CyFlex® Compressed History Data

Version 6

February 7, 2024

Developed by Transportation Laboratories

Version History

Version	Date	Revision Description
1	1/25/2016	Initial publication
2	8/23/2018	Format to SGS brand
3	4/6/20	Retrofit to new template Added CyFlex 6.3 information: • About auto-tuning in *Section 2.1* Tuning Tolerances • About `@IgnoreList` keyword in *Section 4.3* Specification File • About `HST_freq_lmt` in *Section 5 Utilities, Variables, and Associated Tools*
4	9/28/2021	Added hyperlinked cross-references to cyflex.com usage help where applicable
5	5/25/2022	Updated all hypertext linked cross-references to cyflex.com usage help descriptions
6	2/7/2024	Rebrand to TRP Laboratories

Document Conventions

This document uses the following typographic and syntax conventions.

- Commands, command options, file names or any user-entered input appear in Courier type. Variables appear in Courier italic type.

  Example: Select the `cmdapp-`*`relVersion-buildVersion`*`.zip` file….

- User interface elements, such as field names, button names, menus, menu commands, and items in clickable dropdown lists, appear in Arial bold type.

  Example: **Type**: Click **Select Type** to display drop-down menu options.

- Cross-references are designated in Arial italics.

  Example: Refer to *Figure 1*…

- Click intra-document cross-references and page references to display the stated destination.

  Example: Refer to *Section 1* Overview on page 1.

  The clickable cross-references in the preceding example are *1*, *Overview*, and on page 1.

CyFlex Documentation

CyFlex documentation is available at https://cyflex.com/. View **Help & Docs** topics or use the **Search** facility to find topics of interest.

Table of Contents

1 Overview

CyFlex uses a data compression method to store long periods of data history that can later be used to recover views of the data graphically or for export into a spreadsheet. This document describes how the data is managed, stored, and accessed.

Refer to the *CyFlex® HistoryPlot User Guide* for information on plotting the data.

2 Data Stored for Future Recovery

Only REAL, LOGICAL, INTEGER variables and EVENTS can be stored in the compressed data files. ARRAY, STRING, COMPOSITION, PROPERTY, and EMISSION variables are not saved by this method. STRING variables can be stored by an alternative method but cannot be plotted.

Every variable structure for the above data types includes a `tolerance` attribute that determines the resolution to which the data is stored and recovered

The tolerance specification is the key to the data compression algorithm. The value of the tolerance determines the extent to which data is compressed and thereby affects the data storage requirement, the time it takes to recover the data, and the computer processing load to manage the data compression.

The user should think of the tolerance specification as the accuracy to which the data must be recovered and the resolution to which it can be plotted. If the user wishes to view the data graphically, there is no value to saving data to a higher resolution than a single pixel on the screen used to view it. It is recommended that the tolerance be set to about 0.5% of the expected full-scale range of the variable. However, this can be manipulated somewhat based on the importance of the accuracy of recovery. A parameter such as oil pressure may not need more than 1[psi] resolution to be useful, while a key performance parameter such as intake manifold pressure might benefit from `0.1[in_hg]` resolution. Noisy parameters such as a disconnected analog input channel should have the tolerance set to a very high value or it may create a stream of useless information and a lot of space on the disk. Refer to *Section 2.1 Tuning Tolerances* below for diagnostics.

2.1 Tuning Tolerances

An estimate of the effectiveness of the tolerance setting can be obtained by viewing the size of the “split” files. Use the following command:

```
cd /data/compressed6/split/
ls –lrS
```

This will display the files in order of size. A very large size may indicate a noisy channel or tolerance value that is too small.

This method applies to CyFlex.6.3 and earlier versions, but the file sizes should be smaller in 6.3 due to the auto-tuning algorithm which will continuously modify the tolerance specification, so it is not necessary to select a tolerance as in the earlier versions. If there are large files near the end of this output for variables that will never be of interest for recovery, they can be excluded by editing the specification file. Refer to *Section 4.3 Specification File* on page 5for an example of how to do this (only in 6.3 and newer).

The auto-tuning feature can be adjusted to collect more or less data based on a frequency calculation. Two variables are available to adjust the auto-tuning algorithm, `HST_freq_lmt` and `HST_int_freq_lmt` for REAL and Integer variable, respectively. These two variables are created in the `perf_labels` file but can be overridden by an entry in the `/cell/cell_special` file as in the example in *Section 6 Split Files* on page 7, or temporarily with the `svar` command. Setting these variables to a larger value will result in lower tolerance settings and larger history files and vice-versa.

3 Data Compression

3.1 Data Compression Method

For LOGICAL and INTEGER variables, every change of value is saved. For REAL variables, the data compression manager uses two methods for reducing the storage.

1. First, no data will be saved until the value changes by at least the value of the tolerance.
2. Secondly, the manager keeps track of a small portion (up to about 20 values) of the changes which do exceed the tolerance.

