CyFlex® Logging Data using dlogger User Guide

Version 8

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 Updated Output Files example in Section 2.2.1 Updated Section 3.2 Specification File Keywords: • Corrected the examples for keywords `@OUTPUT_PATH` and `@FIFO_LOG_BUFFER` • Added keyword `@META_DATA`
3	4/2/2020	Retrofit to new template
4	4/21/2020	Added descriptions of keywords `@ECM_LIST` and `@SCAN_LIST_AUX_USER` to *Section 4.2 Specification File Keywords* on page 40
5	10/28/2020	Added *Section 2 Using dloggereditor* on page 2
6	9/7/2021	Added hypertext linked cross-references to mentioned CyFlex manuals Removed table that listed `dlogger` options in Section 3.4 Command Options on page 39 and added hypertext linked cross-reference to its usage help on cylflex.com.
7	5/24/2022	Updated all hypertext linked cross-references to cyflex.com usage help descriptions
8	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

LIST OF FIGURES

LIST OF TABLES

1 Overview

The `dlogger` program collects and logs test results on the CyFlex test system for storage and use in the DARTS system. This program is one of many software tools included with CyFlex.

This document describes the `dlogger` program and how to use it.

1.1 DARTS

Once dlogger collects and logs the test results, CyFlex transfers the data to the DARTS system using a CyFlex external data manager service specific to that transfer. The DARTS system provides data storage and analysis.

1.2 Data Sampling

The `dlogger` program samples and logs the test results according to the user specified setup.

The data sampling rate is defined in the specification (spec) file. However, external events and even logical variables can be used to start and stop data sampling. As an alternative to time-based logging, a named event can also be used to cause sampling.

The `dlogger` program can log up to 384 channels of data at rates up to 500 samples per second. Additionally, dlogger can log any real, integer, logical or string variable. It can also log any member of a statistical, composition, property, or emission variable. For an explanation of variable types, refer to *Creating User Computations and User Variables*.

Note:

In order for a variable value to be logged, the dlogger specification file must include a DARTS (PAM) keyword as shown in *Section 4.2 Specification File Keywords* on page 40.

1.3 Events

Users familiar with the software tools in CyFlex probably recognize the term events, which the test system uses to communicate between processes. For example, the Test Manager application relies on various events to automate and control testing. Events can tell Test Manager when to transition between modes and execute procedures.

Events can be used to start and stop dlogger data sampling and control the sampling rate. The `dlogger` program can detect an event if specified by a keyword(s) in the associated `dlogger` specification file. If the event does not already exist, `dlogger` will create it. `dlogger` attaches to the event which means that it will take action when the event is set by another process. When the event occurs, `dlogger` executes an action associated with the event per the spec file. See *Section 4.2 Specification File Keywords* on page 40 for additional information.

Other CyFlex programs, such as Test Manager, can create events. Additionally, the user can create events to control dlogger using the commands shown in *Section 3.4 Command Options* on page 39. For more about events and creating them, refer to *CyFlex Events*.

2 Using dloggereditor

Use the **Dlogger Specification Editor** interface to create and edit a dlogger spec file. Enter `dloggerditor` at the command prompt to start the interface. Refer to usage help for `dloggerditor` on cyflex.com for related information.

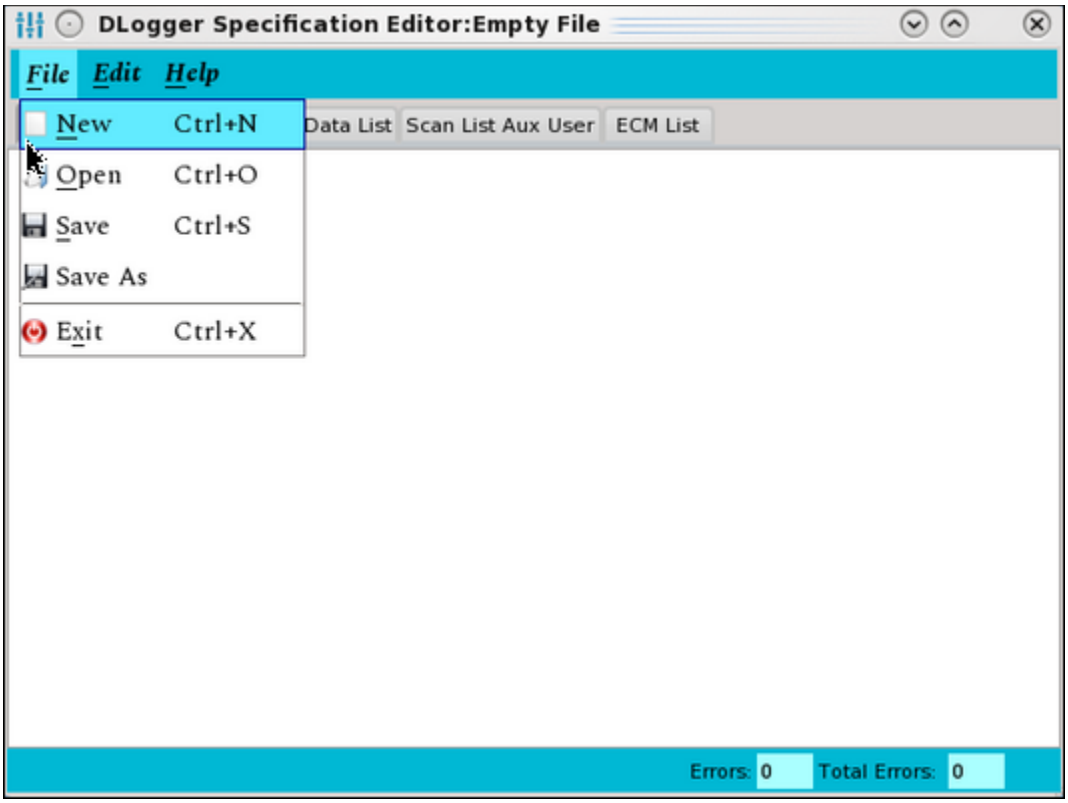
2.1 General Actions and Information

2.1.1 Creating a New dlogger Specs File

Execute the following steps to create a new file:

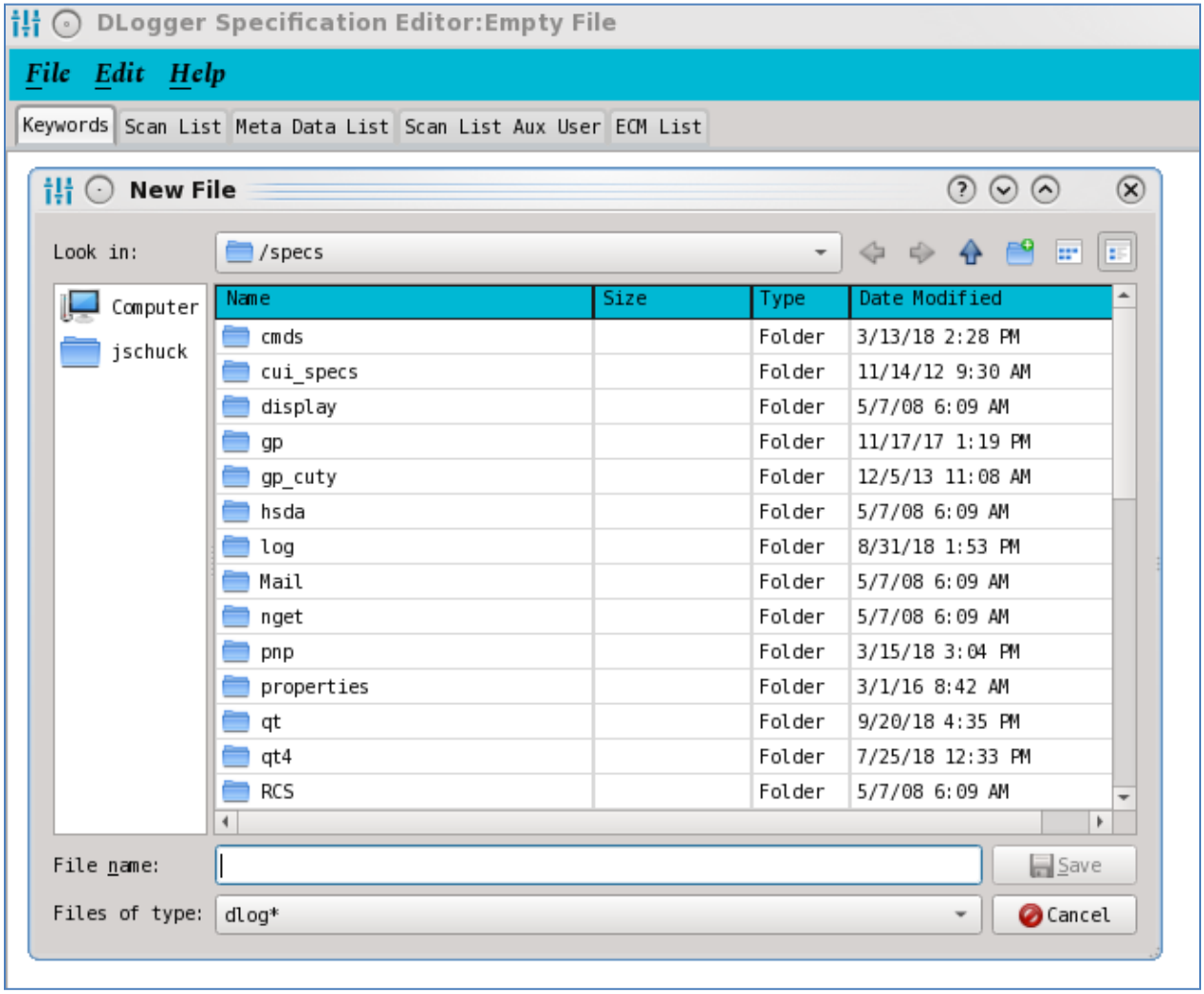
1. Select **File** – **New** from the menu bar as in *Figure 1*.

Figure 1: New-File Selection



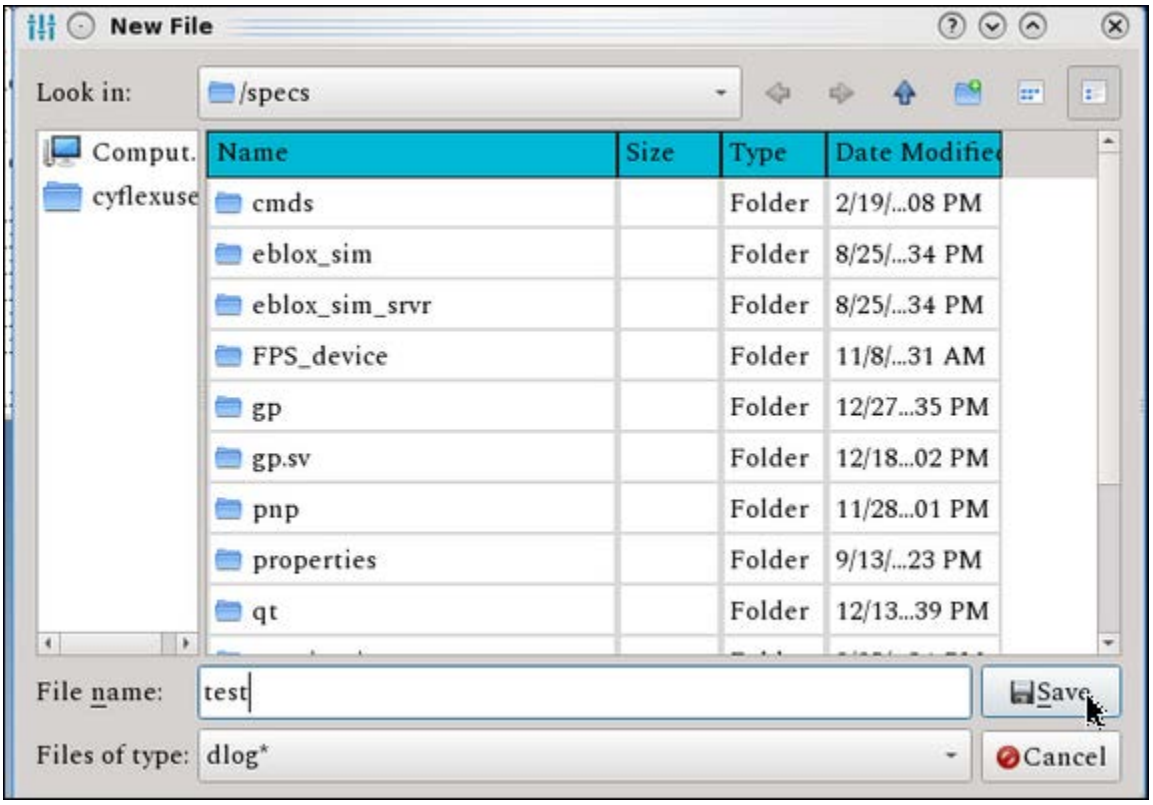
2. The **New File** screen is displayed as in *Figure 2*.

Figure 2: New File Screen



3. Specify the location and name of the dlogger specification file to create in the **File name:** field as in *Figure 3*.

Figure 3: Populated New File name Field



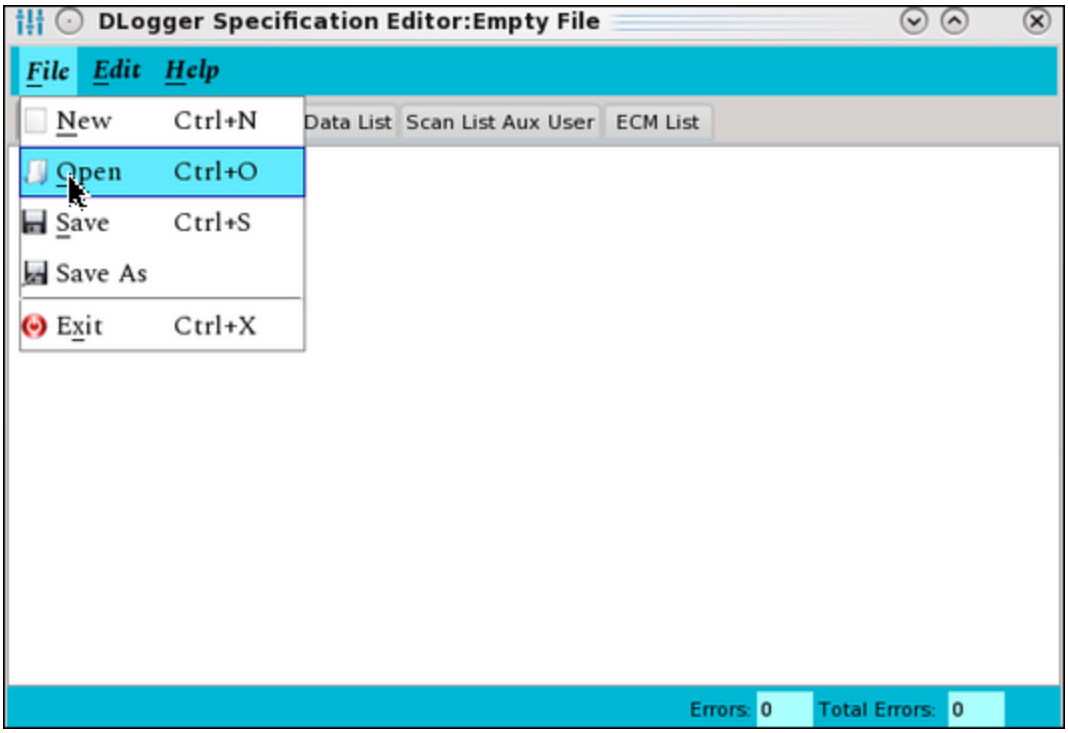
4. Select **Save** to create the file as in *Figure 3*.

