Mission Name

core Flight System (cFS)

Limit Checker (LC)

Version 2.1.0.0

APPLICATION USER'S GUIDE

Flight Software Systems Branch – Code 582

Release Version 1.1 – 8/29/17

582-2012-004

National Aeronautics and Space Administration

FORWORD

This Core Flight System (CFS) Limit Checker (LC) Application User's Guide provides guidance for the Flight Operations Team (FOT) for the CFS LC Application.

This is one of a set of enhanced User's Guides planned for the CFS Product Documentation Suite. While the main audience is the FOT, the Guides also help serve the needs of flight software developers; Flight Software Sustaining Engineering (FSSE), Integration and Test (I&T), and others who support missions which use CFS.

This generic CFS LC Guide is set up so that missions can convert it into a mission-specific guide.

SIGNATURES

Approved by:

X SUSANNE STREGE

Digitally signed by SUSANNE STREGE
Date: 2017.08.29 15:55:26 -04'00'

Susanne Strege/582
cFS Flight Software Product Development Lead

AUTHOR

[illegible]

Gary M Smith

Technical Writer

9/19/2012

Date

APPROVALS

Susanne Strege

Susanne Strege / 582

Core Flight Executive (cFE) Core Flight System (CFS) / PDL

9/26/2012

Date

Charles Wildermann

Charles Wildermann / 582

Flight Software Systems Branch / Head

9-28-2012

Date

UPDATE HISTORY

Version	Date	Description	Affected Pages
Draft 0.1	07/27/12	1st Draft – Partial	All
Draft 0.2	09/04/12	2nd Draft	All
1.0	09/19/12	Release Version	All
1.1	01/19/17	Combines the LCX supplement information into a single LC users guide. Added new signature page. Keeping old signature page for historical purposes.	All

CONTENTS

TABLE OF FIGURES

TABLES

Chapter 1. **Introduction to the CFS LC User's Guide**

1.1 ***Purpose and Scope of this Guide***

The primary purpose of this Application User's Guide is to help the Flight Operations Team (FOT) understand the Limit Checker (LC) application.

Many other purposes will be found for this Guide, including helping mission flight software personnel populate the ground system Record Definition Language (RDL) files in the ground system used later by the FOT, e.g., Advanced Spacecraft Integration & System Test software (ASIST).

Further purposes of this Guide are to help mission developers, system integration and test team members, and Flight Software Sustaining Engineering (FSSE) to understand the LC application for their own specific needs, such as using the software to perform certain hardware tests.

As delivered, this is a generic document ready to insert mission defined values to serve the needs of specific missions.

1.2 ***Acknowledgements***

This Application User's Guide relies heavily on the content of earlier heritage LC publications, presentations, and interviews with flight software engineers. Appendix A is based on information from CFS LC source code as processed by Doxygen in Rich Text Format (RTF) output, and reformatted for this linear publication.

1.3 ***Conventions***

- In this document, flight controller, ground controller, and the Flight Operations Team (FOT) are used interchangeably.
- In this document, the percent sign (%), when followed by a string, may indicate variable text. See Appendix A for references to the text that may be substituted in each case.
- Core Flight Executive is abbreviated cFE (lower case "c" with uppercase "FE").
- *See Appendix C for more.*

1.4 ***Related Documents***

Documents used in the preparation of this Guide are listed in the table below.

Table 1 Related Documents

Item No.	Document ID	Document Source
1	N/A	Hardison, David. *Core Flight System Limit Checker (LC) Application [Design Presentation].* Greenbelt, MD: Goddard Space Flight Center, Code 582 (Flight Software Branch), 25 Jun 2008. PPT.
2	N/A	Hardison, David. *CFS_LC_Heritage_Analysis*. Greenbelt, MD: Goddard Space Flight Center, Code 582 (Flight Software Branch), 5 Jun 2008. PPT.
3	582-2008-007	Flight Software Branch. *Core Flight System (CFS) Limit Checker Application Heritage Analysis, Version 1.1.* Greenbelt, MD: Goddard Space Flight Center, Code 582 (Flight Software Branch), 10 Jun 2008. DOC.
4	N/A	*CFS LC Requirements Document V1_2 021710.* Greenbelt: NASA Goddard Space Flight Center, Code 582, Flight Software Systems Branch, 17 Feb. 2010. PDF.
5	N/A	Strege, Susie. *CFS_LC-UsersGuide-6-7-12.* Greenbelt: NASA Goddard Space Flight Center, Code 582, CFS Product Development Team, 7 June 2012. Doxygen Output via RTF.

1.5 *Assumptions*

1.5.1 Personnel

This Application User's Guide assumes the primary reader is either a new or long term member of the FOT at NASA or the equivalent elsewhere.

This Guide also assumes that many other audiences may find it useful, as described in *Purpose and Scope of this Guide* above.

1.5.2 CFS LC Application

The following list summarizes the assumptions made about the CFS Limit Checker Application as documented in this Guide:

- The CFS LC code has not been modified.
- The CFS LC Application has been configured using the standard CFS LC configuration parameters.
- Core Flight Executive, (cFE) Application Programming Interface (API), and Operating System Abstraction Layer (OSAL) are being used.
- Consultative Committee for Space Data Systems (CCSDS) File Delivery Protocol (CFDP) is being used.
- Commands are sent to LC by the CFS Scheduler (SCH) application to sample actionpoints at a periodic user defined rate.

1.6 *How to Use this Document*

This Application User's Guide is set up as a learning tool before launch, with an Appendix as a reference tool after launch. It can be read offline as a paper printout, or the electronic file can be searched. It may be used along with Doxygen generated .html files for more complete understanding.

Read the sections from the beginning up to just before the Appendix as needed to understand how LC works. Refer to the Appendix for specifics when setting up operational scenarios.

Experienced flight controllers may only need to browse the Guide, and use the Appendix as needed. New flight controllers may wish to get more familiar with the entire Guide well before needing to rehearse operational scenarios or set up flight software procedures ("procs").

After launch, refer to the Appendix as needed.

1.7 *Acronyms and Abbreviations*

Acronyms and abbreviations in this publication are shown in Table 2 below. Telemetry and command mnemonics and similar terms, are not shown here.