These values and their corresponding time stamps are saved in a buffer for each variable. Each time a new value is added to the buffer, a linear regression analysis is performed on the data in the buffer. As long as the values in the buffer do not deviate from a straight line by more than the tolerance values, the values are not stored on disk. If the buffer fills or a value deviates by more than the tolerance, then the end points of the portion of the buffer which met the tolerance specifications are saved.

3.2 Data Compression Manager

The data compression manager, delta_hst scans all REAL, INTEGER, and LOGICAL variables in the system to perform the basic check for whether or not the change in value exceeds the tolerance specification. Scanning is performed at the MED process interval specified for the system on the command line of the “scheduler” application in the /cell/go.scp startup script. If the tolerance is exceeded, the manager then performs the linearity check and decides whether or not to save the data on disk. If the manager is not running in the system, none of this processing will take place.

3.3 Compressed Data Storage

Data that is to be saved is stored in a binary format in the /data/compressed6/ directory. The filenames indicates the date and time that data was most recently written to the file. All the compressed data filenames have the .hst extension.

Example:

```
201412261000.hst     (10:00am on Dec 26, 2014)
```

The compressed files are mostly unreadable since they contain mostly binary data, but there is a header section of each file which is written in text format and contains all the variable labels that are included in the file and their initial values when the file was first opened. The compressed files normally contain only 30 minutes of information and should have a timestamp that occurs on the half hour.

The compressed files may contain less than 30 minutes of data if the user is changing specifications for variables that are to be included or excluded from the list of stored variables. When the list of variables changes, the previously open *.hst file will be closed and a new one created.

The compressed files are removed from the system after they reach a certain age. This is managed by the `cleanup` utility run by the system `cron` process. Typically, there is a specification file named `/specs/usercron` that supplies the cron utility with the specification for how to manage the removal of old `*.hst` files. The following entry in a `/specs/usercron` file will remove `*.hst` files that are older than 14 days and the older files if the number exceeds 1000.

```
/data/compressed/*.hst  14  1000  0 NO
```

or

```
56 * * * * /cyflex/bin/cleanup 14 1000
“/data/compressed/*.hst” 1>/dev/null 2>&1
```

See *CyFlex Usercron Utility* the for `usercon` information. Refer to cyflex.com usage help for details on the `cleanup` command.

Type `crontab -l` on the command line to view the current settings of a system.

The 14-day cleanup means that the user will be able to recover history data for active variables for up to the previous 14 days. This is calendar time, not engine running time. With the large disk sizes currently available, this time period could be extended considerably without stressing disk capacity.

4 Starting the Applications

4.1 Data Compression Manager

Command line format for the data compression manager:

```
delta_hst &
```

The manager is normally launched in the `/cell/go.scp` script file. Its position in this file is not important and it could be one of the last applications launched, but it must be followed by the `delta_specs` command. The `delta_hst` command has no options.

4.2 Delta Specs

The `delta_specs` application configures and activates `delta_hst`. It reads the specification file, `/specs/delta_specs.`*NNN* where *NNN* is the cell name. The `delta_specs` application may be run again at a later time without restarting `delta_hst` if the contents of `/specs/delta_specs.`*NNN* are changed. The `delta_specs` command has no options.

Command line format for the delta specs application:

```
delta_specs
```

4.3 Specification File

The `/specs/delta_specs.NNN` file is used only to identify events that are to be included in the history files. Up to 64 event names may be listed, one per line. In addition, for CyFlex.6.3, it is possible to exclude variables from being saved. For example, the `watch_dog` variable continually toggles between 0 and 1 and is not of much interest, so it can be excluded. Likewise, for systems with Gantner I/O there is a garbage variable called `Timestamp_eblox` which contains random data and should always be excluded. After the section of event names, enter the `@IgnoreList` keyword followed by the variables to be excluded.

Example file format:

```
emergency
abort_limit
@IgnoreList
watch_dog
countdown
Timestamp_eblox
asam3_1_Interval1
asam3_1_Interval2
asam3_1_Interval3
asam3_1_Interval4
asam3_1_Update_Interval1
asam3_1_Update_Interval2
asam3_1_Update_Interval3
asam3_1_Update_Interval4
$
```

5 Utilities, Variables, and Associated Tools

There are several command line utilities available that are not normally required if the HistoryPlot application is used to view that data, but they may prove useful for special cases. Detailed command line options are available for each utility on cyflex.com.

Refer to *Section 6 Split Files* on page 7 for applications of the following.