2.1.2 Loading an Existing dlogger Specs File

Execute the following steps to load an existing file:

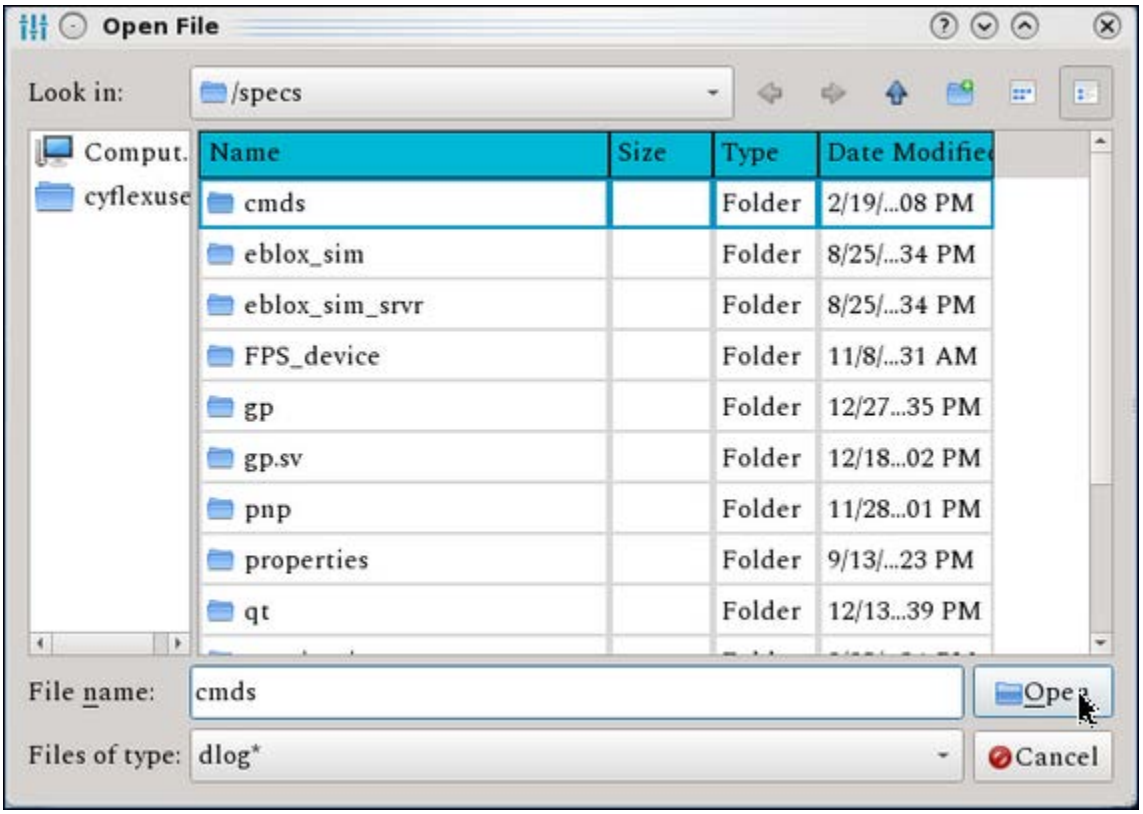
1. Select **File** – **Open** from the menu bar as in *Figure 4*.

Figure 4: Open File Selection



2. Specify the location and name of the dlogger specification file to open in the **File name:** field as in *Figure 5*.

Figure 5: Populated Open File name Field



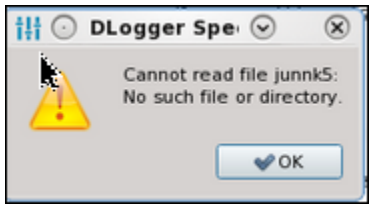
3. Select **Open** to display the file as in *Figure 5*.

2.1.2.1 *Loading a File in the Command Environment*

Enter `dloggereditor` *`[spec_filename]`* & at the command line. Refer to cyflex.com usage help for `dloggereditor`.

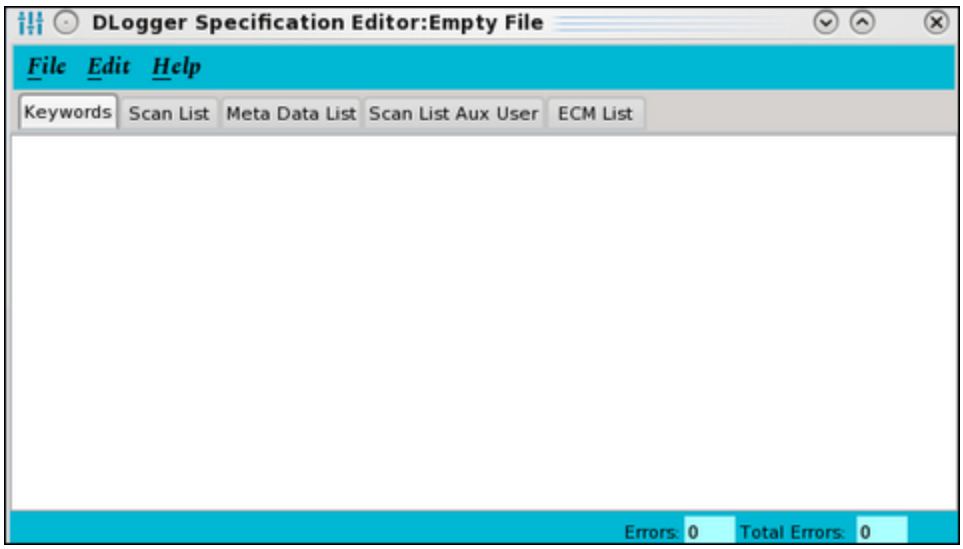
A pop-up message as in *Figure 6* is displayed if an invalid file is specified.

Figure 6: Invalid File Specification Message



Click **OK** and the **Empty File** screen will display as in *Figure* 7.

Figure 7: Empty File

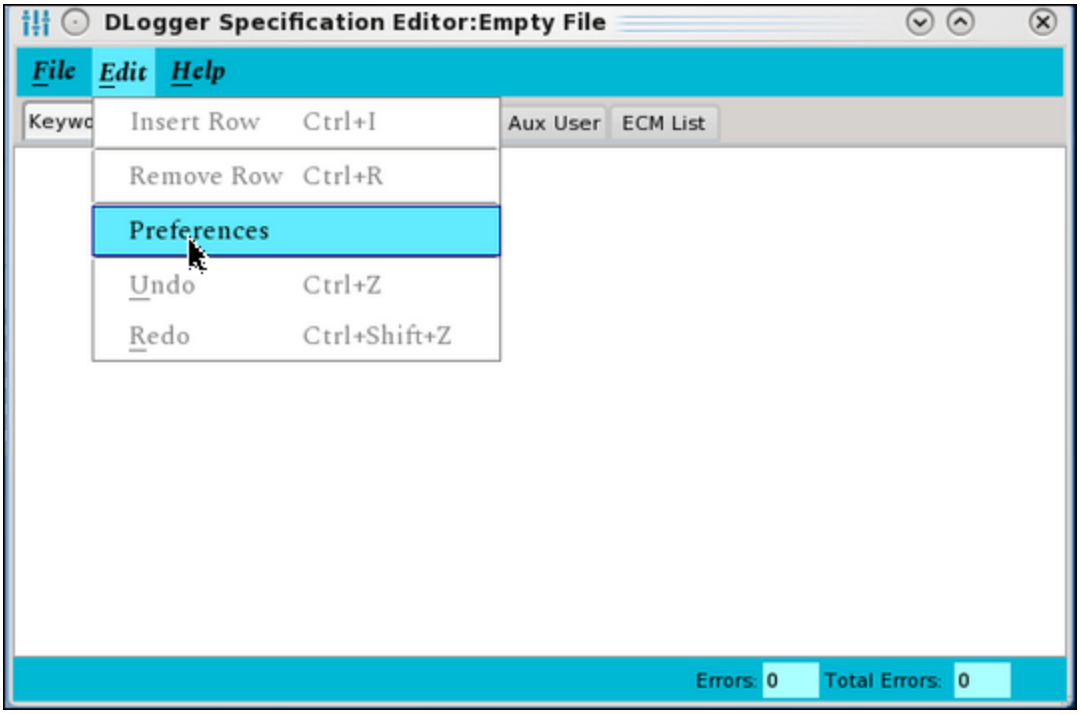


2.1.3 Setting Font Preferences

Execute the following steps to set font preferences:

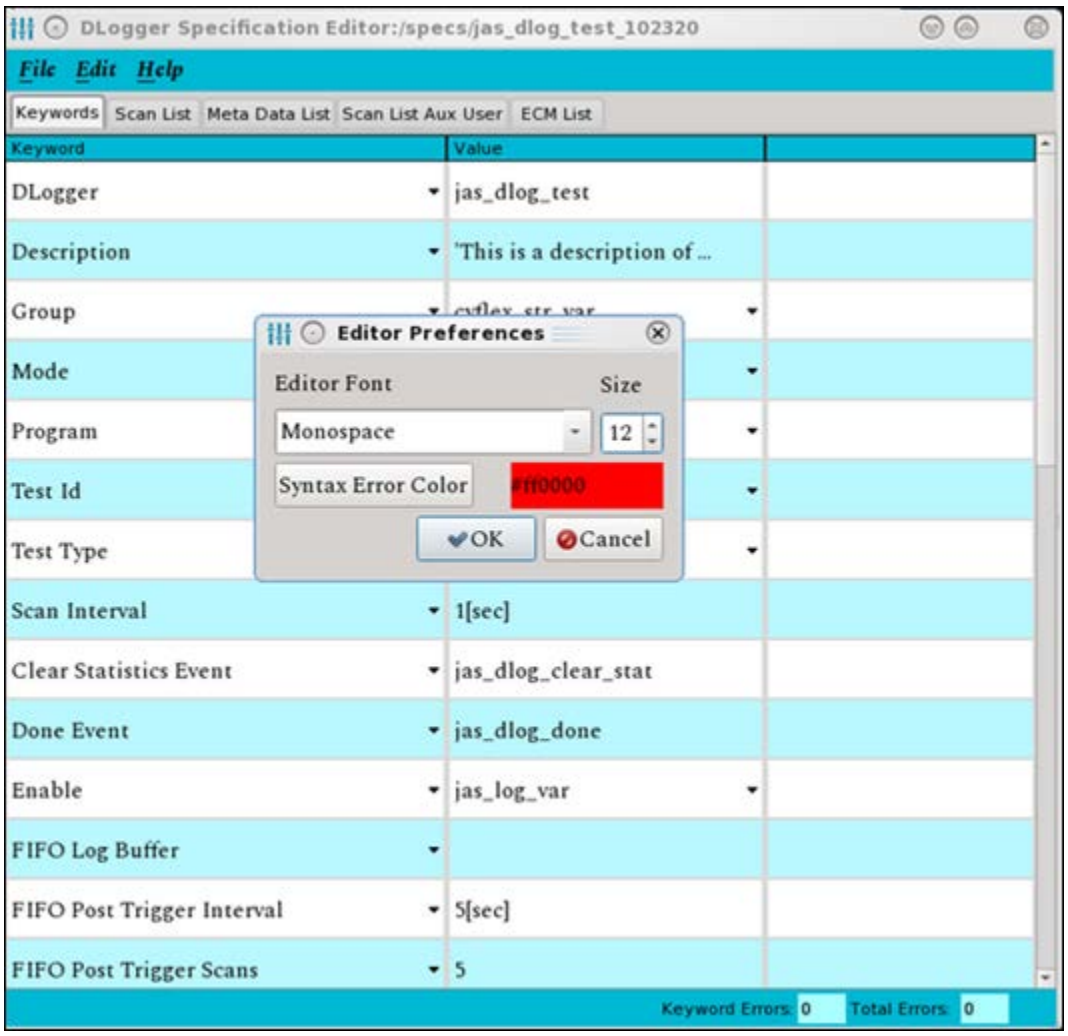
1. Select **Edit** - **Preferences** from the menu bar as in *Figure 8*.

Figure 8: Edit - Preferences Selection



2. The **Editor Preferences** dialog is displayed as in *Figure 9*.

Figure 9: Editor Preferences Dialog



3. Select preferences:
   a. **Editor Font**: Click the drop-down and select the desired font.
   b. **Size**: Click the up/down arrow heads to scroll and select a font size.
   c. Select **OK** to incorporate selections.

2.1.4 Hovering the Mouse to Display Information

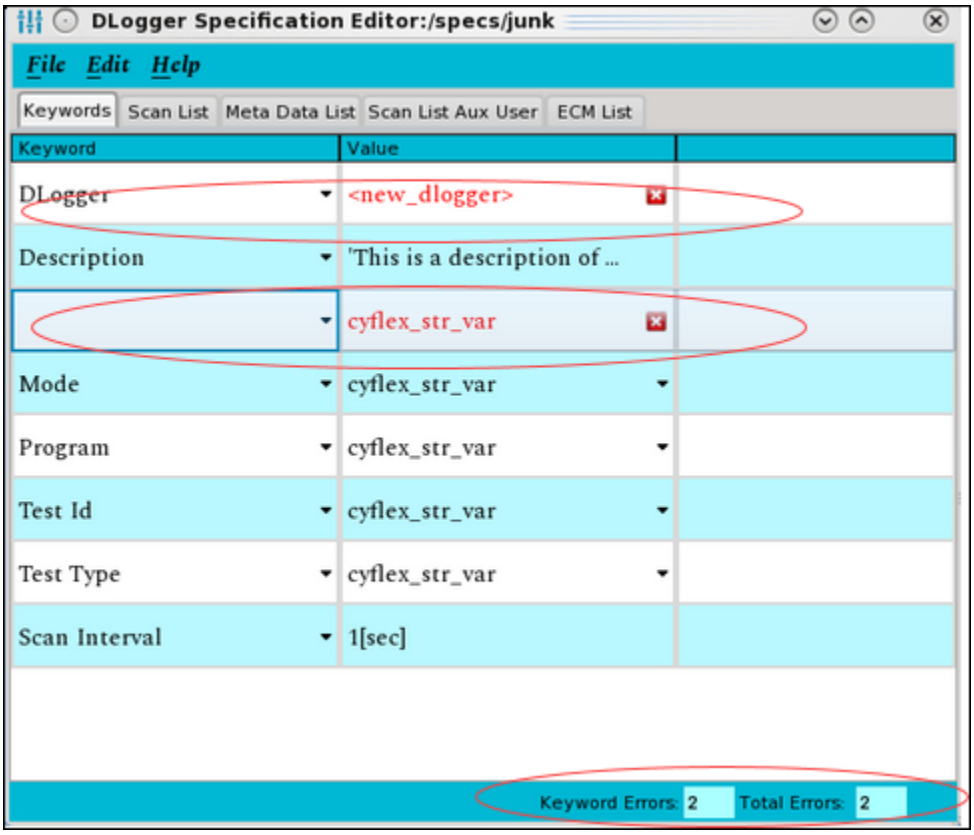
Hover the mouse over data in a tab screen column or row entry to display a description of that file element's usage.