Table 2 Acronyms and Abbreviations

Acronym/Abbreviation	Description
AD	Actionpoint Definition
ADT	Actionpoint Definition Table
AP	Actionpoint
API	Application Programming Interface
ART	Actionpoint Results Table
ASIST	Advanced Spacecraft Integration & System Test software
CC	Command Code
CCSDS	Consultative Committee for Space Data Systems
CDS	Critical Data Store
cFE	Core Flight Executive
CFS	Core Flight System
CI	Command Ingest Application
CM	Configuration Management
CMD	Command
Dis	Disable
DType	Data Type
DWord	32 bit signed double word
Ena	Enable

Acronym/Abbreviation	Description
Err	Error
Evt	Event
FOT	Flight Operations Team
FSSE	Flight Software Sustaining Engineering
FSW	Flight Software
HK	Housekeeping Application
Hz	Hertz, Cycles per Second
I&T	Integration and Test
LC	Limit Checker Application
Len	Length
LRO	Lunar Reconnaissance Orbiter
MID	Message ID
MMS	Magnetospheric Multiscale Mission
MOT	Mission Operations Team
ms	Milliseconds
MS	Microsoft
MsgLen	Message Length
NAN	Not a Number
No-op	No operation
Oper	Operator
OS	Operating System
OSAL	Operating System Abstraction Layer
PDF	Portable Document Format
PDL	Project Development Lead
Perm	Permanently
PKT	Packet
Proc	Process
RC	Return Code
RCVD	Received
RDL	Record Definition Language
RPN	Reverse Polish Notation; also known as Postfix notation
RTF	Rich Text Format
RTS	Relative Time tagged command Sequence

Acronym/Abbreviation	Description
SB	Software Bus Service application
SC	Stored Command Application
SCH	Scheduler Application
SDO	Solar Dynamics Observatory
StackPtr	Stack Pointer
TBL	Table Services application
TLM	Telemetry
TO	Telemetry Output Application
TRACE/WIRE	Transition Region and Coronal Explorer / Wide-field Infrared Explorer
TS	Telemetry and Statistics Monitor
UByte	8 bit unsigned byte
UDWord	32 bit unsigned double word
uint16	Unsigned 16-bit integer
uint32	Unsigned 32-bit integer
uint8	Unsigned 8-bit integer
UWord	16 bit signed word
WDT	Watchpoint Definition Table
WP	Watchpoint
WRT	Watchpoint Results Table

This page intentionally blank

Chapter 2. **Introduction to the CFS LC Application**

2.1 ***Heritage***

In June of 2008, a Heritage Analysis was performed on Lunar Reconnaissance Orbiter (LRO) Limit Checker (LC) and Solar Dynamics Observatory (SDO) Telemetry and Statistics Monitor (TS). Each of these tasks had heritage from earlier missions. While both the SDO TS and LRO LC applications were developed to solve a similar problem (how to monitor and respond to spacecraft anomalies), they each took very different approaches.

Based upon the heritage analysis it was recommended that the LRO LC design with Transition Region and Coronal Explorer (TRACE) / Wide-field Infrared Explorer (WIRE) heritage be used as the basis for the CFS Limit Checker. This was preferred by Flight Software Sustaining Engineering (FSSE) personnel over the SDO TS approach.

Further, during the design of the custom function for LC, it was decided to keep the heritage design using a function stub, instead of using the OSAL module loader to allow dynamic updating during flight. It was decided that in-flight updating of custom functions is rare and not worth the added complexity of a dynamic loading approach. It was decided that the normal methods of patching or rebuilding the LC application would be sufficient.

In the spring and summer of 2012, the Magnetospheric Multiscale Mission (MMS) requested additional functionality to LC to add support for Stale Watchpoint/Actionpoint Results. This support was originally added to a new implementation of the LC application called Limit Checker eXtended (LCX). LCX has now replaced the original LC application and is simply LC.

2.2 ***CFS Limit Checker (LC) Overview***

The LC application monitors telemetry data points in the flight system and compares the values against predefined or computed threshold limits. When a threshold condition is encountered, an event message is issued and a Relative Time Sequence command script may be initiated to respond to the threshold violation. ***See also:*** Core Flight System (CFS) Stored Command (SC) Application User's Guide.

LC is responsible for monitoring watchpoints and evaluating actionpoints. LC is table driven and also command driven.

Each **watchpoint** compares a telemetry data value with a predefined constant threshold limit in the normal case. By using a custom function the threshold can be a computed value.

The comparison result may be:

