WHEN YOU NEED TO BE SURE

CyFlex® Statistical Variables and Sampling

Version 7

May 15, 2023

Developed by SGS North America, Inc.

Version History

Version	Date	Revision Description
1	12/2/2014	Initial publication
2	9/18/2015	Added section Releasing an Application
3	8/23/2018	Formatting/branding revisions
4	3/31/2020	Retrofit to new template
5	8/26/2021	Removed all inline command usage content and replaced with hyperlinked cross-references to cyflex.com usage help
6	5/23/2022	Updated all hypertext linked cross-references to cyflex.com usage help descriptions
7	5/15/2023	Added descriptions of new specification file keywords `@ALWAYS_UPDATE_CHANNELS` and `@EXCLUDE_DUPLICATES`. Added *Section 5 Minimum Statistical Sampling* on page 10

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

LIST OF TABLES

1 Overview

CyFlex performs statistical analysis using several different sampling methods. Sampling inputs can be any Real variable in the system. The different sampling methods are accomplished by the following techniques:

- Periodic computes statistics on samples at regular intervals over a time duration controlled by CyFlex events. Refer to *Section 2 Periodic Statistical Sampling* on page 3 for details.
- Event-driven computes statistics timed solely on CyFlex events, specified for sampling and computation. Refer to *Section 3 Event-Driven Statistical Sampling* on page 5 for details.
- Continuous computes statistics continuously for sampling during a moving time window. Refer to *Section 4 Continuous Statistical Sampling* on page 7 for details.

1.1 Statistical Variables

Statistical variables contain several double precision floating point values which are the computed results from statistical sampling operations performed on a "real" variable. There is a single real variable associated with each statistical variable. This is called the "source variable".

The label of a statistical variable always ends in a "dot" (period), for example, `blow_by.` Access to the different values is obtained through the use of a 2-character extender following the “dot”. For instance, the average or mean can be obtained by using the label `blow_by.`AV.

The information contained in a statistical variable is a function of the type of sampling process. For example, the event-driven `snap_stats` application will not compute the slope and intercept of a linear regression since there is no defined rate for sampling. Refer to *Section 3.1 Starting the Event-Driven Statistical Sampling Application* on page 5 for related information.

Table 1 summarizes the available extenders.

Table 1: Available Extenders and Values

Extender	Value
AV	mean
SD	standard deviation
MN	minimum value
MX	maximum value
RN	range (MX-MN)
IV	initial value
FV	final value
ST	start time
EN	end time

Extender	Value
DL	delta
TN	time of minimum
TX	time of maximum
CV	coefficient of variation
IN	integral (newton's method)
IT	integral (trapezoidal method)
NM	number of samples
C1	slope
C0	intercept
RS	r-squared
RA	rate of sampling
E0	standard error
E1	standard error of slope
CI	confidence interval – per requested confidence RC
I0	90% confidence interval
I5	95% confidence interval
I9	99% confidence interval
ME	median of data set
MD	abs value of deviation from median
SE	student-t
RC	requested confidence

2 Periodic Statistical Sampling

Use the `statistics` application to sample over some time interval determined by system events.

The events may be derived from some other CyFlex process, such as the start and end of a fuel reading. They might be set by the Test Manager, or they could be set through hotkeys or commands.

The `start_event` starts the sampling session. The session may be terminated either by the number of samples reaching a specified maximum or by another occurrence of the `start_event`. If a sampling session is still in progress each time the event is received, the sampling statistics are computed and a new sampling interval is automatically started. In this way, the same event can define both the end of the previous interval and the start of the next one.

If no `start_event` is included in the specification file, then sampling begins immediately when the application is launched. In this case, when sampling is complete, it cannot be restarted without slaying and restarting the application. Refer to *Section 2.2 Periodic Statistical Sampling Specification File Format* below,

An event may be specified that will signal completion of the session and computed results are complete. This event indicates the results are available to other processes and can be used to synchronized data logging or other data collection processes. This event, the `done_event` will be created if it does not already exist.

The output variables will be created as STAT_VARIABLEs. The input variables would usually be analog inputs or performance variables but can be any REAL variable.