Hovering the mouse also displays error information for a file element. See *Section 2.1.5 Error Counts* on page 10 for related information.

2.1.5 Error Counts

The bottom row of each tab screen lists the number of errors in the currently displayed tab and total number of errors in the file as in *Figure 10*. Errors in the currently displayed tab screen are indicated by red text accompanied by a red'"x". Errors may be syntax errors, excluded or empty fields, or any other file element that fails checking.

Figure 10: Error Counts Example



2.2 Tab Editing Actions

2.2.1 Keywords Tab

Select the **Keywords** tab to display `dlogger` spec file keywords. When a new dlogger spec file is created, default values appear within the screen for several of the required spec file keywords as in *Figure 11.*

Figure 11: Keywords Tab Screen

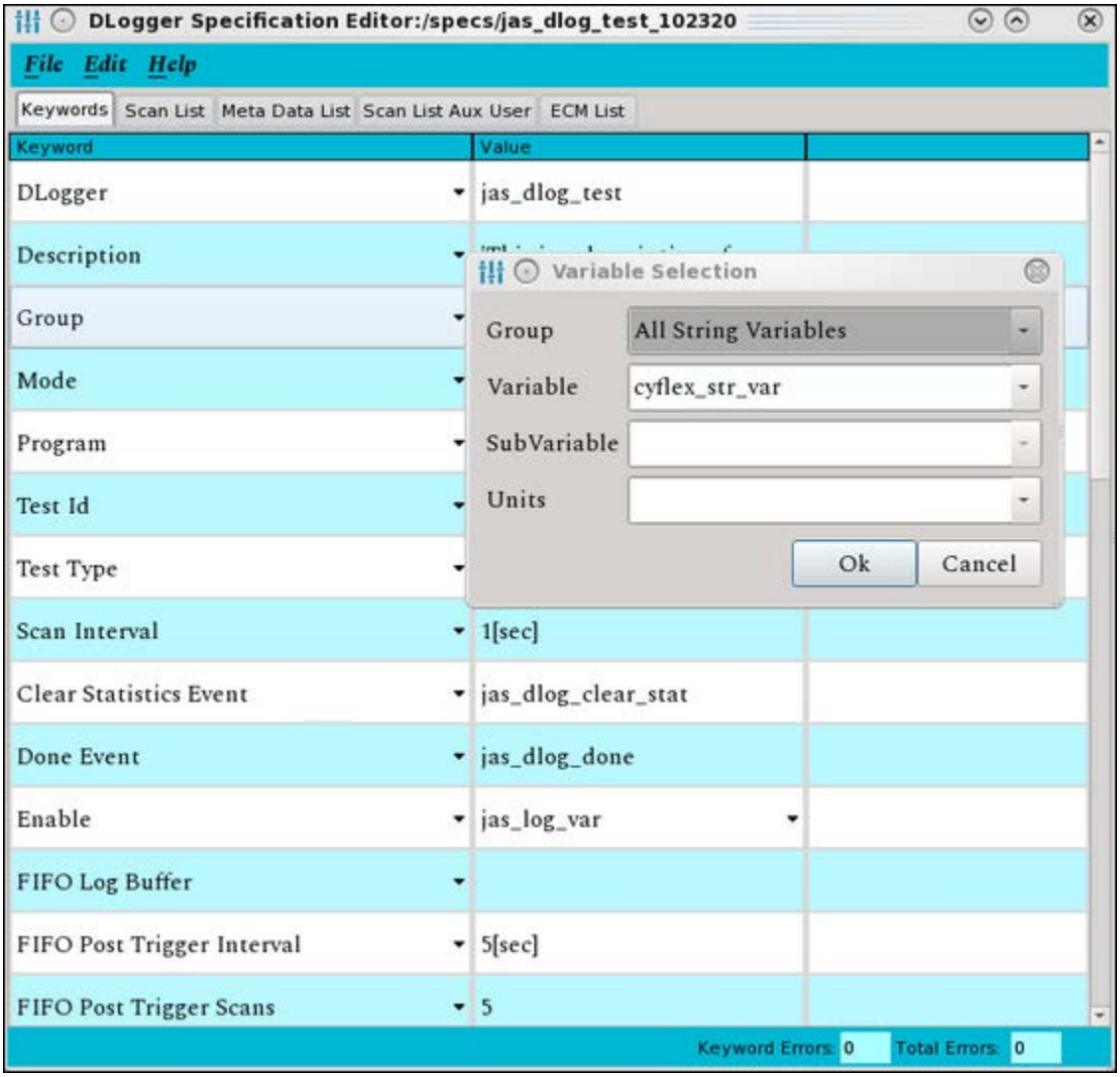
Refer to the following sub-sections for descriptions of **Keywords** tab screen functions.

2.2.1.1 *Editing Value Column Information*

Execute the following steps to edit applicable **Value** column information:

1. Double-click the **Value** column of a row to edit.
2. The **Variable Selection** dialog is displayed as in *Figure 12*.

Figure 12: Variable Selection Dialog



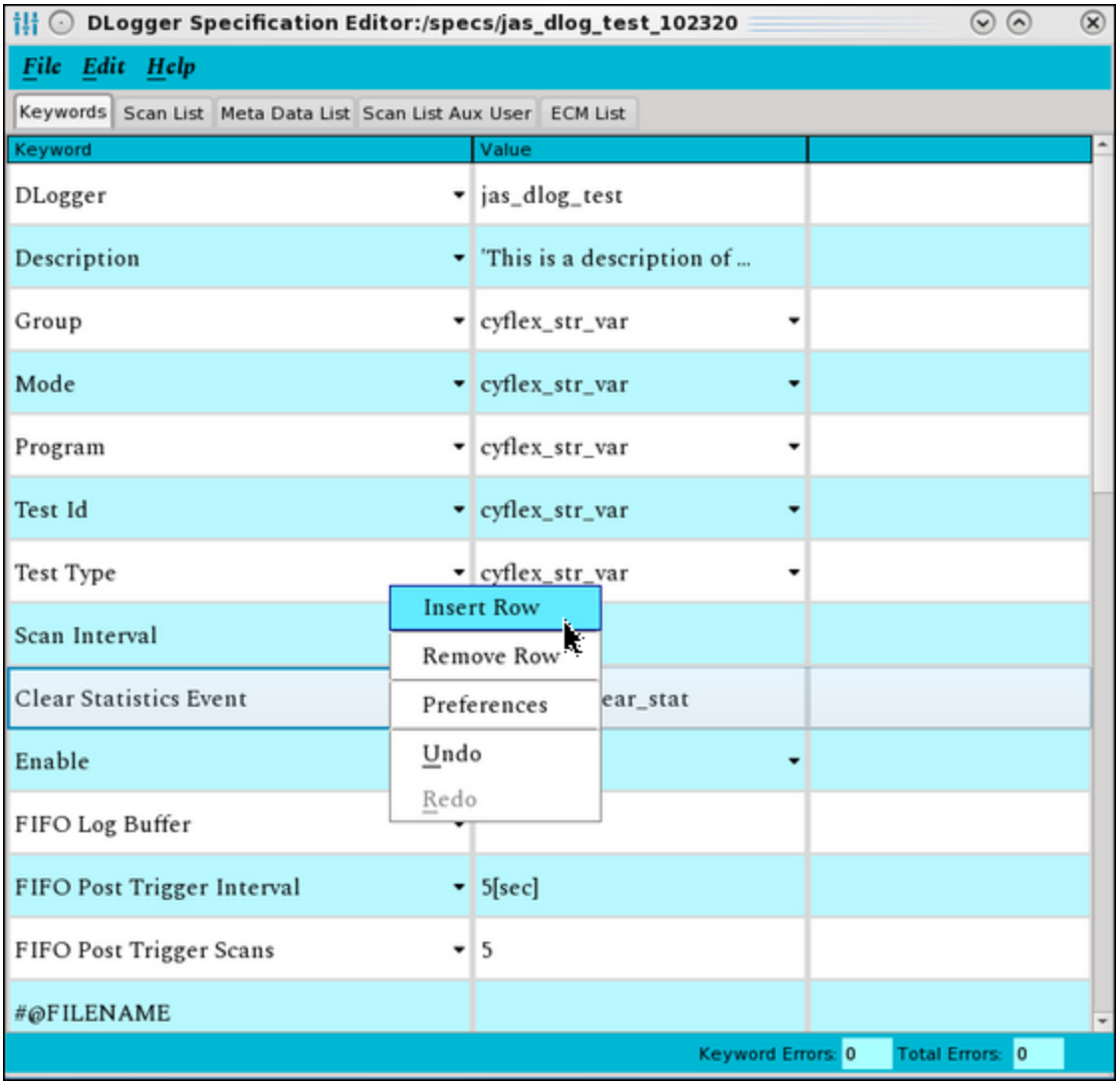
3. Click the drop-downs to select new variable names or enter the information in the appropriate input fields.
4. Select **OK** to incorporate changes.

2.2.1.2 *Inserting Additional Keywords*

Execute the following steps to insert additional keywords into a `dlogger` spec file:

1. Right-click an existing **Keywords** tab row and select **Insert Row** on the resulting pop-up menu as in *Figure 13*. A new row is inserted below the selected row.

Figure 13: Insert Keyword Row



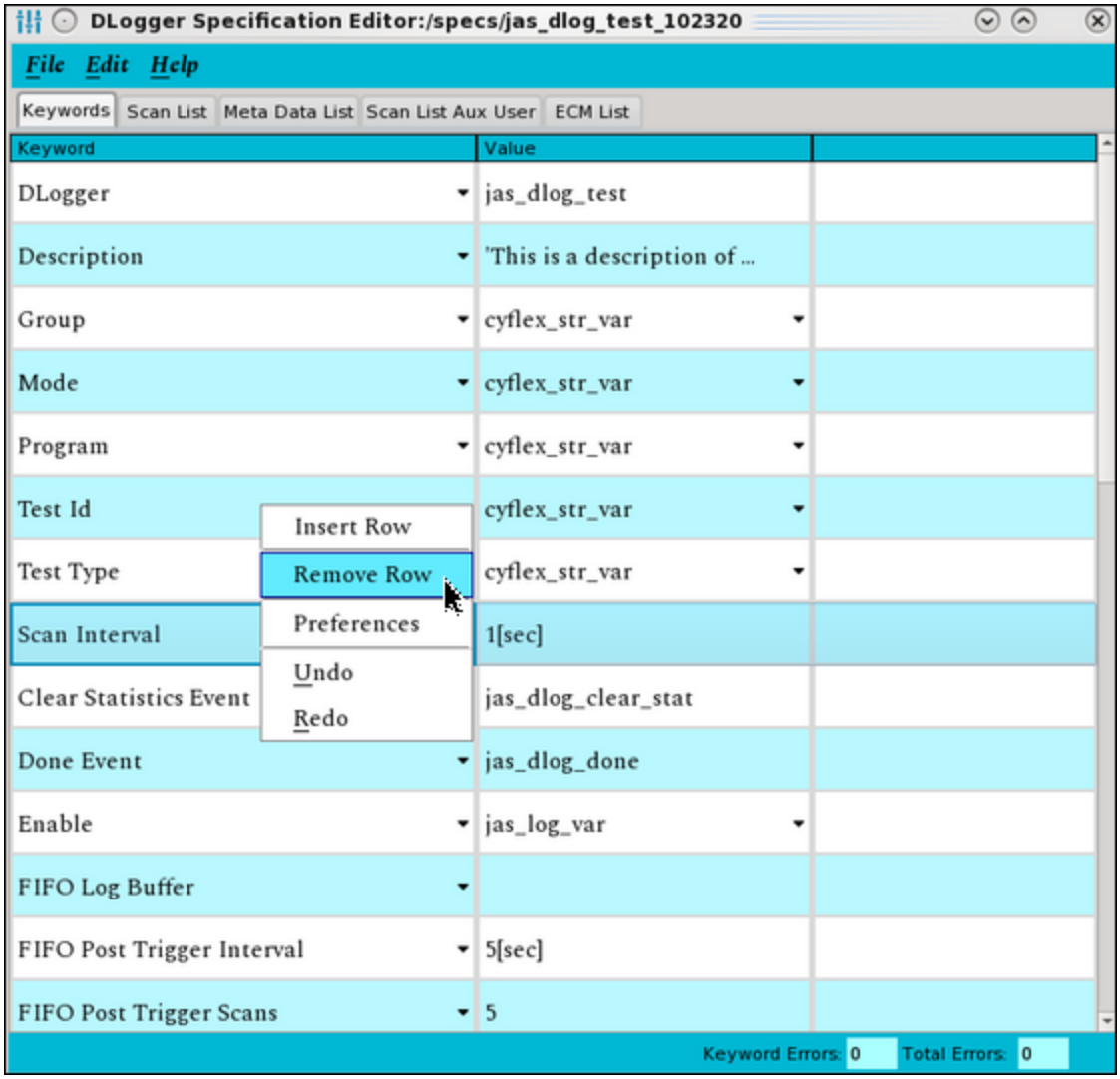
2. Select the drop-down list within the **Keyword** column of the newly inserted row and select a new keyword to add to the specification file as in *Figure 14* on page 14.

Figure 14: Add New Keyword

2.2.1.3 *Deleting Keywords from the Spec File*

Right-click over the row to remove on the **Keywords** tab and select **Remove Row** as in *Figure 15*.

Figure 15: Remove a Keyword Row



2.2.1.4 *Keywords that can be Added to the Spec File via the Keywords Tab*

Table 1 lists the keywords that can be added via the **Keywords** tab.

