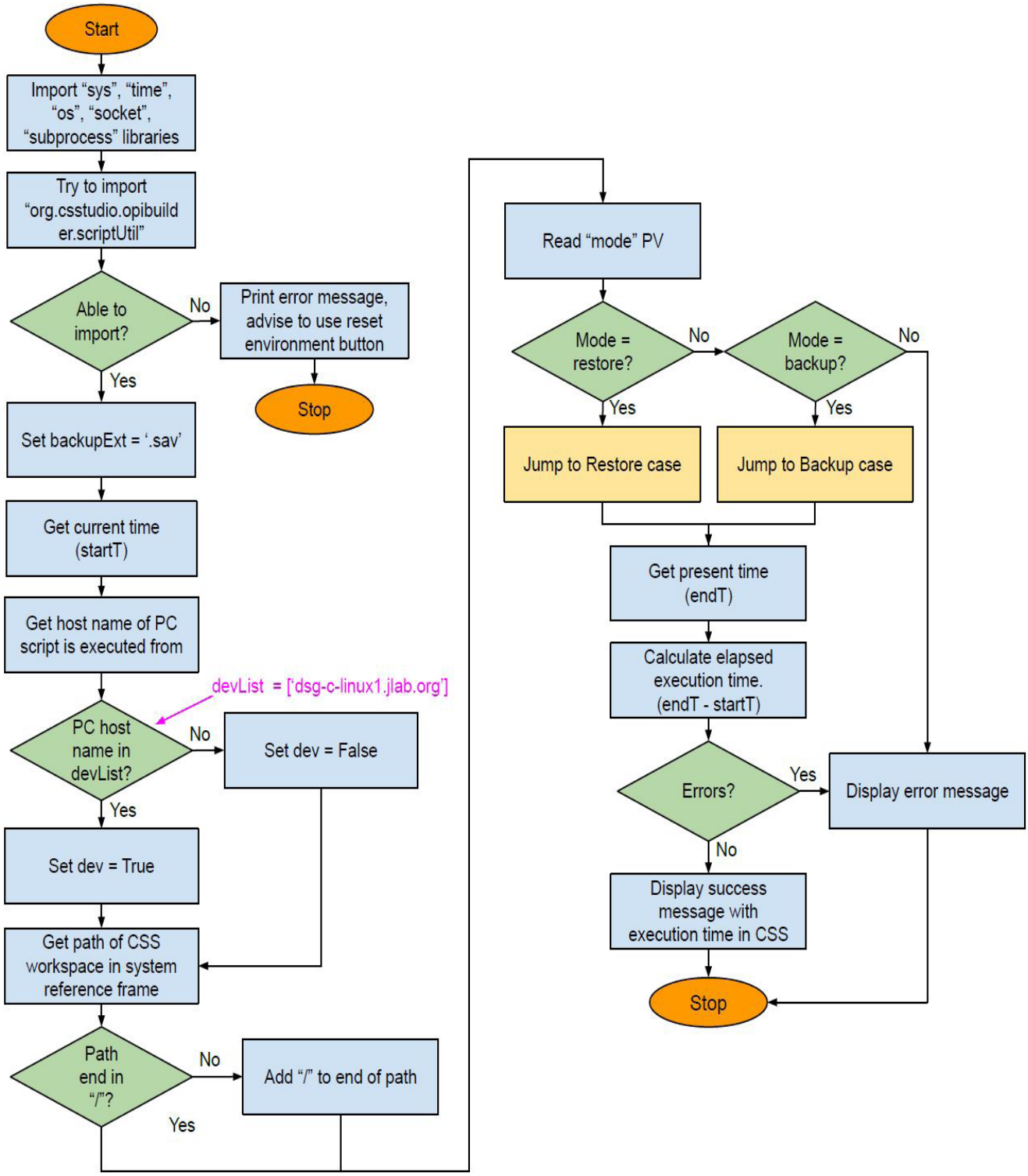

FIG. 2. Flow chart for the main structure of HV B/R program. When executed, the program begins by importing the necessary Python modules and initializing variables needed by the backup and restore cases. Next, depending on the run mode of the program, it enters either the backup or restore case. After completing the appropriate case, it calculates the total execution time of the program and prints this information along with a success or error message to the user in CSS.