2.1 Starting the Periodic Statistical Sampling Application

Launch the application from either the command line or from a script file. The only argument is the name of the specification file. Multiple instances of the application may be running simultaneously with different specifications. There is no default filename.

Examples:

```
statistics /specs/asset/specs/stats.315 &
statistics /specs/asset/specs/ns &
```

Refer to cyflex.com usage help for `statistics` for related information.

2.2 Periodic Statistical Sampling Specification File Format

The specification file contains a header line which defines the events which will signal the start and end of a sampling interval and the sampling rate. This is followed by up to 32 lines of variable names. Each line contains the label of the variable to be sampled and the names of the variables where the results will be placed. The following is an example of a specification file.

```
#start_event  end_event    done_event  Interval  max_samples
fr_ave_strt   fr_ave_stop  sync_log    1[s]      0
#input_variable   output_variable
rail              srail.
RPM               sRPM.
flotron           sflotron.
```

Table 2 describes the purpose of each data field.

Table 2: Periodic Statistic Sampling Specification File Data Fields

Data Field	Function
`start_event`	The name of an event which signals the start of a sampling interval. If a sampling session is already in progress when this event is received, the session is stopped, results computed, and a new session is immediately started.
`end_event`	The name of an event which signals the end of a sampling session. This event is optional and is not needed if sessions are to be performed repetitively and controlled only by the `start_event`.
`done_event`	The name of an event which signals the results of the session are available to other processes.
`Interval`	The sampling rate in milliseconds/sample.
`max_samples`	The maximum number of samples which comprise a sampling session. The session will be terminated when the number of samples equals this number, or the `stop_event` is received. A zero is used to indicate an indefinite number of samples.
`input_variable`	The variable which is to be sampled.
`output_variable`	The variable where the results will be written.

3 Event-Driven Statistical Sampling

Use the `snap_stats` application when the sampling and computation of results are to be completely controlled by system events.

The events may be derived from some other CyFlex process or set through hotkeys or commands.

A `sample_event` is required in the specification file to start each sampling scan. Multiple variables may be sampled when this event is received. Use the `compute_event` to signal that the computation is to be done and results placed in the output variables. Optionally, specify a `results_event` to signal that the computation is complete. Finally, a `clear_event` must be received if a sampling session is to be restarted. The `clear_event` causes the application to reset various internal buffers and counters so that the next sampling session will begin with a clean start. Refer to *Section 3.2 Event-Driven Statistical Sampling Specification File Format* below for related information.

A sampling session consists of setting a `clear_event`, one or more `sample_events` to control collecting of data and a `compute_event` to compute results It is possible to get intermediate results after each sample by setting the `compute_event`. The internal buffers will continue to accumulate data until the `clear_event` is received. It is also possible to specify the `sample`" and `compute` events as one and the same, and thus compute the results for each sample automatically.

The specification file permits up to eight (8) groups of variables to be specified, each with different controlling events.

Refer to cyflex.com usage help for `snap_stats`.

3.1 Starting the Event-Driven Statistical Sampling Application

Use the `snap_specs` application to configure and launch `snap_stats` with a specification file. Start the application the command line or from a script file such as `/cell/go.scp`. The only argument is the name of the specification file. The default filename is `/specs/ss_specs.`*`NNN`*, where *NNN* is the test cell name. Refer to cyflex.com usage help for `snap_specs`.

Examples:

```
snap_specs &
snap_specs /asset/specs/ns &
```

3.2 Event-Driven Statistical Sampling Specification File Format

The specification file consists of up to eight (8) groups of specifications, separated by the $ character. Each group contains a header line which defines the controlling events. This is followed by up to 64 lines of variable names. Each line contains the label of the variable to be sampled and the names of the variables where the results will be placed.

The following is an example of a `snap_stats` specification file.

```
#sample_event computed_event results_event clear_event
ss_samp       ss_compute     ss_result     ss_clear
#input_variable     output_variable
```

```
rail                ss_rail.
RPM                 ss_RPM.
flotron             ss_flotron.
$
#sample_event computed_event results_event clear_event
ss_samp       ss_compute     ss_result     ss_clear
#input_variable     output_variable
oilp                ss_oil.
$
```

Note:

In the preceding example, the `compute` and `clear` events are the same for each group, but the `sample_event` is different. The `results_event` is optional.