- True
- False
- Error, or

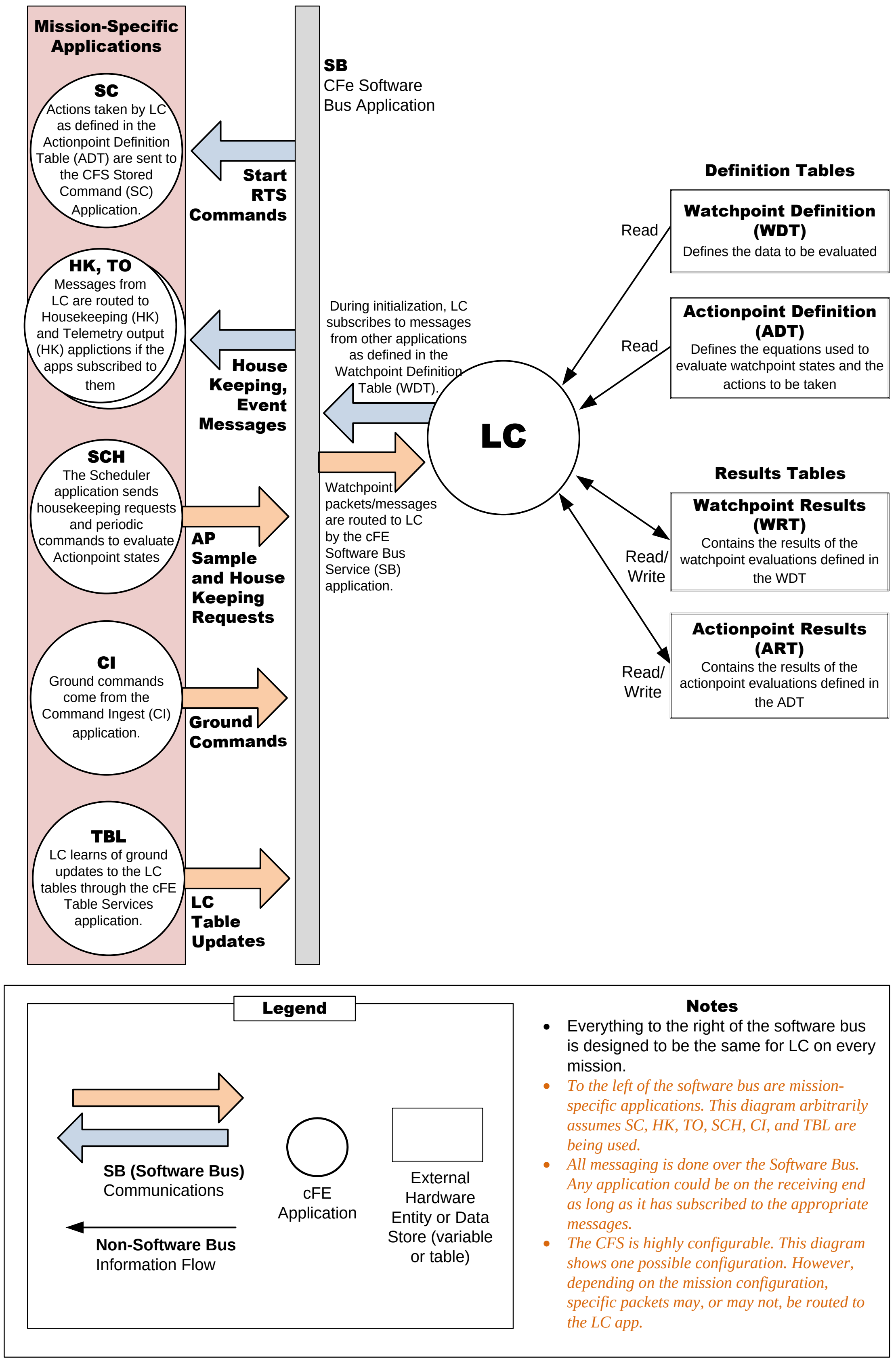
- Stale

Each **actionpoint** analyzes the results of one (or more) watchpoints. Analysis result may be:

- Pass
- Fail
- Error, or
- Stale.

If the number of consecutive fails exceeds a preset limit, then LC sends an event and optionally invokes an RTS. No RTS or event is issued until the Actionpoint (AP) expression is evaluated based on the current watchpoint states.

Figure 1 below shows the software context for the CFS Limit Checker (LC) Application.


Mission-Specific Applications
SC
Actions taken by LC as defined in the Actionpoint Definition Table (ADT) are sent to the CFS Stored Command (SC) Application.
HK, TO
Messages from LC are routed to Housekeeping (HK) and Telemetry output (HK) applictions if the apps subscribed to them
SCH
The Scheduler application sends housekeeping requests and periodic commands to evaluate Actionpoint states
CI
Ground commands come from the Command Ingest (CI) application.
TBL
LC learns of ground updates to the LC tables through the cFE Table Services application.
Start RTS Commands
House Keeping, Event Messages
AP Sample and House Keeping Requests
Ground Commands
LC Table Updates
SB
CFe Software Bus Application
During initialization, LC subscribes to messages from other applications as defined in the Watchpoint Definition Table (WDT).
Watchpoint packets/messages are routed to LC by the cFE Software Bus Service (SB) application.
LC
Definition Tables
Read
Watchpoint Definition (WDT)
Defines the data to be evaluated
Read
Actionpoint Definition (ADT)
Defines the equations used to evaluate watchpoint states and the actions to be taken
Results Tables
Read/ Write
Watchpoint Results (WRT)
Contains the results of the watchpoint evaluations defined in the WDT
Read/ Write
Actionpoint Results (ART)
Contains the results of the actionpoint evaluations defined in the ADT
Legend
SB (Software Bus) Communications
Non-Software Bus Information Flow
cFE Application
External Hardware Entity or Data Store (variable or table)

Notes

- Everything to the right of the software bus is designed to be the same for LC on every mission.
- *To the left of the software bus are mission-specific applications. This diagram arbitrarily assumes SC, HK, TO, SCH, CI, and TBL are being used.*
- *All messaging is done over the Software Bus. Any application could be on the receiving end as long as it has subscribed to the appropriate messages.*
- *The CFS is highly configurable. This diagram shows one possible configuration. However, depending on the mission configuration, specific packets may, or may not, be routed to the LC app.*

Figure 1 LC Application Software Context

2.2.1 Tables in Support of Watchpoints and Actionpoints

LC is a table driven application but can also accept commands. LC uses Watchpoint and Actionpoint Definition Tables for characterizing mission specific telemetry limits and actions, and Watchpoint and Actionpoint result tables for statistics.

LC contains these tables in support of Watchpoints and Actionpoints:

Definition Tables

1. Watchpoint Definition Table (WDT) – defines the data to be evaluated, such as, for example, message ID, offset, comparison value, and age when watchpoint comparison result becomes stale. The WDT specifies a Stale Limit for each watchpoint. The Stale Limit specifies how long after evaluation before the watchpoint result becomes stale.
2. Actionpoint Definition Table (ADT) – defines the equations used to evaluate watchpoint states and the actions to be taken

Results Tables

1. Watchpoint Results Tables (WRTs) – contain the results of the watchpoint evaluations defined in the WDT. The WRT specifies a Stale Counter for each watchpoint. The Stale Counter specifies how much longer before the watchpoint result becomes stale.
2. Actionpoint Results Tables (ARTs) – contain the results of the actionpoint evaluations defined in the ADT.

2.2.2 Scheduler Application (SCH)

As shown in Figure 1 above, the SCH application sends periodic commands to LC, as defined in the SCH Schedule Table (not shown), in order to wake up the LC application. The scheduler application only drives the rate Actionpoints are evaluated. Watchpoints are evaluated whenever a packet is received that contains a watchpoint. Additionally, SCH generates a request for LC's housekeeping message(s).

2.2.3 Command Ingest Application (CI)

Ground commands come from the Command Ingest Application (CI), as shown in Figure 1 above. Messages are routed to LC by the cFE Software Bus Service (SB).

2.2.4 Table Services Application (TBL)

LC learns of any ground updates to LC tables through the cFE Table Services application (TBL).

2.2.5 Housekeeping (HK), Telemetry Output (TO)

Messages generated by LC are routed to whatever mission-specific applications subscribe to them, such as Housekeeping (HK), Telemetry Output (TO) Applications, and/or Stored Command (SC) as shown in Figure 1 above.

Figure 2 below shows the internal, overall program flow of the CFS LC application, from start to exit. The B & C circles are highlighted in yellow to indicate a difference in flow control between the original implementation of LC and the latest implementation of LC that adds support for stale watchpoint/actionpoint results.