Table 1: Keywords that can be Added via the Keywords Tab

Generated Keyword In Spec File	Definition	Type
`*@DLOGGER`	Identifies this is a dlogger spec file	String
`*@DESCRIPTION`	A title to be written to the output file	Computed Expression
`*@GROUP`	Measurement name in meta-data section of output file for `GROUP`	String Variable
`*@MODE`	Test mode to include in the output file meta-data section for `MODE`	String Variable
`*@PROGRAM`	Program name to include in the output file meta-data section for `PROGRAM`	String Variable
`*@TEST_ID`	Test Id to include in the output file meta-data section for `TEST ID`	String Variable
`*@TEST_TYPE`	Test Name to include int the output file meta-data section for `TEST TYPE`	String Variable
`*@SCAN_INTERVAL`	Time between data samples in output file	Literal value (<time>[units]) or Variable name
`@CLEAR_STATISTICS_EVENT`	Event to trigger statistical buffers within dlogger to reset to 0	Event Name - String
`@DONE_EVENT`	Name of event set at completion of data collection	Event Name - String
`@ENABLE`	Logical variable that must be `TRUE` before logging can start	Logical Variable

Generated Keyword In Spec File	Definition	Type
@FIFO_LOG_BUFFER	Presence of keyword signifies to activate First-In First-Out Logging	
@FIFO_POST_TRIGGER_INTERVAL	Length of time to obtain scans after the FIFO trigger event (stop or release event) has been received	
@FIFO_POST_TRIGGER_SCANS	Number of scans to obtain after the FIFO trigger event (stop or release event) has been received	Integer value
@FILENAME	Enables a computed expression to be entered after the keyword. The expression is evaluated during each scan to see if the result has changed. If it has, then the present file is closed and transferred to DARTS, and a new file is opened. It does not affect the name of the output file, which is a fixed format that includes the date, time, and specification file name.	Computed Expression
@FORCE_DIRECT_FILE_WRITE	Indicates that data should be written directly to the output file when high data rates are used	Yes or No
@FTP_EVENT	Event that triggers the output file to be finalized, and initiates file transfer	Event Name - String
@GET_NEW_SCAN_INTERVAL	Event that triggers a re-evaluation of the SCAN_INTERVAL computed expression	Event Name – String

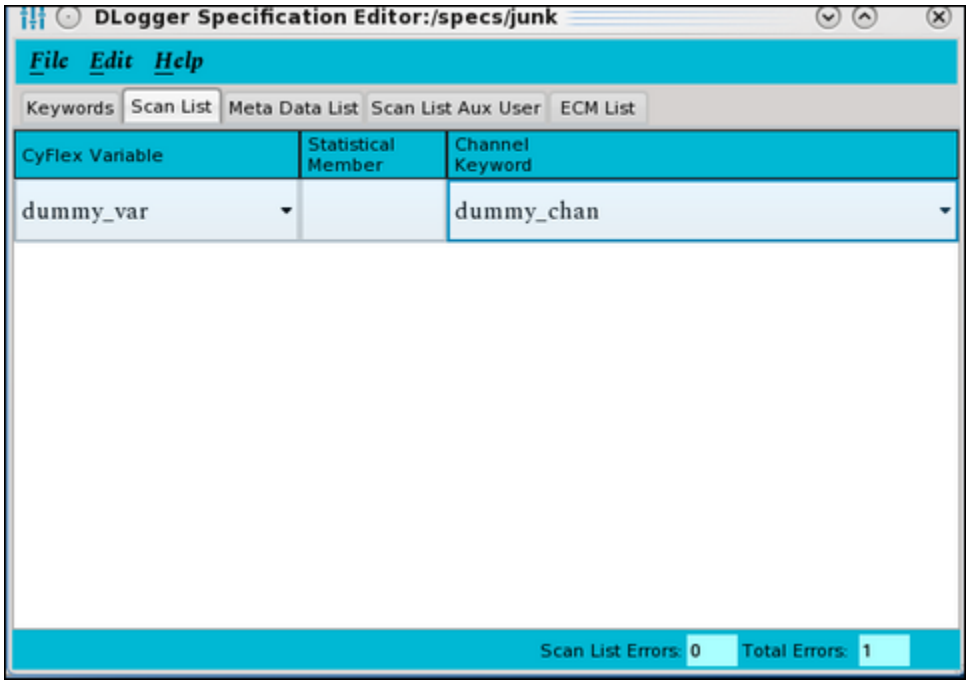
Generated Keyword In Spec File	Definition	Type
`@LOG_DIGITAL_DESCRIPTION`	Flag used to determine if logical variable descriptions for all logical variables should be logged instead of 0 or 1	Yes or No
`@LOG_STATISTICS`	Flag used to specify that statistics should be computed for the variables specified via the `@SCAN_LIST` keyword	Yes or No
`@LOGGING_ACTIVE_LABEL`	The name of a CyFlex logical variable that indicates dlogger is actively collecting data and logging it.	Logical Variable
`@MAX_SCANS`	Maximum number of samples in a sampling session	Integer value
`@MAX_STATISTICAL_SCANS`	Maximum number of scans when the `@LOG_STATISTICS` keyword is specified	Integer value
`@OUTPUT_PATH`	Directory path to the output file	String variables or Directory Path - String
`@DARTS_STEADY_STATE`	Presence of keyword signifies the `darts_ss_specs` file variables should be included in the scan list	
`@READ_SPEC_FILE_EVENT`	Event that triggers the dlogger to re-read the spec file	Event Name - String
`@REG_NAME`	Name that identifies the instance of `dlogger` within the OS	String

Generated Keyword In Spec File	Definition	Type
`@RELEASE_EVENT`	Event that signals the end of a sampling interval and terminates the dlogger task after the data files are written	Event Name - String
`@RUNNING_AVERAGE`	Window width of a running average window in units of time and the event that causes the data to be logged	**Note:** The second and third column are used to enter information for this keyword. The Value column should contain the `window_width[units]`. The right most column should contain the name of the event that causes the averages to be written to the output file.
`@SEND_ON_PATH_CHANGE`	By default, the existing file is closed, transferred to DARTS, and a new file is opened when one of the following values change: • `Mode` • `Test ID` • `Test type` • `Program` • `Group` Select **No** from the drop-down to disable the default behavior.	Yes or No
`@START_EVENT`	Event that triggers the start of a sampling interval	Event Name - String
`@STOP_EVENT`	Event that triggers the end of a sampling interval	Event Name - String
`@SYNC_EVENT`	Event that triggers a scan of all channels	Event Name - String
* - Denotes required keyword		

2.2.2 Scan List Tab

Select the **Scan List** tab to display CyFlex Variables, associated Statistical Members, and Channel Keywords. When a new dlogger spec file is created by `dloggereditor`, a default dummy variable row is added to the **Scan List** Tab as in *Figure 16*.

Figure 16: Scan List Dummy Row

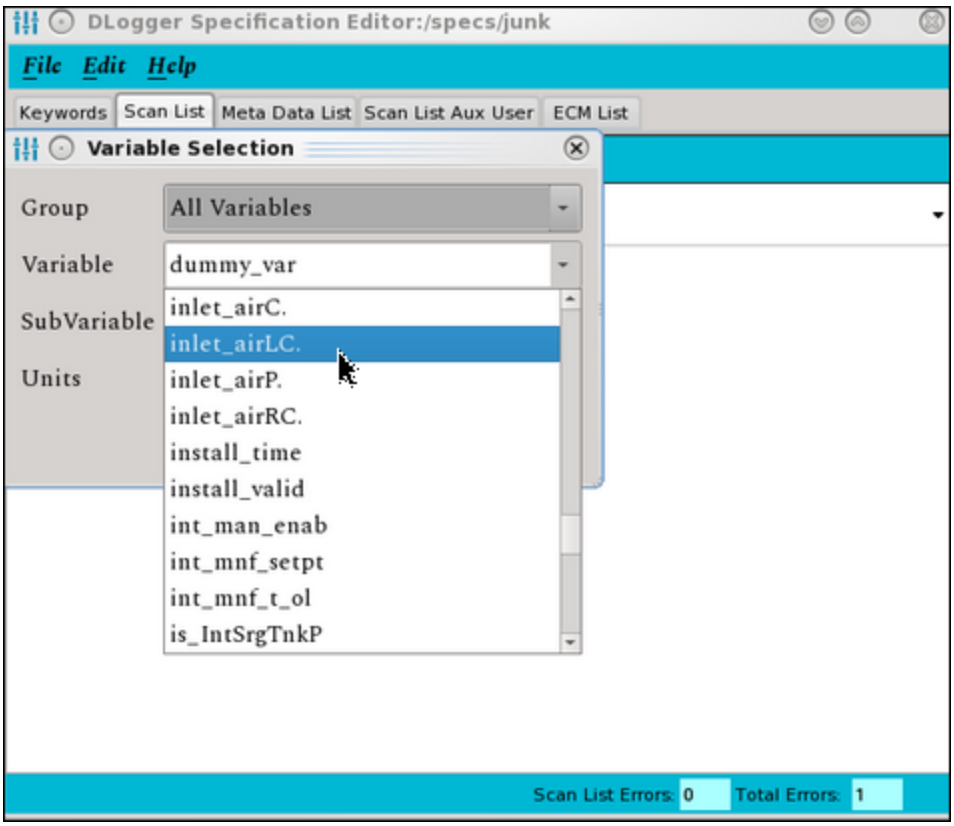


This dummy row is available to edit and add rows to the **Scan List**; refer to *Figure 17* on page 21.

Execute the following steps to edit a **Scan List** dummy variable row:

1. Double-click the **CyFlex Variable** column to display the **Variable Selection** dialog as in *Figure 17*.

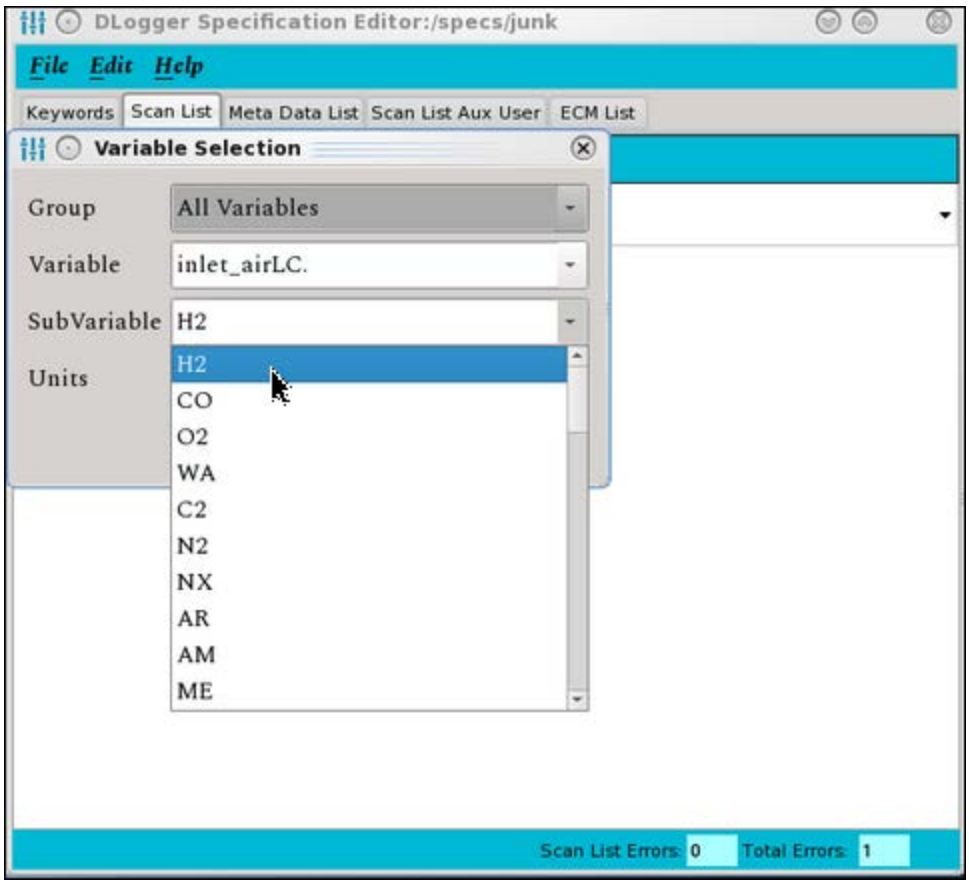
Figure 17: Scan List Variable Selection Dialog



2. In the **Variable Selection** dialog, select the type of variable to log from the **Group** drop-down as in *Figure 17* above.

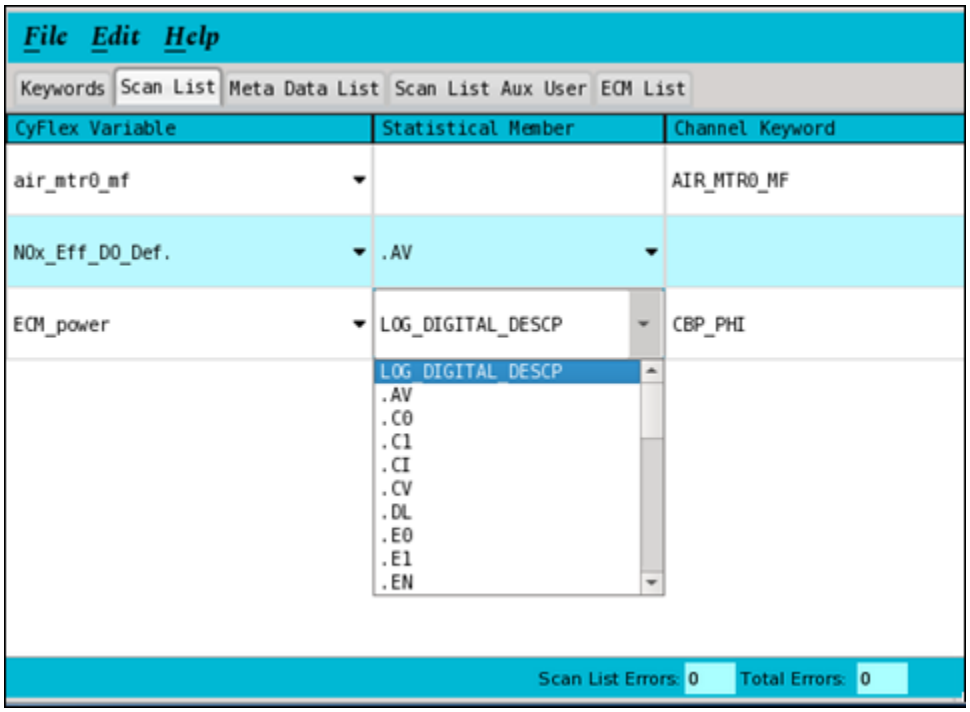
3. Select the variable name to log from the **Variable** drop-down in *Figure 17* on page 21.
    - If a Statistical **Variable** type variable name is selected, click the drop-down to select the **SubVariable** statistical member to be logged as in *Figure 18*.

Figure 18: Scan List SubVariable Selection



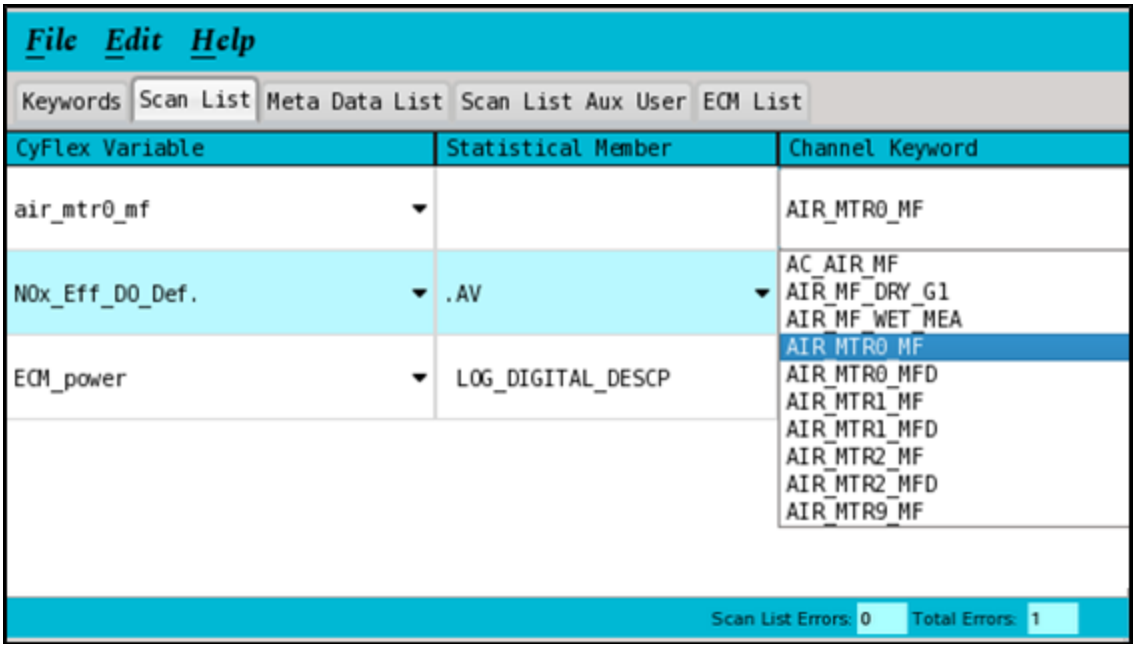
- If a Logical **Variable** type variable name is selected, a choice between logging the logical variable true and false state descriptions or 0 or 1 can be made. To log the logical variable true and false state descriptions, select `LOG_DIGITAL_DESC` within the **Statistical Variable** column for the selected variable as in *Figure 19*.