- The `delta_split_all` application will re-build the split files from the `*.hst` files using the current value of `hst_window` to decide the time period for the split files. Refer to cyflex.com usage help for `delta_split_all`.
- The `list_all_split_data` application will print a text-readable listing of the contents of a split file. Refer to cyflex.com usage help for `list_all_split_data`.
- The `flush_delta` command will send a message to the `delta_hst` application telling it to update the latest `*.hst` file with current data. This will ensure that the split data files will contain data up to the current time. It is recommended that on a running system that this be executed prior to running HistoryPlot if the data of interest includes the prior 30 minutes.
- The `hst_window` variable defines the time period for split files. The default value at system startup is defined in the `/cell/cell_special` file. The value may be temporarily modified with the `svar` command prior to running the `delta_split_all` application.
- `HST_freq_lmt` is an adjustable auto-tuning variable with units of [hz]. This variable applies to REAL variables. The value of these variables is used as a limit for the rate at which history data is captured and is used to adjust the tolerance value of variables. If a variable exceeds the current tolerance specification at a rate higher than this limit, then the tolerance value will be increased slightly. If a change in variable value doesn't exceed the tolerance for over 1 minute, then the tolerance value is decreased slightly.
- `HST_int_freq_lmt` is an adjustable auto-tuning variable with units of [hz]. This variable applies to `INTEGER` variables and functions the same as `HST_freq_lmt`.

6 Split Files

The `*.hst` files in the `/data/compressed6/` directory are extremely large and contain intermingled data from all active variables. To speed up the extraction and decompression process, the individual variables get separated out by a low-priority non-real-time process called `delta_split`. This process is a child process of `delta_hst` that is launched when `delta_hst` is launched. When the delta compression manager `delta_hst` closes its most recent `*.hst` file, it then sends a message to `delta_split` to extract all of the variables in the `*.hst` file into individual files for each variable and store them in the `/data/compressed6/split/` directory. These are the files which will be read by the HistoryPlot user interface and other extraction tools. The filenames in the `/data/compressed6/split/` directory have the same name as the variable label.

The time period covered by each of the split files will be the time period defined by the CyFlex variable `hst_window`. This variable will normally be 7 to 14 days. The `hst_window` variable default values is created by the `perf_labels` utility and can be modified at each cell by an entry in the `/cell/cell_special` file. Refer to cyflex.com usage help for `perf_labels`. A file example follows.

```
#################################################################
#                     /cell/cell_special
#
# This file contains information about the test cell
# configuration and should be modified by the site
# administrator.  Do not add lines or change the index
# numbers, labels, or units.  Modify only the initial value,
# history flag (hst), and history tolerance (tol)
#
#index      label        units   format initial_value   hst     tol

28          hr_meter_th    rpm         0        300      OFF       1.0
65          hst_window     day         1        3        OFF       1.0
$PI
22          pwr_meas_type  none        1                OFF          -
# pwr_meas_type   0=none, 1=thru speed/torque, 2=electrical monitor

23          elog_UNX       none        0               OFF          -
#elog_UNX  1=launch elog_entry upon restart after unexpedted shutdown


$PFR

18          stoic_fa       NONE        4       .067      OFF       1.0
19          fuel_density   lb/ft3      3        53.065   OFF       1.0
20          target_fr_tm   min         1        2.0      OFF       1.0
30          buoyancy_CF    none        4        1.0      OFF       1.0
33          TRV_max_age    hr          0        -        OFF       1.0
$PFI
#index      label          units initial_value  hst   transition_event
```

```
$PFRDC

$PFRTC

$PFSTRINGS

0        FR_log_file "/data/fuel_log/default.log"
$END
```

The contents of a split file are still in binary form just like the `*.hst` file, except that each file contains only the values for a single variable. The contents of a single file may be viewed with the utility `list_all_split_data`.

Example:

```
list_all_split_data  [label/filename]
list_all_split_data   acc_rtn_t | less
```

The output will show the timestamp and value of the variable for a period of time defined by the `hst_window` variable starting from *approximately* the current time and going back in time by the value of `hst_window`. In other words, the `end_time` will usually be from the previous even half hour back to a `start_time` which will be (`end_time` – `hst_window`). This period of time may not be from exactly the current time, because the split files are normally only updated every half hour. The period may be forced to have an `end_time` which is equal to the current time by running the "`flush_delta`" process before running `list_all_split_data`.

Example:

```
flush_delta
list_all_split_data acc_rtn_t  |  less
```

6.1 Rebuilding Split Files

The split files are continuously produced while CyFlex is running and the `delta_hst` application is active, but since they are derived from the compressed `*.hst` files in the `/data/compressed6/` directory, they can be reconstructed if desired. Note that the reconstruction of split files will be based on the time period that is specified with the `hst_window` variable and cannot be larger than the period of time spanned by the `*.hst` file. Refer to *Section 3.3 Compressed Data Storage* on page 3 for related information.

Use `delta_split_all` to build the split files by extracting data from the `*.hst` files but limiting the split files to the `hst_window` time period.

Example:

```
delta_split_all –r
```

The `–r` option removes all of the existing split files and then rebuilds them.

Running `delta_split_all` may take several minutes.

To review data only for a short period of time, such as the previous 24 hours, the performance of the HistoryPlot utility or other extraction utilities can be improved considerably by limiting the size of the split files to this time period.

Example:

```
svar hst_window 1      (set window to 1 day)
delta_split_all –r     (run viewing utility when delta_split_all completes)
```

Conversely, when a longer time window is desired.

```
gvar hst_window        (determine current window size)
svar hst_window 14     (increase the window size if necessary)
delta_split_all –r     (run viewing utility when delta_split_all completes)
```