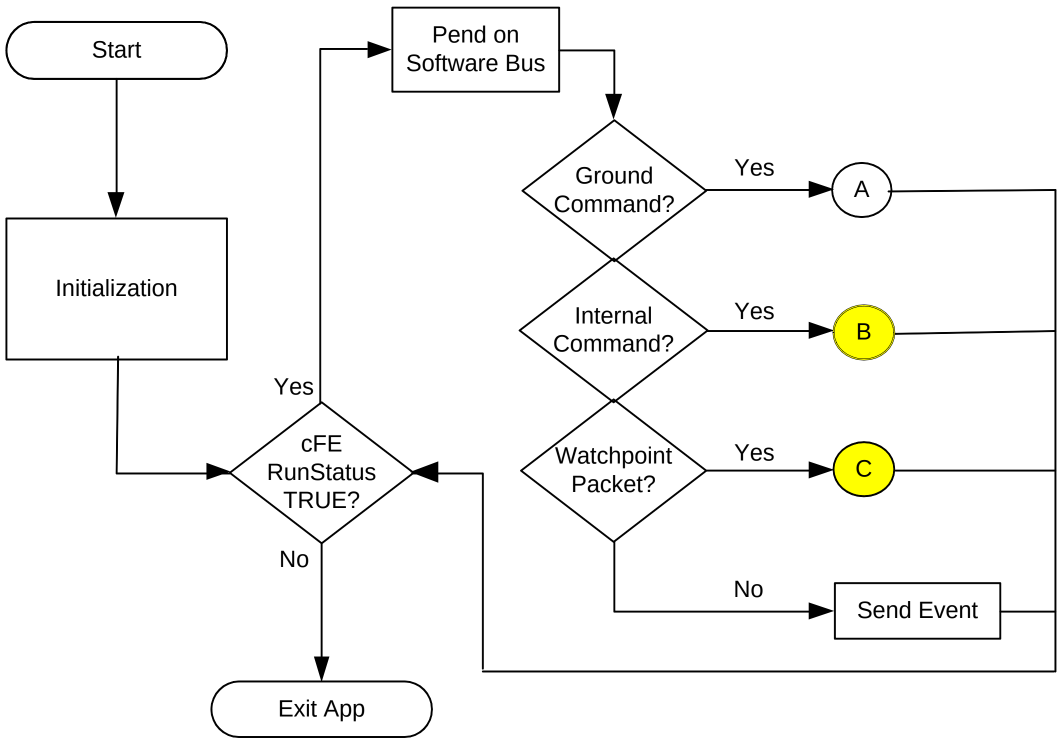


Figure 2 LC Overall Flow Control

Figure 3 below shows the flow control of CFS LC for *ground* commands.

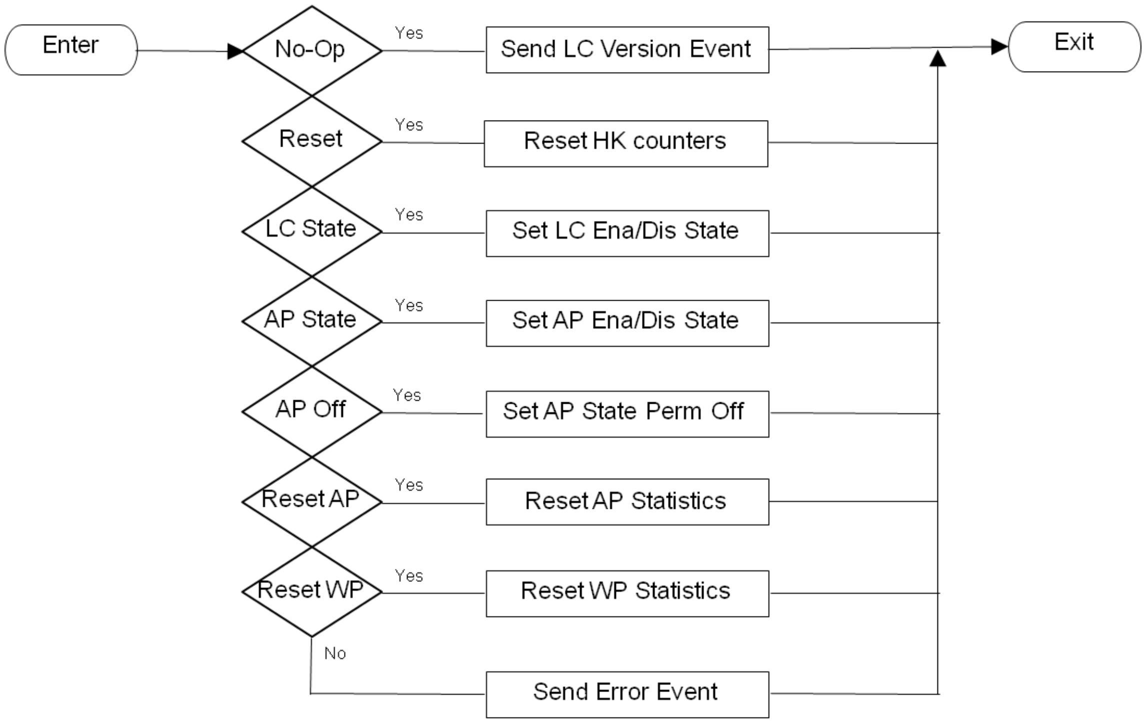


Figure 3 LC Flow Control for Ground Command

Figure 4 below shows the flow control of CFS LC for *internal* commands. The B1 circle is highlighted in yellow to indicate a difference in flow control between the original implementation of LC and the latest implementation of LC that adds support for stale watchpoint/actionpoint results.

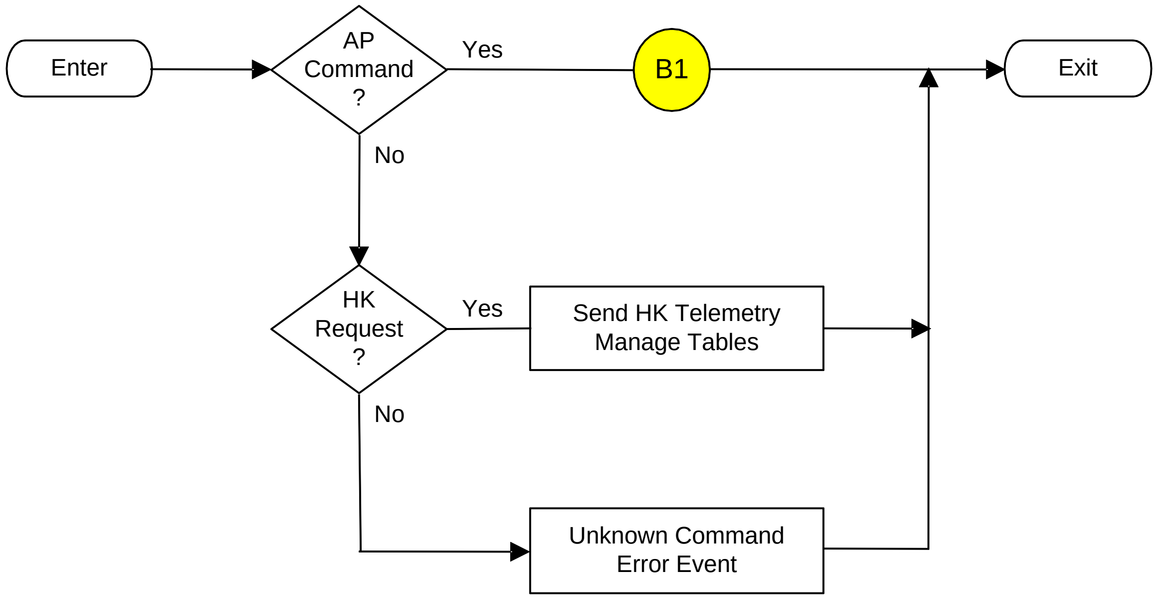


Figure 4 LC Flow Control for Internal Command

Figure 5 LC Flow Control Detail (B1) – Actionpoint Command shows the flow control of CFS LC for *Actionpoint* commands. Evaluating all actionpoints, as shown in Figure 5, is the accepted method, though LC does support issuing a command to evaluate just a single actionpoint. The B2 circle is highlighted in yellow to indicate a difference in flow control between the original implementation of LC and the latest implementation of LC that adds support for stale watchpoint/actionpoint results.