Table 3 describes the purpose of each data field.

Table 3: Event-Driven Statistic Sampling Specification File Data Fields

Data Field	Function
`sample_event`	The name of an event which signals that a sample should be taken.
`compute_event`	The name of an event which signals that results are to be computed for all samples taken since the last `clear_event`.
`results_event`	The name of an event which signals that the results of the session are available to other processes.
`clear_event`	The name of an event which signals that the internal buffers which have been used to collect samples are to be cleared
`input_variable`	The label of the variable to be sampled.
`output_variable`	The label of the `STAT_VARIABLE` where the results will be written.

4 Continuous Statistical Sampling

Use the `run_aver` application to sample and compute during a moving time window. The length (number of samples) of the window and the sample rate can be specified independently for each variable.

The `run_aver` process obtains a "running average" by maintaining a FIFO (first in, first out) buffer of data of the specified number or samples. Each time a new value is added, the oldest value is removed from the buffer. The computations are done each time a new value is sampled for each variable.

Typical applications of `run_aver` use the mean value as a method of filtering a variable. The standard deviation can be used to gauge the variability or noisiness of a variable.

4.1 Starting the Continuous Statistical Sampling Application

Start the application by using the typical method of using the `/cell/go.scp` startup script. It may also be started from the command line. The arguments are the registered task name, the priority, and the supported sampling rates.

The process may also be made "critical" with the optional `+c` argument. A "critical" task will cause the watchdog timer to activate if the task fails. Refer to cyflex.com usage help for `run_aver`.

Examples:

```
run_aver RunAver 11 40 100 1000 +c &
run_aver RA_smoc 12 40 &
```

Configure the task with the `ra_specs` command. The only argument is the name of the specification file. The default filename is `/specs/ra_specs.`*NNN*, where *NNN* is the test cell number. Refer to cyflex.com usage help for `ra_specs`.

```
ra_specs <filename>
```

Examples:

```
ra_specs
ra_specs /specs/ra_specs.spcl
```

4.2 Continuous Statistical Sampling Specification File Format

The specification file may contain up to 100 variables to be sampled and the corresponding output variables for the outputs. The output variables will be created as STAT_VARIABLEs. The input variables can be any REAL variable.

The interval of the running average for `rave` may be defined in one of two ways.

1. Define the interval by specifying the number of samples that should be taken at the specified rate.
2. Define the averaging interval via 'start' and 'stop' events. The variable(s) are averaged at the specified rate.

The following optional keywords may be specified in the spec file.

4.2.1 @REG_NAME

```
@REG_NAME
     GL_smoc
```

When this keyword is specified, the line following the keyword registers this running average task with the operating system. If this keyword is not present, the running average task is registered as `RunAver` with the operating system.

4.2.2 @START_STOP_FIFO_EVENTS

A special optional keyword can be included in the specification file if it is necessary to dynamically change the sampling interval.

```
@START_STOP_FIFO_EVENTS

# start event name   stop_event_name   valid_to_compute_label
    profile_start       profile_done              -
```

When this keyword is specified, the line following the keyword contains the names of the CyFlex events that will define the averaging interval.

Table 4 describes the fields in more detail.

Table 4: Continuous Statistic Sampling Specification File Data Fields

Data Field	Function
`start event name`	The name of the event that, when received, defines the start of the averaging interval. **NOTE:** The `valid to compute` label, if specified, must be `TRUE` when this event is processed.
`stop event name`	The name of the event that, when received, defines the end of the averaging interval
`valid to compute label`	The label of a `LOGICAL` variable that can be used to reset the running average. When the value of this variable is `TRUE` the running average is being computed. When the value of this variable is `FALSE`, the running average buffers are zeroed and no sampling or computations are made until the variable becomes `TRUE`. If this feature is not desired then enter a '`-`' for the variable label

The file may contain up to 100 lines of variable names. The format of each line depends on whether or not the `@START_STOP_FIFO_EVENTS` keyword is present. If this keyword is not present, then the averaging interval is defined via of the specification of the variables to average as described in *Section 6.1.1 Variable Specification when @START_STOP_FIFO_EVENTS is not Present* on page 13.