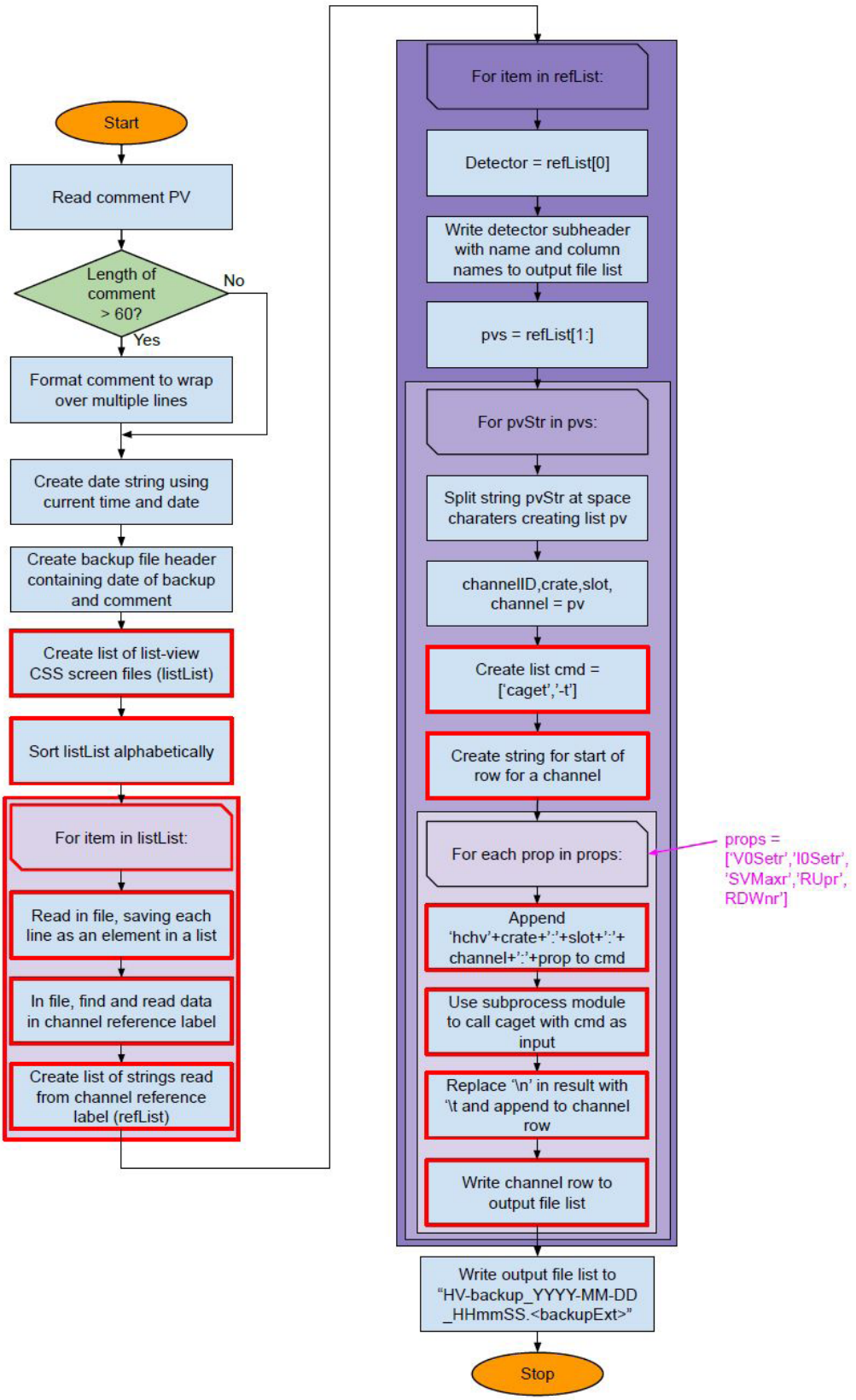

FIG. 3. Flow chart for backup case of HV B/R program. Items in red boxes are modifications for version 2. The backup case of the program starts by formatting user's inputs for the backup file's header. It then reads all CSS-BOY list view screens to get channel information to back up. Next, it formats channel information into the caget command needed by the subprocess module to read channels' present settings. Finally, caget commands are executed and the results are written into a backup file whose name contains the date and time of the backup.