Figure 19: LOG_DIGITAL_DESC Selection



4. Select the PAM keyword to associate with the selected variable name. Double-click the **Channel Keyword** column and select the corresponding PAM keyword from the drop-down as in *Figure 20*.

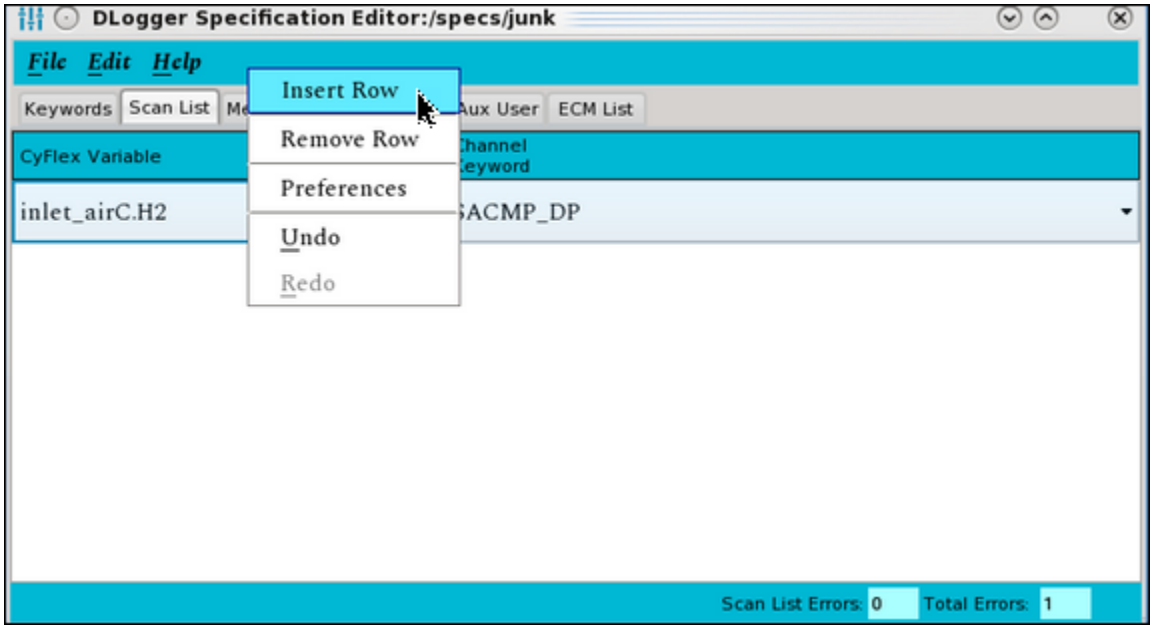
Figure 20: Scan List Select PAM Keyword



2.2.2.1 *Inserting Additional Scan List Rows*

Right-click a table row within the **Scan List** tab and select **Insert Row** from the pop-up menu as in *Figure 21*.

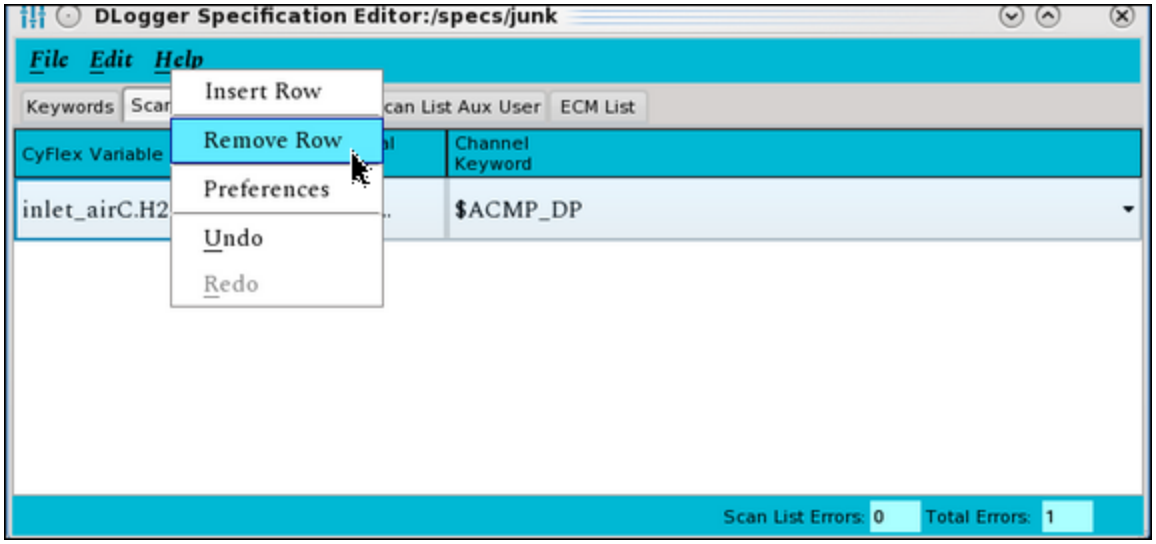
Figure 21: Insert Scan List Row



2.2.2.2 *Removing a Scan List Row*

Right-click a table row within the **Scan List** tab and select **Remove Row** from the pop-up menu as in *Figure 22*.

Figure 22: Remove Scan List Row



2.2.2.3 *Keywords that can be Added to the Spec File via the Scan List Tab*

Table 2 lists the keyword that can be added via the **Scan List** tab.

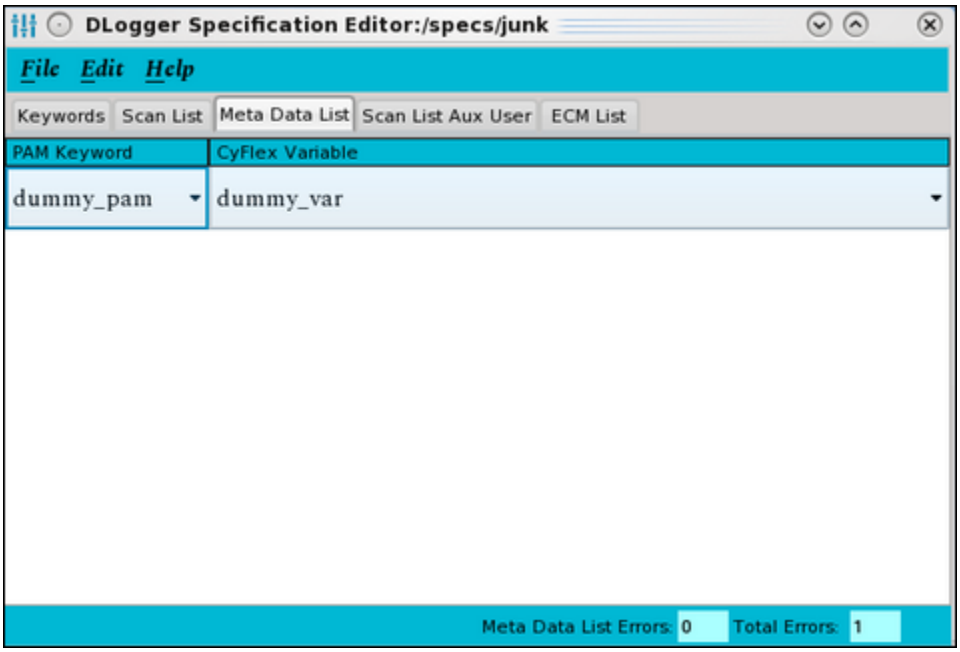
Table 2: Keywords that can be Added via the Scan List Tab

Generated Keyword In Spec File	Definition
`*@SCAN_LIST`	PAM keyword names correlating to CyFlex Variable Names to be logged
* - Denotes required keyword	

2.2.3 Meta Data List Tab

Select the **Meta Data List** tab to display the meta data list of PAM keywords and associated CyFlex variables. When a new dlogger spec file is created by `dloggereditor`, a default dummy variable row is added to the **Meta Data List** Tab as in *Figure 23*.

Figure 23: Meta Data List Dummy Row



This dummy row is available to edit and get additional rows added to the **Meta Data List**; refer to *Figure 24*.

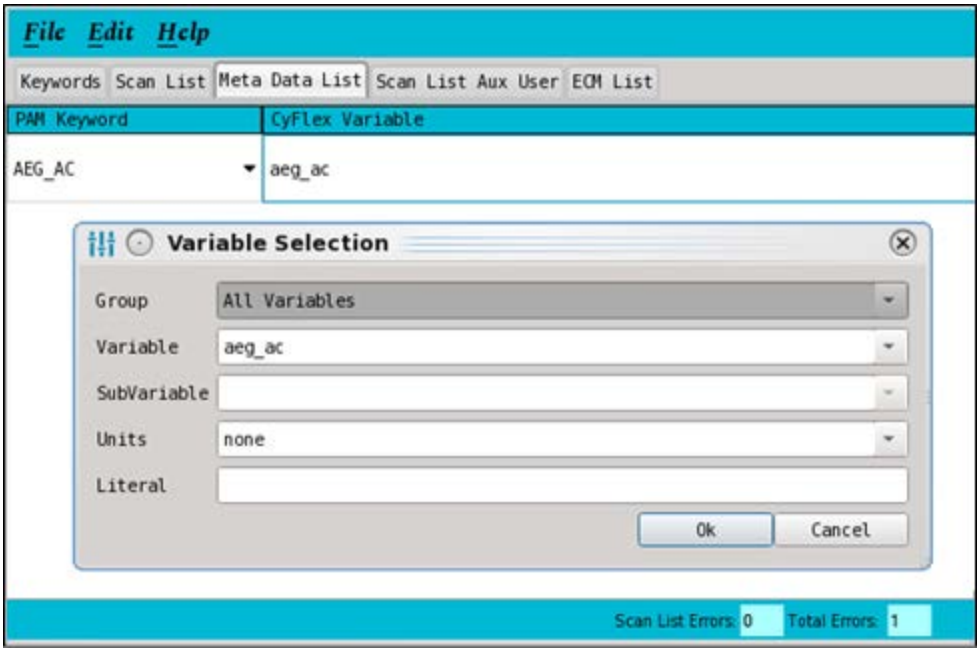
Execute the following steps to edit a default **Meta Data List** dummy variable row:

1. Double-click the **PAM Keyword** column to display the **PAM Keyword** drop-down list as in *Figure 24*.

Figure 24: PAM Keywords List

2. Select a corresponding **CyFlex Variable** for the selected **PAM Keyword**. Double-click within the **CyFlex Variable** column to display the **Variable Selection** dialog. Select the variable name from the **Variable** drop-down as in *Figure 25*.

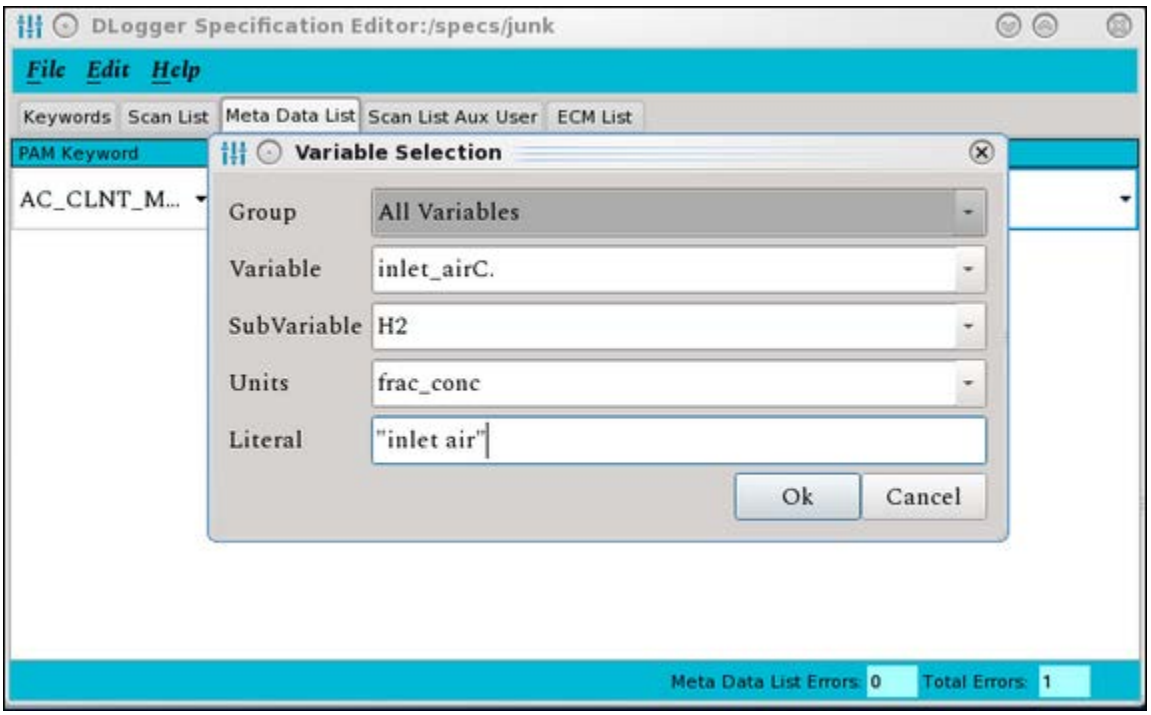
Figure 25: Meta Data List Variable Selection – Add Corresponding Variable



3. Select **OK** to incorporate the change.
4. Enable a string literal value written to the output file for a selected PAM Keyword. Double-click the **CyFlex Variable** column to display the **Variable Selection** dialog.