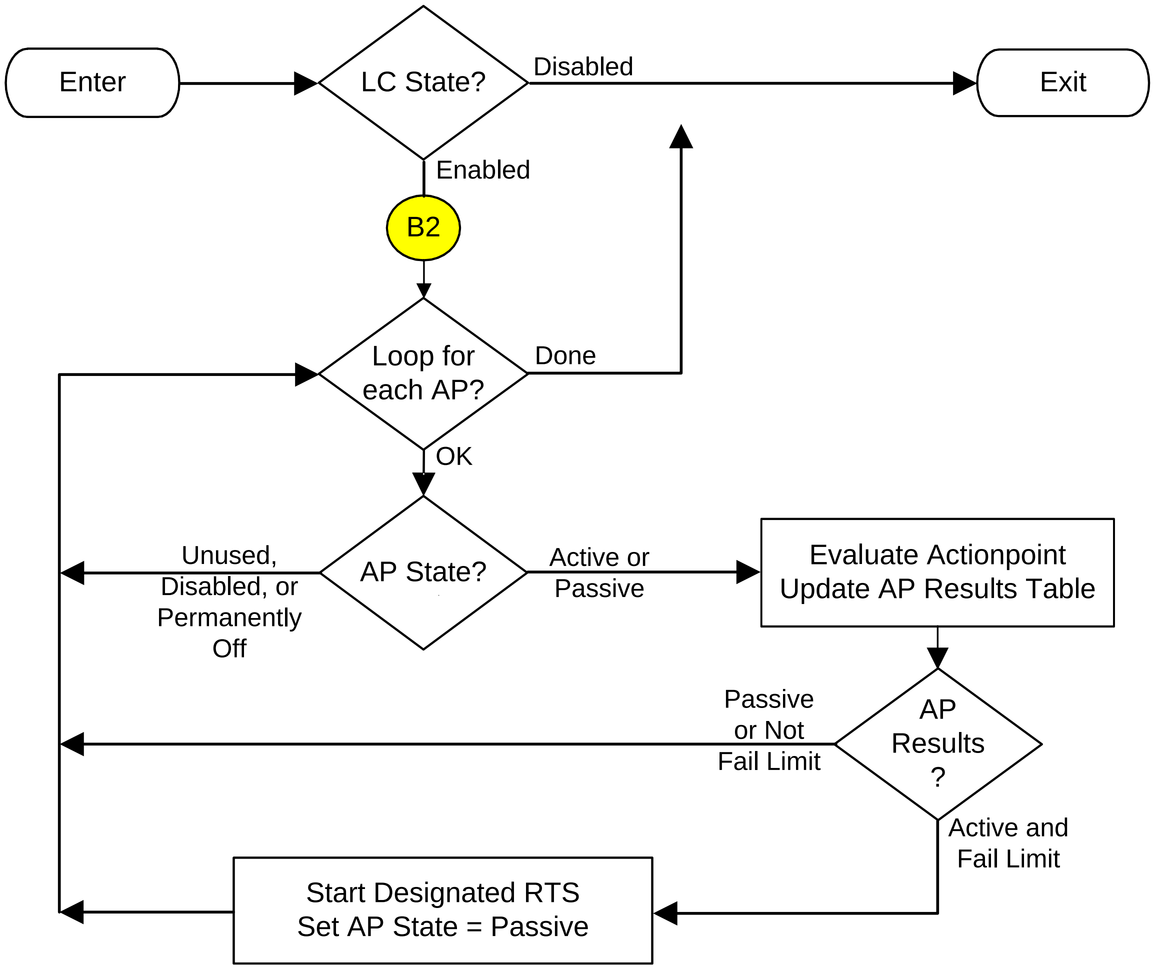

Enter
LC State?
Disabled
Exit
Enabled
B2
Loop for each AP?
Done
OK
AP State?
Unused, Disabled, or Permanently Off
Active or Passive
Evaluate Actionpoint Update AP Results Table
AP Results ?
Passive or Not Fail Limit
Active and Fail Limit
Start Designated RTS Set AP State = Passive

Figure 5 LC Flow Control Detail (B1) – Actionpoint Command

Figure 6 LC Flow Control Detail (B2) – Actionpoint Command below shows LC Flow Detail (B2) to update the WP stale counters.