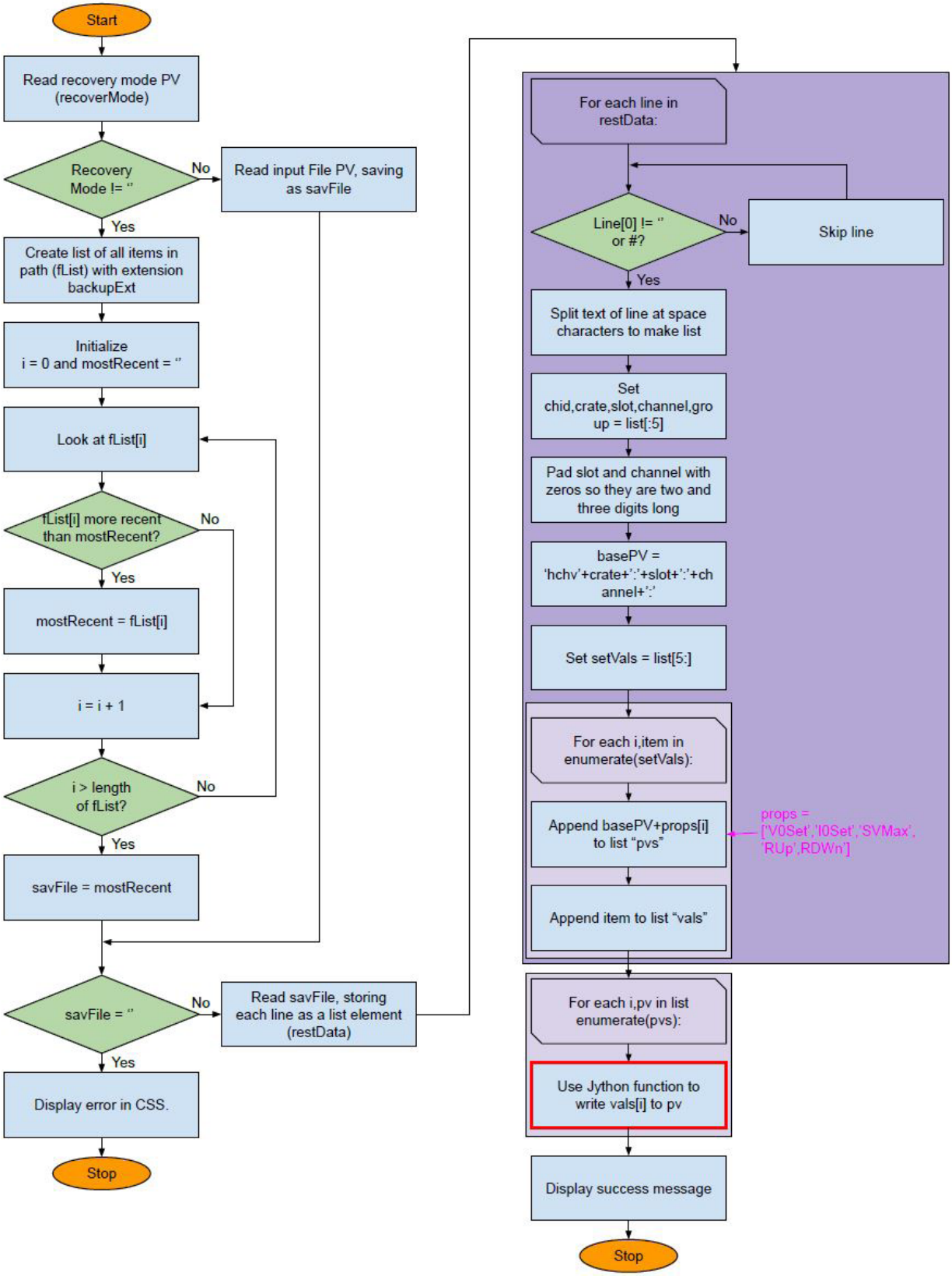

FIG. 4. Flow chart for restore case of HV B/R program. Items in red boxes are modifications for version 2. The restore case starts by determining whether to use the user's input for the file to restore from or to default to the latest backup file. The case then reads the backup file, parsing it for channel information to create PV names and for channel settings. After parsing the backup file, it uses Jython function setValue to write backup values to PVs.