5. Enter the desired string enclosed by quotes in the **Literal** input field as in *Figure 26*.

Figure 26: Meta Data List Variable Selection – Add Literal Value

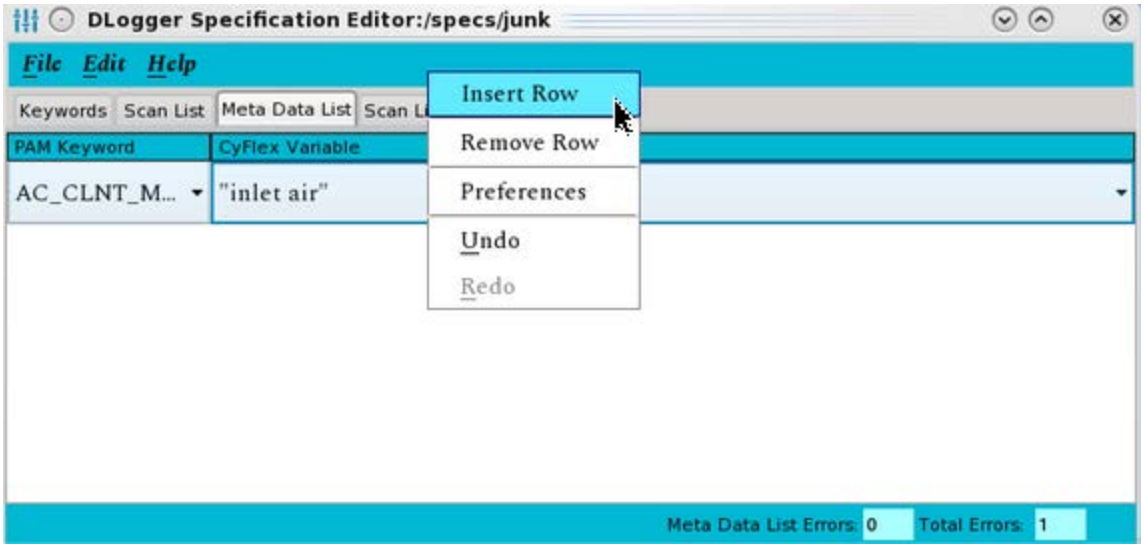


6. Select **OK** to incorporate the change.

2.2.3.1 *Inserting Additional Meta Data List Rows*

Right-click a table row within the **Meta Data List** tab and select **Insert Row** from the pop-up menu as in *Figure 27*.

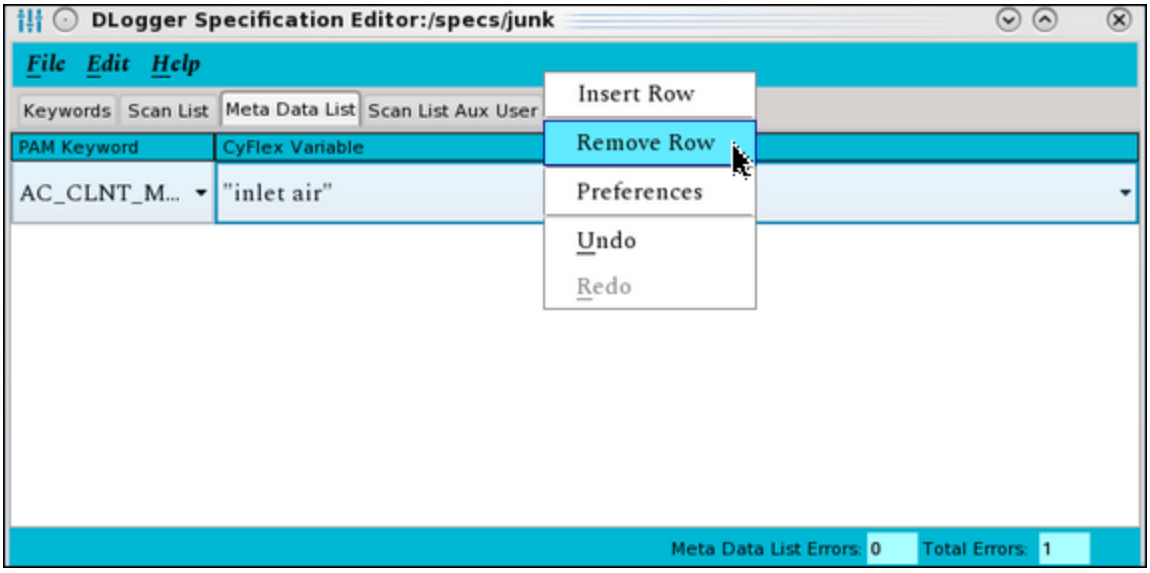
Figure 27: Insert Meta Data List Row



2.2.3.2 *Removing a Meta Data List Row*

Right-click a table row within the **Meta Data List** tab and select **Remove Row** from the pop-up menu as in *Figure 28*.

Figure 28: Remove Meta Data List Row



2.2.3.3 *Keywords that can be Added to the Spec File via the Meta Data List Tab*

Table 3 lists the keyword that can be added via the **Meta Data List** tab.

Table 3: Keywords that can be Added via the Meta Data List Tab

Generated Keyword In Spec File	Definition
* `@META_DATA`	List of specified PAM Keywords and corresponding CyFlex variable names where the CyFlex variable values will be captured once when the dlogger output file is written.
* - Denotes required keyword	

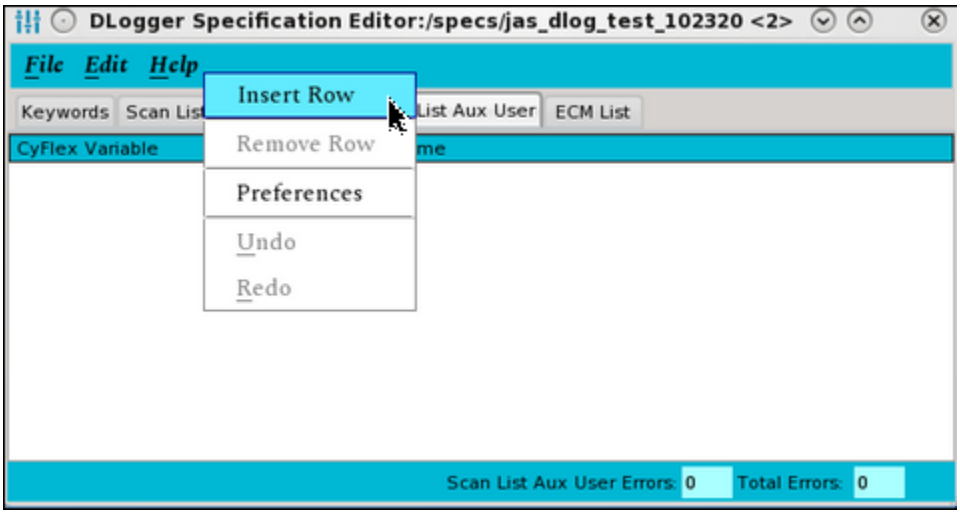
2.2.4 Scan List Aux User Tab

Select the **Scan List Aux User** tab to display **CyFlex Variable**s and associated **Alternate Name**s.

2.2.4.1 *Inserting Additional Scan List Aux User Rows*

Right-click a table row within the **Scan List Aux User** tab and select **Insert Row** from the pop-up menu as in *Figure 29*.

Figure 29: Insert Scan List Aux User Row

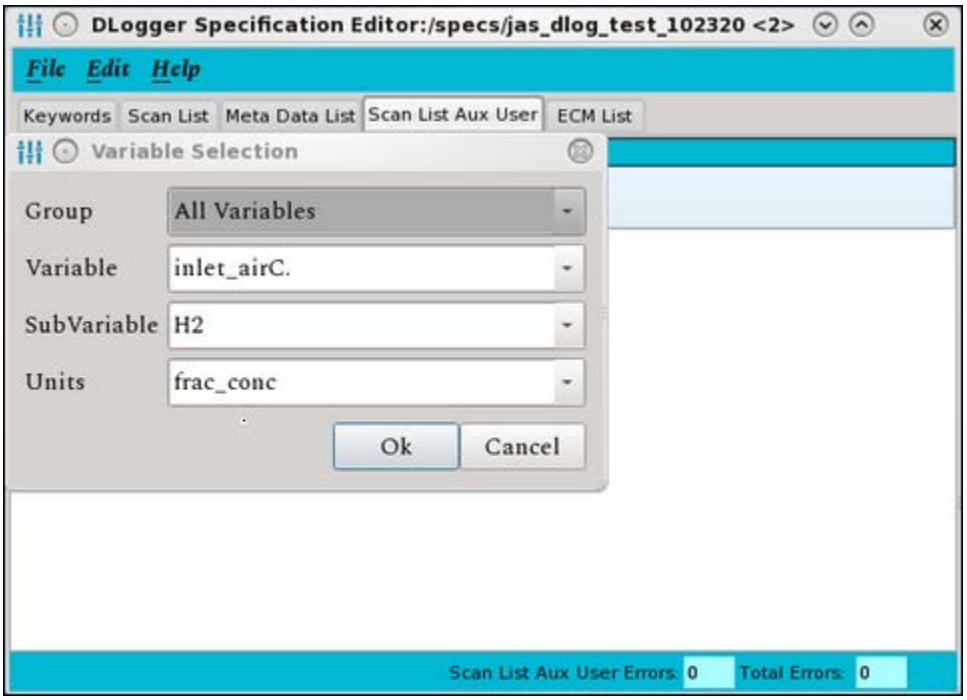


2.2.4.2 *Editing Scan List Aux User Rows*

Execute the following steps to edit data in a **Scan List Aux User** row:

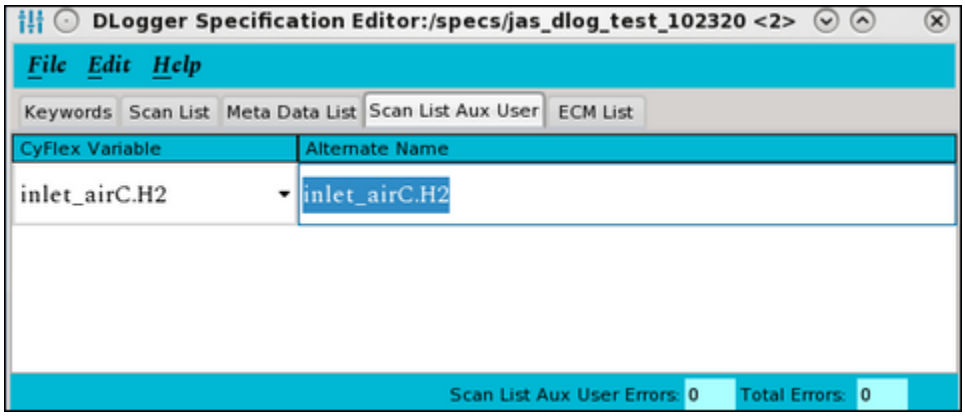
1. Double-click the **CyFlex Variable** column to display the **Variable Selection** dialog as in *Figure 30*.

Figure 30: Scan List Aux User Variable Selection



2. Select the type of variable to log from the **Group** drop-down and select the variable name to log from **Variable** drop-down.
3. Select **OK** to incorporate changes.
4. When selecting a variable name from the **Variable** Selection dialog, the **Alternate Name** box is automatically populated with the selected variable name. To edit the data within **Alternate Name** box, double click within the **Alternate Name** column and enter the correct information as in *Figure 31*.

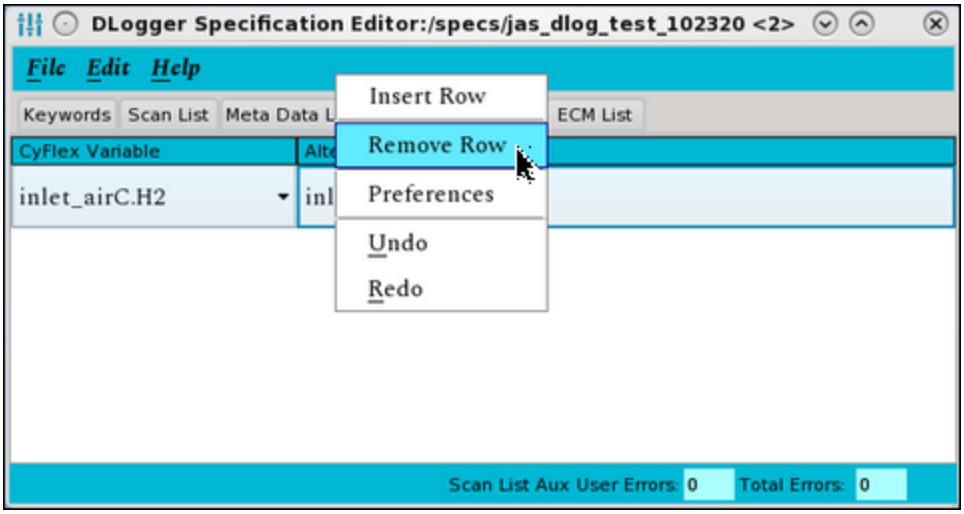
Figure 31: Edit Alternate Name



2.2.4.3 *Removing a Scan List Aux User Row*

Right-click a table row within the **Scan List Aux User** tab and select **Remove Row** from the pop-up menu as in *Figure 32*.

Figure 32: Remove Scan List Aux User Row



2.2.4.4 *Keywords that can be Added to the Spec File via the Scan List Aux User Tab*

Table 4 lists the keyword that can be added via the **Scan List Aux User** tab.

Table 4: Keywords that can be Added via the Scan List AUX User Tab

Generated Keyword In Spec File	Definition
`@SCAN_LIST_AUX_USER`	List of CyFlex variable values that are logged within the dlogger output file as the **Alternate Name** if one is specified. If the **Alternate Name** is not specified, the **CyFlex Variable** is referenced as the CyFlex Variable in the output file.

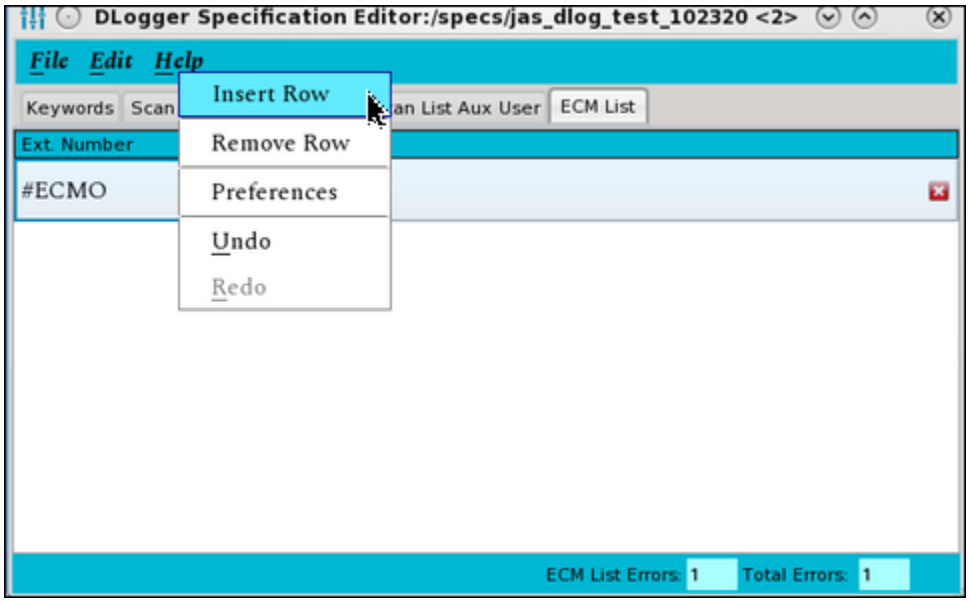
2.2.5 ECM List Tab

Select the **ECM List** tab to display **Ext. Number**s and associated **ASAM Specs Path**s.

2.2.5.1 *Inserting Additional ECM List Rows*

Right-click a table row within the **ECM List** tab and select **Insert Row** from the pop-up menu as in *Figure 33*.