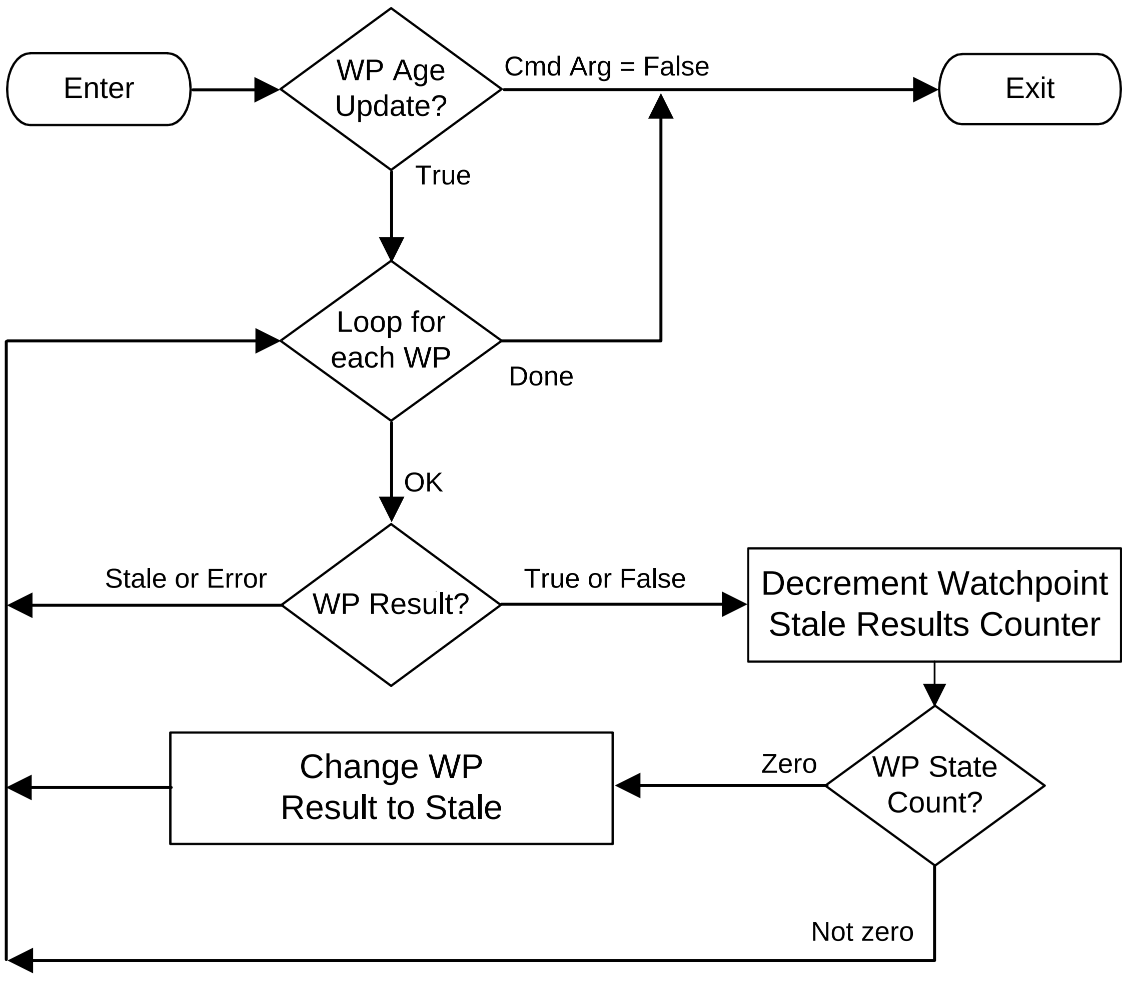


Figure 6 LC Flow Control Detail (B2) – Actionpoint Command

Figure 7 LC Flow Control Detail (C) for Watchpoint Packet below shows the flow control detail (C) of CFS LC for *Watchpoint* packets.

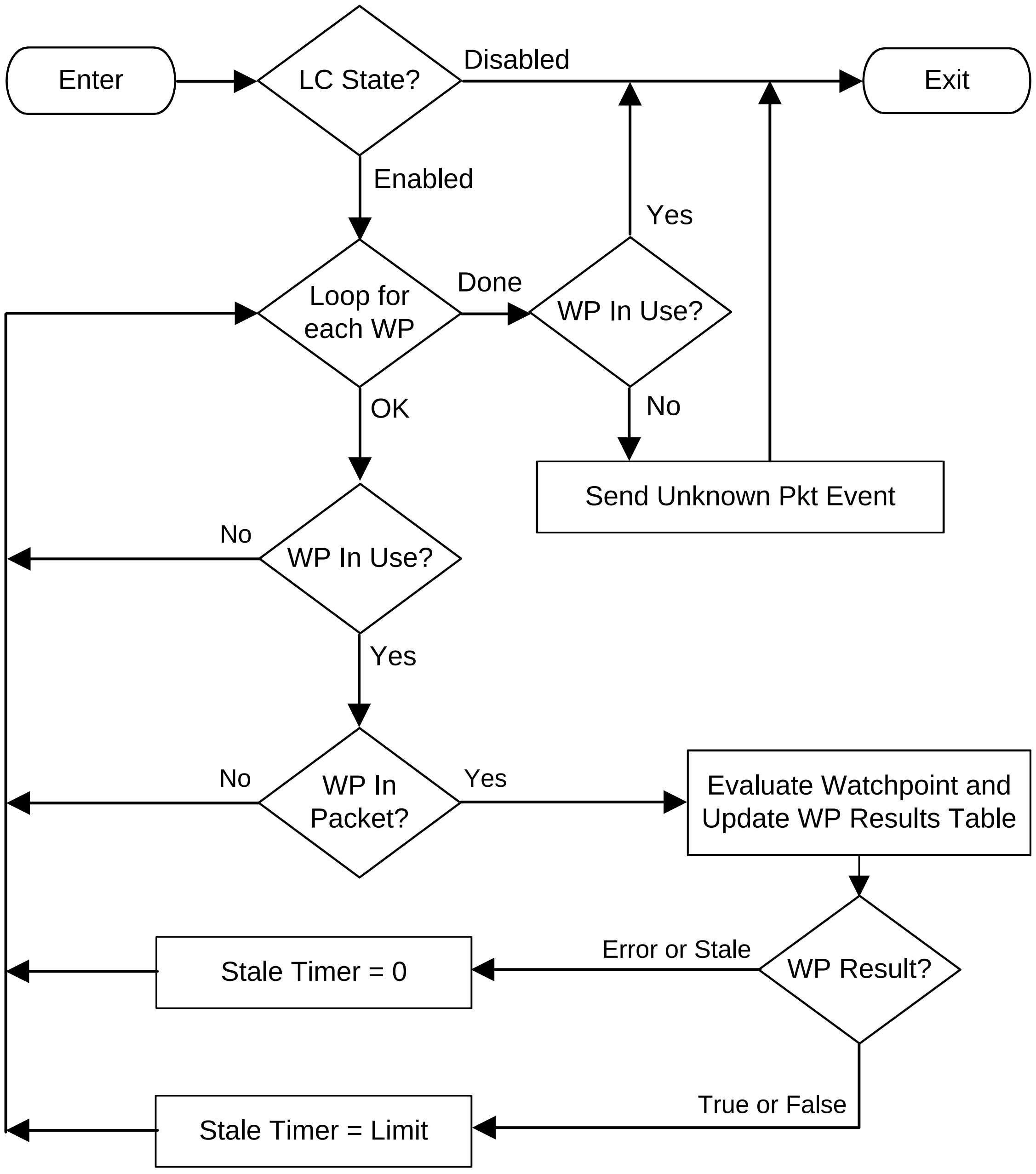


Figure 7 LC Flow Control Detail (C) for Watchpoint Packet

2.2.6 Result Transition to Stale

2.2.6.1 Watchpoint Table

The LC watchpoint definition table specifies a Stale Limit for each watchpoint. The Stale Limit specifies how long after evaluation before the watchpoint result becomes stale.

The watchpoint results table specifies a Stale Counter for each watchpoint. The Stale Counter specifies how much longer before the watchpoint result becomes stale.