4.2.3 @ALWAYS_UPDATE_CHANNELS

The @ALWAYS_UPDATE_CHANNELS keyword always updates the calculated STAT_VARIABLE at the rate listed in the spec file. This is the default behavior for run_aver. This is performed even when the source variable is not updated. Use caution because it is possible to average an unchanging variable. This keyword is mutually exclusive with the @EXCLUDE_DUPLICATES keyword.

4.2.4 @EXCLUDE_DUPLICATES

The @EXCLUDE_DUPLICATES keyword skips a stat variable calculation if the source variable was not updated. This is based on the timestamp of source variable being changed from the previous calculation.

This keyword is mutually exclusive with the @ALWAYS_UPDATE_CHANNELS keyword and overrides the default behavior of @ALWAYS_UPDATE_CHANNELS. @EXCLUDE_DUPLICATES takes precedence when used.

4.3 Sample run_aver Specification

The registered name below requires a corresponding version of run_aver running with that registered name. A sample command string:

```
run_aver RA2_smoc 11 1000 &
```

```
@REG_NAME
  RA2_smoc

@START_STOP_FIFO_EVENTS

 # start event name  stop_event_name    valid_to_compute_label
   profile_start       profile_done              -

#in_variable  num_samples  rate (FAS/MED/SLO)  result_variable

   TORQUE        200              MED            a_torque
   Dyno_CM       200              MED            a_Dyno_CM
   tvo            30              SLO            a_tvo
   tvc            30              SLO            a_tvc
   some_var      100              my_event       a_some_var
```

5 Minimum Statistical Sampling

Use the `rave` application to calculate a minimum number of statistical fields. This application performs a running mean average of a `REAL_VARIABLE`. It is a trimmed-down version of the `run_aver` application and only calculates a minimum number of statistical fields.

5.1 Starting the Minimum Statistical Sampling Application

Start the application within the `/cell/go.scp` startup script. It may also be started from the command line.

The process may also be made "critical" with the optional `+c` argument. A "critical" task will cause the watchdog timer to activate if the task fails. Refer to cyflex.com usage help for `rave`.

Example:

```
rave RA_smoc 14 1000 +c &
```

Configure the task with the `rave_specs` command. The only argument is the name of the specification file. The default filename is `/specs/ra_specs.`*NNN*, where *NNN* is the test cell number. Refer to cyflex.com usage help for `rave_specs`.

```
rave_specs <filename>
```

Examples:

```
rave_specs
rave_specs /specs/rave_specs.spcl
```

5.2 Minimum Statistical Sampling Specification File Format

The interval of the running average for `rave` may be defined in one of two ways.

1. Define the interval by specifying the number of samples that should be taken at the specified rate.
2. Define the averaging interval via 'start' and 'stop' events. The variable(s) are averaged at the specified rate.

The following optional keywords may be specified in the spec file.

5.2.1 @REG_NAME

```
@REG_NAME
      GL_smoc
```

When this keyword is specified, the string on the line following the keyword registers this as the running average task with the operating system. If this keyword is not used, the running average task is registered as `Rave` with the OS.

5.2.2 @START_STOP_FIFO_EVENTS

A special optional keyword can be used in the specification file if it is necessary to dynamically change the sampling interval.

```
@START_STOP_FIFO_EVENTS

# start event name   stop_event_name   valid_to_compute_label
    profile_start       profile_done            -
```

When the @START_STOP_FIFO_EVENTS keyword is specified, the line following the keyword contains the names of the CyFlex events that define the averaging interval.

Table 5 describes the fields in more detail.

Table 5: Minimum Statistic Sampling Specification File Data Fields

Data Field	Function
start event name	The name of the event that, when received, defines the start of the averaging interval.
stop event name	The name of the event that, when received, defines the end of the averaging interval.
valid to compute label	The label of a LOGICAL variable that can be used to reset the running average. When the value of this variable is TRUE, the running average is being computed. When the value of this variable is FALSE, the running average buffers are zeroed and no computations are made until the variable becomes TRUE. If this feature is not desired, then enter a '-' for the variable label.