Figure 33: Insert ECM List Row

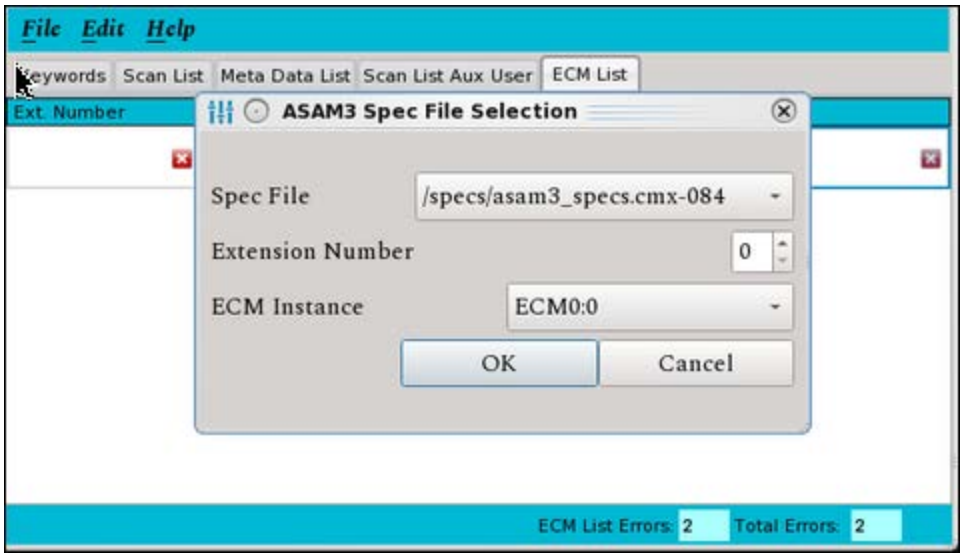


2.2.5.2 *Editing ECM List Rows*

Execute the following steps to edit data in an **ECM List** row:

1. Double-click the **ASAM Specs Path** column to display the **ASAM3 File Selection** dialog as in *Figure 34*.

Figure 34: ASAM3 File Selection

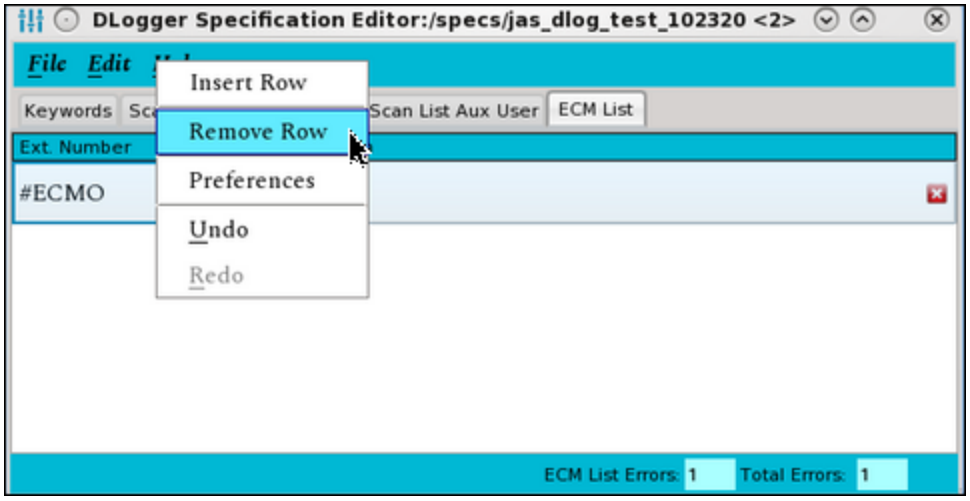


2. In the **ASAM3 File Selection** dialog, select the **Spec File** drop-down to select the location and filename for the ECM parameters to be logged.
3. Select the **ECM Instance** drop-down to select the ECM name and index number.
4. Select **OK** to incorporate changes.

2.2.5.3 *Removing an ECM List Row*

Right-click a table row within the **ECM List** tab and select **Remove Row** from the pop-up menu as in *Figure 35*.

Figure 35: Remove ECM List User Row



2.2.5.4 *Keywords that can be Added to the Spec File via the ECM List Tab*

Table 5 lists the keyword that can be added via the **Scan List Aux User** tab.

Table 5: Keywords that can be Added via the ECM List Tab

Generated Keyword In Spec File	Definition
`@ECM_LIST`	Location and filename of an ECM spec file where ECM variables names will be read to add to the list of variables that are logged within the dlogger output file

3 Using dlogger

3.1 Starting and Stopping dlogger

3.1.1 Starting dlogger

Start `dlogger` either from the command line or by using a script file. All arguments are optional. Refer to *Section 3.4 Command Options* on page 39 and to cyflex.com usage help for `dlogger`.

When started from the command line, dlogger reads a specification file. Refer to *Section 4 Specification Files* on page 40.

Example syntax:

```
$ dlogger dlogger_spec.315 &
```

If the file name is not included, the default file `/specs/dlogger_spec.`*nnn*) will be read where *.nnn* is the test cell name.

Another method for starting and controlling dlogger is through a test procedure which is a text file containing instructions for a test. The CyFlex program Test Manager (`gp_test`) reads the test procedure and directs the test accordingly. For more about test procedures and managing tests, refer to the *Test Manager User Guide*.

3.1.2 Stopping dlogger

Terminate `dlogger` program from the command line or by using an event called `release_event` in the specification file. Refer to *Section 4 Specification Files* on page 40.

A `release_event` signals the end of a sampling interval and terminates `dlogger` after it writes the data file.

To stop multiple instances of `dlogger` using the `release_event`, modify the respective specification files for each instance. If each instance is specified with a different `release_event`, the instances can be released separately. If all instances specify the same `release_event`, they are released at the same time. Refer to *Section 3.3 Multiple dlogger Instances* on page 39 for related information

If a `dlogger` task has no `stop_event` specified, the program closes the data file when the maximum number of scans is reached. This number is defined by the `@MAX_SCANS` keyword in the specification file. If the `@MAX_SCANS` keyword is not defined in the specification file, `dlogger`, once started, continues collecting data until a stop event is received.

3.2 Output Files

The `dlogger` output files reside in a directory on the test cell called `/data/dlog`. This is the default location.

1. While data is being collected, the file is written to a sub-directory called `/data/dlog/logging`.
2. When data collection is completed, the file is automatically moved to another sub-directory called /`data/dlog/ready`.

3. Next, the `tranMove` process transfers the file to a central node into the directory `/data/darts_dlogger/ready/$tc` where `$tc` is the test cell name.
4. An external data manager then moves the file to the server where the data is stored and available to the user for analysis using other tools that support the DARTS system.
5. When the process is complete, the system places an acknowledgement file and a copy of the analyzed data file in the test cell sub-directory called `/data/dlog/complete`.

The user can change the test cell directory where the output files are written. If in the `dlogger` specification file the user adds the `@OUTPUT_PATH` keyword followed by a new directory name, the output file will be written to that location. However, the specified directory must contain the same sub-directories for the output file that exist under the directory `/data/dlog/`. If those sub-directories are not present, an error message occurs and `dlogger` will not start.

Output files follow this naming convention:

`spec_filenameYYYYMMDDHHmmSSsss.dlog`

where:

`YYYYMMDDHHmmSSsss` is the Month/Day/Year/Hour/Minute/Second/Microsecond when the file was created.

3.2.1 Example Output File

```
$FormatRev
   dlogger_id
$FixedMetaData
   DESCRIPTION='dlogger description'
   TEST_ID=''
   TEST_TYPE=''
   MODE='TC103'
   GROUP=''
   PROGRAM='67781'
   SCAN_INTERVAL=0.020000000000000004163[sec]
   SYNC_EVENT='dlog_sync'
   REGISTERED_NAME='dlogger_spec.103'
   SPEC_FILENAME='/specs/dlogger_spec.103'
$MetaData
   AC_AIR_OT_P 9.458397 [in_hg]
   A/F 10.745051 [NONE]
   TEST_TM '3[hrs]'
   BLOW_BY 16.138042 [l/min]
   ECM_ACTIVE_FAULTS "789 222"
   ECM_ACTIVE_FAULTS 2 [none]
   USER30@11 "Disabled"
   H_PK_CYL_P@1 0.000000 [in_hg]
   CHP_O2_FLOW_DELAY -42.635000 [in_h2o]
$Data
time,AC_AIR_OT_P,BLOW_BY,A/F,H_PK_CYL_P@1,USER30@11,INT_MNF_P_VARIANCE,INT_MNF_P_VARIANCE,INT_MNF_P_VARIAN
CE,TEST_TM,ECM_ACTIVE_FAULTS

$Units
   date-time,in_hg,l/min,NONE,psi,none,in_h2o,in_h2o,in_h2o,MIN,none
$Values
"20160602 143543.107",9.45840,16.1380,10.7,22.5,"no",10.7457,10.7457,10.7457,240.000,"789 222"
"20160602 143543.114",9.45840,16.1380,10.7,22.5,"no",10.7457,10.7457,10.7457,240.000,"789 222"
"20160602 143543.133",9.83594,16.7393,11.1,23.7,"no",11.1041,11.1041,11.1041,240.000,"789 222"
"20160602 143543.153",10.20922,17.3131,11.4,24.8,"no",11.4450,11.4450,11.4450,240.000,"789 222"
```

```
"20160602 143543.173",10.58604,17.8504,11.8,25.9,"no",11.7631,11.7631,11.7631,240.000,"789 222"
"20160602 143543.193",10.95018,18.3428,12.1,26.8,"no",12.0534,12.0534,12.0534,240.000,"789 222"
"20160602 143543.213",11.29621,18.7825,12.3,27.7,"no",12.3113,12.3113,12.3113,240.000,"789 222"
"20160602 143543.233",11.62508,19.1625,12.5,28.4,"no",12.5328,12.5328,12.5328,240.000,"789 222"
"20160602 143543.254",11.93097,19.4769,12.7,29.0,"no",12.7143,12.7143,12.7143,240.000,"789 222"
"20160602 143543.273",12.20116,19.7206,12.9,29.5,"no",12.8531,12.8531,12.8531,240.000,"789 222"
"20160602 143543.293",12.44128,19.8899,12.9,29.8,"no",12.9468,12.9468,12.9468,240.000,"789 222"
"20160602 143543.313",12.63892,19.9821,13.0,30.0,"no",12.9941,12.9941,12.9941,240.000,"789 222"
"20160602 143543.333",12.79695,19.9958,13.0,30.0,"no",12.9941,12.9941,12.9941,240.000,"789 222"
"20160602 143543.353",12.91171,19.9306,12.9,29.8,"no",12.9469,12.9469,12.9469,240.000,"789 222"
"20160602 143543.373",12.97889,19.7877,12.9,29.5,"no",12.8533,12.8533,12.8533,240.000,"789 222"
"20160602 143543.393",12.99993,19.5693,12.7,29.0,"no",12.7146,12.7146,12.7146,240.000,"789 222"
"20160602 143543.413",12.97319,19.2788,12.5,28.4,"no",12.5332,12.5332,12.5332,240.000,"789 222"
"20160602 143543.433",12.89960,18.9208,12.3,27.7,"no",12.3118,12.3118,12.3118,240.000,"789 222"
"20160602 143543.453",12.78157,18.5010,12.1,26.8,"no",12.0539,12.0539,12.0539,240.000,"789 222"
"20160602 143543.473",12.61879,18.0260,11.8,25.9,"no",11.7636,11.7636,11.7636,240.000,"789 222"
"20160602 143543.493",12.41266,17.5033,11.4,24.8,"no",11.4456,11.4456,11.4456,240.000,"789 222"
"20160602 143543.513",12.17255,16.9411,11.1,23.7,"no",11.1047,11.1047,11.1047,240.000,"789 222"
"20160602 143543.533",11.89613,16.3483,10.7,22.5,"no",10.7464,10.7464,10.7464,240.000,"789 222"
"20160602 143543.553",11.58980,15.7342,10.4,21.3,"no",10.3763,10.3763,10.3763,240.000,"789 222"
"20160602 143543.573",11.25527,15.1085,10.0,20.0,"no",10.0003,10.0003,10.0003,240.000,"789 222"
"20160602 143543.593",10.90387,14.4812,9.6,18.7,"no",9.6243,9.6243,9.6243,240.000,"789 222"
"20160602 143543.613",10.54161,13.8620,9.3,17.5,"no",9.2543,9.2543,9.2543,240.000,"789 222"
"20160602 143543.633",10.16752,13.2607,8.9,16.3,"no",8.8959,8.8959,8.8959,240.000,"789 222"
"20160602 143543.653",9.78734,12.6869,8.6,15.2,"no",8.5550,8.5550,8.5550,240.000,"789 222"
"20160602 143543.673",9.41735,12.1496,8.2,14.1,"no",8.2369,8.2369,8.2369,240.000,"789 222"
```

3.3 Multiple dlogger Instances

More than one instance of `dlogger` may run simultaneously with each copy performing different functions based on its specification file. Each copy of the program must have a unique name so that:

- The program instance is registered with the Operating System (OS) and can initialize correctly.
- Other CyFlex programs (such as Test Manager) can communicate with a particular instance if needed.

In the `dlogger` specification file, the `@REG_NAME` keyword identifies the program instance.

3.4 Command Options

An option entered at the command line overrides the same option in the specification file.

Syntax:

```
dlogger [dlogger_spec_file] [switch] [options] &
```

where:

`dlogger_spec_file` = the name of the DARTS logger specification file

This argument is not required.

The default is: `/specs/dlogger_spec.`*nnn*

where:

nnn = the test cell name

Switch:

`+H` Writes a new file when `enable_variable` becomes `TRUE`

Example:

```
dlogger /specs/dlog1 n=1000 interval=1[sec] &
```

The preceding command starts processing of the file `/specs/dlog1`. The first option sets the number of samples to 1,000. The second option instructs that samples be taken every second.

Refer to full cyflex.com usage help for `dlogger`.

4 Specification Files

4.1 Specification File Format

The dlogger specification file is made up of “keywords.” Each keyword begins with the @ symbol which identifies to `dlogger` that the line is a keyword. The text following the symbol on the next line describes an action or process for the program to perform.

Example:

```
@START_EVENT
```

The keyword is uppercase text without any blank spaces. The next line(s), called the *`keyword_value`*, specifies the action or process.

Example:

```
@START_EVENT
start_logging
```

Certain keywords must be specified in the specification file before `dlogger` can run. Those keywords are identified in *Table 6* as “Required.” However, if a function will not be used, the corresponding keyword does not have to be included in the file.

4.2 Specification File Keywords

Table 6 lists all keywords available for dlogger specification files.

Several keywords are required in the specification file as indicated in the table below.

Certain keywords support computed expressions. Those keywords and more about computed expressions are described in *Section 4.3 Computed Expressions* on page 50.

Table 6: Specification File Keywords

Keyword	Required	Description

`@CLEAR_STATISTICS_EVENT`	No	This event causes the statistical buffers within *dlogger* to be reset to 0.
		Example:
		`@CLEAR_STATISTICS_EVENT`
		`clear_stats`
		This might be used at the start of a test mode so that statistics only apply to data taken in that mode.

`@DARTS_STEADY_STATE`	No	The presence of this keyword will cause two things to happen. First, the scan list (described below) will automatically include all entries in the default PAM specs file. Second, the output format will be changed to comply with the input requirements of the RAPID data analysis process. This keyword does not require a follow-on entry.