2.2.6.2 Watchpoint and Actionpoint Result Initialization

All watchpoint and actionpoint results are initialized to Stale at startup.

2.2.6.3 Watchpoint Result Transition to Stale

The Stale Counter is set to equal the Stale Limit as specified in the WDT when the watchpoint evaluates to True or False.

Stale Counters decrement upon receipt of a scheduled command (assuming a Sample Actionpoints command with "update stale counters" set to true.)

The Watchpoint Result is set to stale when the Stale Counter becomes zero.

Note: The use of zero as the stale limit (ResultAgeWhenStale = 0) will result in bypassing the stale process for the watchpoint. When using zero as the stale limit the WatchResult will never be changed from true or false to stale, since the CountdownToStale will never decrement if ResultAgeWhenStale is already zero at the start of the process.

ResultAgeWhenStale is the age when a Watchpoint comparison result becomes stale. This value is contained in the WDT, as shown in Table 5, **Error! Reference source not found.**, on page A-2.

2.2.6.4 Actionpoint Result Transition to Stale

The Actionpoint Result is set to Stale when the sampled actionpoint has stale watchpoints.

2.2.7 LC – Actionpoint Equations

This section discusses AP Equations.

AP equations consist of operators and operands. The AP operators are AND, OR, XOR, NOT, and EQUAL. AP operands are a single WP result or the result from a previous operator in the equation.

Equations are expressed in Reverse Polish Notation (RPN), also known as postfix notation. For example (23, 24, OR, EQUAL).

- Step 1 – OR the results from WP 23 and WP 24
- Step 2 – compare the result from Step 1 to TRUE
- Equation result = the result from Step 2
- Note: all AP equations must terminate with the EQUAL operator

AP equations use a true result to indicate an error condition.

- AP result = Fail if result of equation = True
- AP result = Pass if result of equation = False
- AP result = Error if result of equation = Error
- AP result = Stale if result of equation = Stale

2.2.8 LC – Actionpoint Operators

Actionpoint Operators consist of AND, OR, XOR, NOT, and EQUAL. Details are discussed below.

2.2.8.1 AP Operator = AND

When the AP Operator is AND there are two operands.

- If either operand evaluates to False, the operator result evaluates to False, even if the other operand is Error or Stale.
- If either operand evaluates to Error, the operator result evaluates to Error.
- If either Operand evaluates to Stale, the Operator result evaluates to Stale.
- Otherwise, the Operator result evaluates to True.

2.2.8.2 AP Operator = OR

When the AP Operator is OR there are two operands.

- If either operand evaluates to True, the operator result evaluates to True, even if the other operand is Error or Stale.
- If either operand evaluates to Error, the operator result evaluates to Error.
- If either Operand evaluates to Stale, the Operator result evaluates to Stale.
- Otherwise, the Operator result evaluates to False.

2.2.8.3 AP Operator = XOR

When the AP Operator is XOR there are two operands.

- If either operand evaluates to Error, the operator result evaluates to Error.
- If either operand evaluates to Stale, the operator result evaluates to Stale.
- If operand 1 equals operand 2, the operator result evaluates to False.
- Otherwise, the Operator result evaluates to True.

2.2.8.4 AP Operator = NOT

When the AP Operator is NOT there is one operand.

- If the operand evaluates to Error, the operator result evaluates to Error.
- If the operand evaluates to Stale, the operator result evaluates to Stale.
- If the operand evaluates to True, the operator result evaluates to False.
- Otherwise, the Operator result evaluates to True.

2.2.8.5 AP Operator = EQUAL

When the AP Operator is EQUAL there is one operand.

- If the operand evaluates to Error, the operator result evaluates to Error.
- If the operand evaluates to Stale, the operator result evaluates to Stale.
- If the operand evaluates to True, the operator result evaluates to True.

Otherwise, the Operator result evaluates to True.

Chapter 3. **CFS LC Normal Operations**

3.1 ***LC Modes of Operation***

The Limit Checker application has three operating modes that can be set via the "Set LC State" ground command: active, passive, and disabled, as explained in the sections below.

No counters are reset when changing mode. (Counters may be reset if the application is reset or if the counters are reset by ground command.)

The three operating modes are explained in the sections below.

3.1.1 Active Mode

Active Mode is the normal operation mode. LC performs all limit tests defined in the watchpoint definition table and invokes stored command sequences as defined in the actionpoint definition table when an actionpoint fails.

3.1.2 Passive Mode

In Passive mode, LC performs all limit tests as in Active mode, but no stored command sequences are invoked as the result of actionpoint failures. Event messages are still generated as they are in the Active mode.

3.1.3 Disabled Mode

In Disabled mode, no watchpoint or actionpoint evaluations take place.

3.2 ***Evaluating Watchpoints and Actionpoints***

Watchpoints are evaluated whenever a packet containing watchpoints arrives. Actionpoints are processed only when an Actionpoint Sample Request is received, as normally sent by the SCH application.

A Sample Request may target one or all actionpoints. The Sample Request is an internal message and will not increment the ground command counter.

Figure 8 LC Monitor Process, Sample Scenario below shows a sample scenario of how the LC monitor process works. An explanation immediately follows.

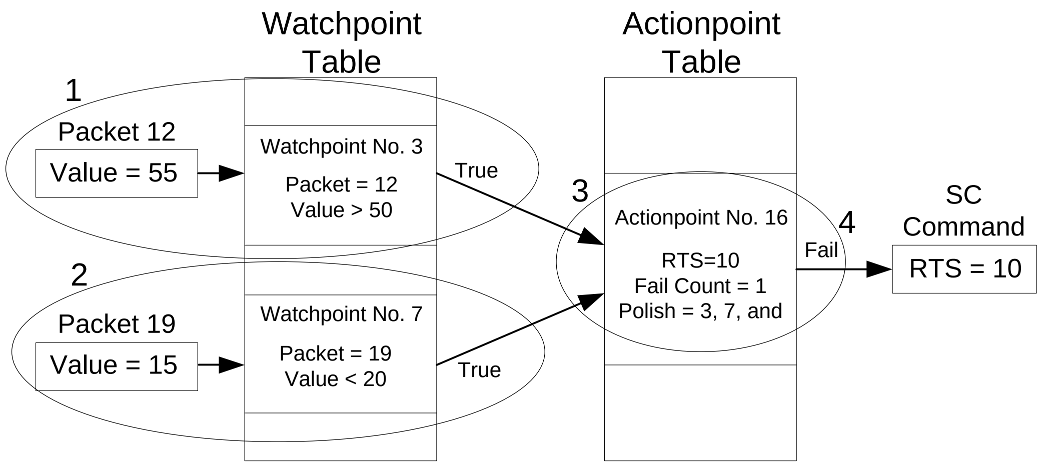


Figure 8 LC Monitor Process, Sample Scenario

1. First, in this example, telemetry packet no. 12 arrives (see section on Timing Issues below). Watchpoint no. 3 results are set to TRUE.
2. Next, telemetry packet no. 19 arrives; watchpoint no. 7 results are set to TRUE.
3. Next, action command arrives. (The sample actionpoint command normally comes from the scheduler at a periodic rate, as specified in the scheduler table, though it can be issued from the ground as well.) Actionpoint no. 16 evaluates to FAIL.
4. Finally, actionpoint no. 16 triggers; LC sends command to start RTS no. 10.