The file may contain up to 100 lines of variable names. The format of each line depends on whether or not the @START_STOP_FIFO_EVENTS keyword is used. If this keyword is not present, then the averaging interval is defined via of the specification of the variables to average as described in *Section 6.2.1 Variable Specification when @START_STOP_FIFO_EVENTS is not Present on page 15*.

5.2.3 @ALWAYS_UPDATE_CHANNELS

The @ALWAYS_UPDATE_CHANNELS keyword always updates the calculated STAT_VARIABLE at the rate listed in the spec file. This is default behavior for rave. This is performed even when the source variable is not updated. Use caution because it is possible to average an unchanging variable. This keyword is mutually exclusive with the @EXCLUDE_DUPLICATES keyword.

5.2.4 @EXCLUDE_DUPLICATES

The @EXCLUDE_DUPLICATES keyword skips a stat variable calculation if the source variable was not updated. This is based on the timestamp of source variable being changed from the previous calculation.

This keyword is mutually exclusive with the @ALWAYS_UPDATE_CHANNELS keyword and overrides the default behavior of @ALWAYS_UPDATE_CHANNELS. @EXCLUDE_DUPLICATES takes precedence when used.

5.3 Sample rave Specification

The registered name below requires a corresponding version of rave running with that registered name. A sample command string:

```
rave RA_smoc 11 1000 &
```

```
@REG_NAME
    RA_smoc

  @START_STOP_FIFO_EVENTS

# start event name    stop_event_name     valid_to_compute_label
   profile_start           profile_done          -

  @EXCLUDE_DUPLICATES

# in_variable          rate (FAS/MED/SLO)     result_variable
   dp_oil_pan           SLO                     ra_dp_oilpan.
```

6 Variable Specification with or without @START_STOP_FIFO_EVENTS

6.1 For Continuous Statistical Sampling

6.1.1 Variable Specification when @START_STOP_FIFO_EVENTS is not Present

Each line contains the label of the variable to be sampled, the number of samples to be maintained in the FIFO buffer, the sample rate option, and the names of the variables where the results will be placed. The following is an example of a specification file.

```
#in_variable  samples  rate      result_variable
   TORQUE       200      MED         a_torque
   Dyno_CM      200      MED         a_Dyno_CM
   tvo           30      SLO         a_tvo
   tvc           30      SLO         a_tvc
   some_var     100      my_event    a_some_var
```

Table 6 describes the purpose of each data field.

Table 6: Continuous Statistic Sampling Specification File Data Fields without @START_STOP_FIFO_EVENTS

Data Field	Function
in_variable	The label of a REAL_VARIABLE to be averaged.
result_variable	The label of a STAT_VARIABLE where the statistical result will be placed.
samples	The number of values to be averaged (maximum 4096).
rate	A code or event name for the rate of sampling. Three codes can be used to reference the 3 standard process intervals that are specified in go.scp. FAS is the rate for the first scheduler, MED for the second list, SLO for the 3rd list. These rates are usually: • FAS - 25 hz • MED - 10 hz • SLO - 1 hz Alternatively, the actual name of an event may be used.

6.1.2 Variable Specification when @START_STOP_FIFO_EVENTS is Present

Each line contains the label of the variable to be sampled, the sample rate option, and the names of the variables where the results will be placed. The following is an example of a specification file.

```
#input_variable  rate      result_variable
rail             SLO       ra_rail.
RPM              MED       ra_RPM.
flotron          f_event   ra_flotron.
```

Table 7 describes the purpose of each data field.

Table 7: Continuous Statistic Sampling Specification File Data Fields with @START_STOP_FIFO_EVENTS

Data Field	Function
`in_variable`	The label of a `REAL_VARIABLE` to be averaged.
`result_variable`	The label of a `STAT_VARIABLE` where the statistical result will be placed.
`rate`	A code or event name for the rate of sampling. Three codes can be used to reference the 3 standard process intervals that are specified in `go.scp`. `FAS` is the rate for the first scheduler, `MED` for the second list, `SLO` for the 3rd list. These rates are usually: • `FAS` - 25 hz • `MED` - 10 hz • `SLO` - 1 hz Alternatively, the actual name of an event may be used.