Keyword	Required	Description
@DESCRIPTION	Yes	The user configurable description appears at the top of the output file. Example of a simple string description is enclosed in single quotes: @DESCRIPTION 'This is a description of my test.' A more complex description can be constructed using a computed expression. @DESCRIPTION 'Torque sweep, model=' + model + ', S/N= ' + serial If the CyFlex variables 'model' and 'serial' had values of 15LTA and 14026490 respectively, the following would be written to the output file: Torque sweep, model=15LTA, S/N= 1402690 An error occurs and the program exits if the keyword is not in the spec file.
@DLOGGER	Yes	This is to clarify that the spec file is a *dlogger* spec file. It presently has no need for associated data. Example: @DLOGGER dlogger_id
@DONE_EVENT	No	The name of the event that is set when the data collection is complete. This is an output event and can be used to inform another process that the output file is now available. Example: @DONE_EVENT logging_done When the event is received, the data file is moved to the /ready directory.
@ECM_LIST	No	This keyword is followed by a list of asam3 specification file names, ECM names and channel lists that are to be logged. All files, ECM names, and channel lists must exist prior to running dlogger. Example: @ECM_LIST 0 /specs/asam3_specs.123:ECM0:0 1 /specs/asam3_specs.123:ECM1:2

Keyword	Required	Description
@ENABLE	No	The enable variable is a logical variable that must be set to TRUE before logging can take place. Typically, this variable is set in a test procedure or manually by the user. It may be used to turn logging on and off at different modes of a test. Example: @ENABLE logging_ok
@FIFO_LOG_BUFFER	No	Activates First-in First-out (FIFO) logging. This keyword allows dlogger to collect data in a “circular” buffer. The buffer fills with data until the maximum number of scans defined by @MAX_SCANS have been captured and then repeatedly refreshes according to the FIFO technique. The buffer is written to the output file when the “trigger” event is received. The trigger event is either the @RELEASE_EVENT or the @STOP_EVENT. The buffer begins filling when the @START_EVENT is received or starts immediately if no @START_EVENT is specified. If the @MAX_SCANS and the trigger event are not specified, FIFO logging is not made active. Example: @FIFO_LOG_BUFFER Specifying the keyword alone within the spec file enables the FIFO logging feature.
@FIFO_POST_ TRIGGER_INTERVAL	No	Specify the length of time to obtain scans after the FIFO trigger event (stop or release event) has been received. The INTERVAL keyword has precedence over the FIFO_POST_TRIGGER_SCANS keyword. If both are specified, the FIFO_POST_TRIGGER_INTERVAL value will be used. Example: @FIFO_POST_TRIGGER_INTERVAL 5[sec]
@FIFO_POST_ TRIGGER_SCANS	No	Specify the number of scans to obtain after the FIFO trigger event (stop or release event) has been received. Example: @FIFO_POST_TRIGGER_SCANS 39

Keyword	Required	Description
@FORCE_DIRECT_FILE_WRITE	No	Indicates that data should be written directly to the output file when high data rates are used. Care should be exercised when using this command with very high data rates so that excessive CPU time is not used by the dlogger program. Example: @FORCE_DIRECT_FILE_WRITE YES Note: if no value follows the keyword, then a value of YES is assumed.
@FTP_EVENT	No	Specifies the CyFlex event that will be set to cause the output file to be finalized. This initiates the transfer of the dlogger output data file to the /ready directory. The default is FTP_write. Example: @FTP_EVENT ftp_log_data
@GET_NEW_SCAN_INTERVAL	No	The name of an event that can be used to trigger a re-evaluation of the SCAN_INTERVAL computed expression. Example: @GET_NEW_SCAN_INTERVAL New_dlog_intvl If the event does not exist when the *dlogger* task is started, dlogger can create the event.
@GROUP	Yes	Label of CyFlex String variable that contains the measurement name to include in the output file meta-data GROUP='<*value*>'
@LOG_DIGITAL_DESCP	No	This keyword causes the LOGICAL_VARIABLE descriptions to be logged for all LOGICAL_VARIABLEs in place of the values 0 or 1. The keyword can have an entry following it of either yes or no. If no entry follows this keyword the value of yes is assumed. Example: @LOG_DIGITAL_DESCP Yes
@LOG_STATISTICS	No	Specifies that statistics should be computed for the variables specified via the @SCAN_LIST keyword. The statistical variables are created locally and are not available to any other process. Data collection begins with the start event (@START_EVENT keyword) and is collected at the rate specified by the @SCAN_INTERVAL keyword. The statistical values are logged to the output file when a

Keyword	Required	Description

		stop event (`@STOP_EVENT`) is received or the maximum number of samples (`@MAX_STATISTICAL_SCANS` keyword) have been collected. This also stops the data collection.
		By default, the average value, `AV` member of the statistical variable, is logged; however, additional members may be logged. See `@SCAN_LIST` keyword on page 47.
		When another start event is received, all statistical buffers are cleared and the data collection process begins again.
		NOTE: If another start event is received before a stop event is, or the maximum number of scans is reached, there is no output. The variables are cleared and the data collection begins again.
		Example:
		`@LOG_STATISTICS`
		`YES`

`@LOGGING_ACTIVE LABEL`	No	The name of a CyFlex Logical variable that indicates `dlogger` is actively collecting data.
		Example:
		`@LOGGING_ACTIVE_LABEL`
		`Loger_collecting`

`@MAX_SCANS`	No	The maximum number of samples in a sampling session. A zero value or the keyword being absent indicates a sampling session will continue until a `stop_event` is received.
		Example:
		`@MAX_SCANS`
		`1000`
		When this scan count is reached, data is moved from the output directory (`/logging`) to a sibling directory (`/ready`).

`@MAX_STATISTICAL_ SCANS`	No	Specifies the maximum number of scans when the `@LOG_STATISTICS` keyword is specified.
		Example:
		`@MAX_STATISTICAL_SCANS`
		`1000`
		This causes statistics to be calculated when 1000 scans have completed.

Keyword	Required	Description

@META_DATA	Yes	This is a list of PAM/DARTS/Mach parameter keywords, each with a corresponding CyFlex variable or literal string. These are written to the `$MetaData` header section of the `dlogger` output file.
		Specification file format:
		<keyword1> <*CyFlexVarLabel1*>
		<keyword2> <*CyFlexVarLabel2*>
		<keyword3> 'literal value3 [<units3>]'
		Example:
		`@META_DATA`
		`#real`
		`AC_AIR_OT_P  ac_air_ot_p`
		`#real`
		`A/F  FR_AF_ratio`
		`#literal string`
		`TEST_TM  '3[hrs]'`
		`#real`
		`BLOW_BY  blow_by_vf`
		`#string`
		`ECM_ACTIVE_FAULTS  active_faults`
		`#integer`
		`ECM_ACTIVE_FAULTS  fault_value`
		`#logical`
		`USER30@11  fish_hook`
		`#real – computed variable`
		`H_PK_CYL_P@1  ac_air_in_p`
		`#real - ai_input`
		`CHP_O2_FLOW_DELAY  air_mtr0_p`
		The parameter keywords are derived from the `/cyflex/parameter.dat` file content.

@MODE	Yes	Label of CyFlex String variable that contains the test mode to include in the output file meta-data `MODE='<value>'`
		Example:
		`@MODE`
		`test_mode`

@OUTPUT_PATH	No	The directory path that specifies where the output file should be written. If this keyword is absent, then the default path is: `/data/dlog/logging`.
		Example:
		`@OUTPUT_PATH`
		`/data/my_data/`
		This can be a string variable that contains the directory path for the output file. The directories, `/logging/ready` and `/logging/complete` must exist in the specified directory or the `dlogger` task will not start properly.

Keyword	Required	Description
@PROGRAM	Yes	Label of CyFlex string variable that contains the program or project name to include in the output file meta-data PROGRAM='<*value*>' Example: @PROGRAM *prog_proj*
@READ_SPEC_FILE_EVENT	No	The name of an event that, when received by dlogger causes *dlogger* to re-read the spec file. @READ_SPEC_FILE_EVENT read_it
@REG_NAME	No	The name that registers the instance of dlogger with the OS. The name must be unique throughout out the system or the task will fail to initialize correctly. Example: @REG_NAME CVS_FTP75
@RELEASE_EVENT	No	The name of the event that signals the end of a sampling interval and terminates the dlogger task after the data file was written. Example: @RELEASE_EVENT release_dlog When received, data is moved from the output location to a sibling directory, /ready.
@RUNNING_AVERAGE	No	Specify the window width of a running average and the event that causes the data to be logged. This keyword causes statistics to be computed for the variables specified via the @SCAN_LIST keyword. The variables are created locally and are not available to any other process. The statistics are computed for the specified window width and continue to be computed as long as data collection is active. The total number of data points making up the running average is a function of the window width and the scan interval as specified by the @SCAN_INTERVAL keyword. Computed expressions are allowed for the window width specification. The values are logged to the output file whenever the specified 'log data event' is received. By default, the average (AV) member is logged; however, additional members may be logged. See @SCAN_LIST keyword on page 47. Example:

Keyword	Required	Description
		@RUNNING_AVERAGE # window width log data event 30[sec] log_average_data
@SCAN_INTERVAL	Yes	The time between lines of data in the output file. Example: @SCAN_INTERVAL 0.20[sec] OR @SCAN_INTERVAL Variable_name Note: Scan intervals that are less than one second cause data to be saved in memory and written to the output file until a stop_event is received or the @MAX_SCANS value is reached. This feature can be overridden with the @FORCE_DIRECT_FILE_WRITE keyword, refer to page 43. If the SCAN_INTERVAL is entered as a computed expression, the expression is evaluated each time that the START_EVENT is set.
@SCAN_LIST	Yes	This is the list of variables that are to be sampled. Each variable specified may have units specified and/or a statistical member, and/or a LOG_DIGITAL_DESCRIPTION keyword for logical variables. Each variable must include a DARTS (PAM) keyword. Statistical members are valid only if the @LOG_STATISTICS or @RUNNING_AVERAGE keyword was specified before the SCAN_LIST keyword is processed. The statistics are computed internally and are not the values of any CyFlex statistical variable. Variables to be sampled are listed in this format: @ SCAN_LIST <variable> <DARTS KEYWORD> Variables (labels) and their corresponding DARTS keywords are defined as follows: label_1 DARTS_KEYWORD1 This is the CyFlex variable to be logged and the corresponding DARTS keyword written to the data header of the output file. label_2[units] DARTS_KEYWORD2 Optionally, output units may be specified for each label. label_2 .MX DARTS_KEYWORD3 A statistical member may be specified when either @LOG_STATISTICS or the @RUNNING_AVERAGE keyword is specified

Keyword	Required	Description

		`label_3  LOG_DIGITAL_DESCP  DARTS_KEYWORD4`
		`Label_3` is a logical variable and the description will be logged instead of a `0` or `1`.
		Note: The optional units, statistical member and `LOG_DIGITAL_DESCP` shown above may be specified in any order. If units are specified, they must immediately follow the label name and be enclosed in [brackets]
		Example 1:
		`@SCAN_LIST`
		`int_mnf_T  DARTS_KEYWORD1`
		`int_mnf_p[in_hg]  DARTS_ KEYWORD2`
		The above will log `int_mnf_p` in units of 'inches of mercury'.
		Note: When specifying the units, there should not be a space between the variable name and the specified units of `[in_hg]`.
		Example 2:
		`@SCAN_LIST`
		`int_mnf_t  DARTS_KEYWORD1`
		`int_mnf_t .MX  DARTS_KEYWORD2`
		`int.mnf_t .SD  DARTS_KEYWORD3`
		If the `@LOG_STATISTICS` or `@RUNNING_AVERAGE` keyword was specified, then the values logged are different than described above. If the `@LOG_STATISTICS` keyword was specified and the scan list is that shown in example 2, the log file will contain the average value of the parameter `int_mnf_t` as the first value and will have the maximum value of the parameter `int_mnf_t` as the second value. The maximum value member was specified by entering the standard two-character CyFlex statistical variable member preceded by a period, and also needs to be separated from the root variable label by a space. Refer to keywords `@LOG_STATISTICS` on page 43 and `@RUNNING_AVERAGE` on page 46 for more information.
		Example 3:
		If the specified variable that is being logged is a `LOGICAL_VARIABLE`, the logical description may be logged in place of the values `0` or `1`.
		For example, if the following channel specification is an entry under the `@SCAN_LIST` keyword,
		`enab_lwr_lmtLOG_DIGITAL_DESCP DARTS_KEYWORD5`
		the `LOGICAL_VARIABLE` description of `enab_lwr_lmt` will be logged in place of the values `0` or `1`. The entry must be exactly `LOG_DIGITAL_DESCP` or the logical description will not be logged.

Keyword	Required	Description

`@SCAN_LIST_AUX_USER`	No	Use this keyword to create a list of variables that are to be logged and stored by another name in DARTS.
		Example:
		`@SCAN_LIST_AUX_USER`
		`# Variable Name  Store in DARTS as`
		`fuel_heater_ot_t  my_variable@1`

`@START_EVENT`	No	The name of an event that signals the start of a sampling interval.
		Example:
		`@START_EVENT`
		`start_logging`

`@STOP_EVENT`	No	The name of an event that signals the end of a sampling interval.
		Example:
		`@STOP_EVENT`
		`stop_logging`
		When this is received, data is moved from the output location to a sibling directory `/ready`.

`@SYNC_EVENT`	No	The name of an event that can be used to trigger a scan of all channels, usually as an alternative to sampling at a periodic interval. If both `@SCAN_INTERVAL` and `@SYNC_EVENT` are specified, the sync scans and interval scans are interlaced.
		Example:
		`@SYNC_EVENT`
		`log_now`

`@TEST_ID`	Yes	Label of a CyFlex string variable that contains the test ID to include in the output file meta-data `TEST_ID='<value>'`
		Example:
		`@TEST_ID`
		`test_id`

`@TEST_TYPE`	Yes	Label of CyFlex string variable that contains the test name to include in the output file meta-data `TEST_TYPE='<value>'`
		Example:
		`@TEST_TYPE`
		`test_type`

4.3 Computed Expressions

Computed expressions may be used in `dlogger` specification files for various keywords.

A computed expression allows the user to specify the value of a variable as a function of other variables in the system. The user may create a variable and assign it a computed expression. The variable value is then computed by CyFlex based on the expression that the user supplies, which arithmetically combines other variable values.

The following specification file keywords support computed expressions:

- `@SCAN_INTERVAL`
- `@DESCRIPTION`
- `@MAX_SCANS`
- `@ENABLE`
- `@RUNNING_AVERAGE`
- `@FIFO_POST_TRIGGER_INTERVAL`
- `@MAX_STATISTICAL_SCANS`
- `@OUTPUT_PATH`

Guidelines for using computed expressions and strings in keywords:

- Computed expressions must be enclosed in double quotes (`"…"`)
- A literal string must be enclosed in single quotes (`'…'`)
- Strings may be joined using the plus (+) sign

Example:

@DESCRIPTION

```
'Engine model = ' + model + ' S/N = ' + serial
```

If the CyFlex string variable <*model*> had a value of <`13LTA`>, and the string variable <*serial*> had a value of <`14014957`>, the test description (`@DESCRIPTION` keyword) for the dlogger file equals:

```
Engine model = 13LTA S/N = 14014957
```

Computed expressions, strings and the variable types that may be assigned a computed expression are described in *Creating User Computations and User Variables*.