Note:

For the running average process to work properly, the `run_aver` application must be spawned with each process interval for which there is a rate specified in the file.

The running average process will produce normal statistical computations for the specified time window. Obtain the results by using the access extenders listed below for statistical variables.

```
VALUE                          EXTENDER
mean;                           /* AV */
min;                            /* MN */
max;                            /* MX */
range;                          /* RN */
std_deviation;                  /* SD */
coeff_of_variance;              /* CV */
start_time;                     /* ST */
end_time;                       /* EN */
time_of_min;                    /* TN */
time_of_max;                    /* TX */
slope;                          /* C1 */
intercept;                      /* C0 */
initial_value;                  /* IV */
final_value;                    /* FV */
delta;                          /* DL */
integral;                       /* IN */
rsquared;                       /* RS */
std_error;                      /* E0 */
std_error_of_slope;             /* E1 */
sample_rate;                    /* RA */  /* hz */
number_of_samples;              /* NM */
```

For example, for the mean value of the TORQUE for the spec listed in *Section 6.1.1* on page 13, use the label `ra_torque.AV`.

6.2 For Minimum Statistical Sampling

6.2.1 Variable Specification when @START_STOP_FIFO_EVENTS is not Present

Each specification for the variables to average occupies one line. Use the following format when the keyword @START_STOP_FIFO_EVENTS has NOT been specified.

The specifications are read by `rave_specs` and sent to the `rave` task.

Table 8 describes the purpose of each data field.

Table 8: Minimum Statistic Sampling Specification File Data Fields without @START_STOP_FIFO_EVENTS

Data Field	Function
`in_variable`	The label of a `REAL_VARIABLE` to be averaged.
`result_variable`	The label of a `STAT_VARIABLE` where the statistical result will be placed.
`samples`	The number of samples to be averaged (maximum 4096).
`rate`	A code or event name for the rate of sampling. Three codes can be used to reference the 3 standard process intervals that are specified in `go.scp`. `FAS` is the rate for the first scheduler, `MED` for the second list, `SLO` for the 3rd list. These rates are usually: • `FAS` - 25 hz • `MED` - 10 hz • `SLO` - 1 hz Alternatively, the actual name of an event may be used.

6.2.2 Variable Specification when @START_STOP_FIFO_EVENTS is Present

Each line contains the label of the variable to be sampled, the sample rate option, and the names of the variables where the results will be placed.

Table 9 describes the purpose of each data field.

Table 9: Minimum Statistic Sampling Specification File Data Fields with @START_STOP_FIFO_EVENTS

Data Field	Function
`in_variable`	The label of a `REAL_VARIABLE` which is to be averaged.
`result_variable`	The label of a `STAT_VARIABLE` where the statistical result will be placed.

Data Field	Function
`rate`	A code or event name for the rate of sampling. Three codes can be used to reference the 3 standard process intervals that are specified in `go.scp`. FAS is the rate for the first scheduler, `MED` for the second list, `SLO` for the 3rd list. These rates are usually: • FAS - 25 hz • `MED` - 10 hz • `SLO` - 1 hz Alternatively, the actual name of an event may be used.

Note:

For the running average process to work properly, the `rave` application must be spawned with each process interval for which there is a rate specified in the file.

The running average process will produce normal statistical computations for the specified time window. These include mean (AV) and number of samples (NM). Obtain the results by using the access extenders listed below for statistical variables.

```
VALUE                          EXTENDER
mean;                           /* AV */
number_of_samples;              /* NM */
```

7 Releasing Statistical Sampling Applications

Each of the statistical sampling applications can be terminated by a `release` command. This is a clean way of stopping the applications because the application will remove all of the statistical variables that were "owned" by the application. A `slay` command will also terminate the application, but the variables will remain in the system which could create some problems later if another application wanted to create variables using those same names. The `release` command sends a message to all the applications. Refer to the following usage help on cyflex.com:

- `release_run_aver`
- `release_rave`
- `release_snap_stats`
- `release_statistics`