See also: For more, see Table 6, Operator ID Comparison Types; and Table 10, Table 8, and Table 12, starting on page A-3.

3.2.1 Timing Issues

When LC subscribes to a packet it will get that packet from the software bus at whatever frequency that packet is being generated by the source of the data. LC does not control the rate in any way.

Watchpoint states are updated when a packet that contains the watchpoint is received by LC. Actionpoints are evaluated when LC receives a sample actionpoint command, normally from the scheduler. If the sample actionpoint command is at a slower rate than the watchpoint packets, LC could potentially be processing "stale" data.

Note that the LC performs additional processing to avoid "stale" data. LC monitors the age of each watchpoint result; the watchpoint result is changed to Stale when the age of the result exceeds the table defined limit.

3.3 ***Initializing LC***

All watchpoint and actionpoint results are initialized to Stale at startup.

3.3.1 Power-On Reset

On power-on reset, LC performs cFE application specific initialization:

1. Loads default watchpoint and actionpoint definition tables.
2. Initializes watchpoint and actionpoint results tables.

3. Initializes housekeeping data.
4. Subscribes to all packets containing watchpoint data.
5. Goes to the operations mode specified by the configuration parameter (active, passive, or disabled).

3.3.2 Processor Reset

On processor reset, LC performs cFE application specific initialization:

1. Restores watchpoint and actionpoint definition tables from Critical Data Store (CDS) if so configured, otherwise loads default tables.
2. Restores watchpoint and actionpoint results tables from CDS if so configured, otherwise initializes results tables.
3. Restores housekeeping data from CDS if so configured, otherwise initializes housekeeping data.
4. Subscribes to all packets containing watchpoint data.
5. Goes to the operations mode specified by the configuration parameter (active, passive, disabled, or state restored from CDS.

3.4 *Critical Data Store (CDS)*

LC has the ability to save state and continue processing through processor or application resets using the CDS. If the mission has non-volatile memory it can configure the apps to save state so they can pick up where they left off across a restart (or reboot).

Most missions do not use CDS. If the mission does use CDS, results tables and housekeeping are only restored if updated on the last application exit. Restoration of state and initial operating mode is controlled through configuration parameters.

The sections which follow discuss what is restored and when defaults will be used when using the CDS.

When the platform configuration parameter LC_SAVE_TO_CDS is defined, LC will attempt to use the CDS to save and restore data across application restarts. For FSSE, configuration parameters are defined in lc_platform_cfg.h. The data that LC will save are:

- Watchpoint and Actionpoint Definition Tables
- Watchpoint and Actionpoint Results Tables
- LC Housekeeping Data

While the definition tables only need to be updated in the CDS when new ones are loaded, it is not practical from a performance standpoint to update the CDS every time the housekeeping data or results data changes. For this reason, results tables and housekeeping are only updated in the CDS if the application is shut down by cFE Executive Services and exits cleanly.

When the application starts, it will check a "saved on exit" flag in the data restored from CDS to see if the data is good. If not, it will be reset to initialization values.

See also: Table 29 Configuration Parameter - Save to CDS Compiler Switch/LC State When CDS Restored, on page A-15.

If it becomes necessary for LC to attempt to restore from CDS on application startup, LC will proceed as follows.

There are no major implications for the FOT; LC will issue event messages on startup stating what has been restored (or what could not be restored) for the WP/AP definition and results tables.

Step 1: LC will try to restore the Watchpoint Definition Table from CDS

If step 1 fails, then LC will:

a. Load default Watchpoint Definition Table from file system
b. Load default Actionpoint Definition Table from file system
c. Clear Watchpoint Results Table to init values
d. Clear Actionpoint Results Table to init values
e. Clear Housekeeping variables to init values

Step 2: If Step 1 succeeded, LC will try to restore the Actionpoint Definition Table from CDS.

If Step 2 fails, then LC will proceed with the same steps (a through e) as shown in step 1 above.

Step 3: If steps 1 and 2 succeeded, LC will try to restore Watchpoint Results Table from CDS.

If step 3 fails, then LC will proceed with the same steps (a through e) as shown in step 1 above.

Step 4: If steps 1 through 3 succeeded, try to restore Actionpoint Results Table from CDS

If step 4 fails, then LC will proceed with the same steps (a through e) as shown in step 1 above.

Step 5: If steps 1 through 4 succeeded, try to restore Application data (housekeeping variables) from CDS

If step 5 fails, then LC will proceed with the same steps (a through e) as shown in step 1 above.

3.5 ***Event Messages***

There are four levels of event types:

- Critical
- Error
- Information
- Debug

The levels are not used for programmatic control by CFS or LC. In other words, CFS and LC do not treat the level labels differently. However, ASIST or other ground flight control software may display event messages in different colors or otherwise differentiate them.

Debug level messages are traditionally used before launch for testing but then configured to be turned off to avoid cluttering flight operator telemetry.

Event messages can also be **user defined** in the actionpoint table. ***See also:*** Notes in Table 125, Event ID 1000, on page A-47.

3.6 *Message Subscription*

LC keeps a list of message IDs to which it has subscribed.

When a new watchpoint definition table (WDT) is loaded, LC unsubscribes from all previously subscribed messages and then subscribes to all message IDs specified by the watchpoint definitions in the new WDT.

3.7 *Housekeeping Packet Structure*

Housekeeping reference material is located in Appendix A. See Table 14 Watchpoint (WP) Results Housekeeping Telemetry on page A-8; and Table 16 Actionpoint (AP) Results Housekeeping Telemetry on page A-9.

3.8 *Custom Functions*

A custom function is a way to do more complicated processing on a watchpoint when the standard comparison operators are not adequate. A custom function can be used in place of a standard comparison operator in watchpoint definitions.

A custom function returns **True** or **False** as the result of the comparison for the watchpoint that triggered the call. LC can have as many custom functions as monitor points.

Custom functions cannot be created or changed by the FOT. In flight update requires rebuilding, testing, and reloading the LC application.

Mission developers need to modify the LC source code for LC_CustomFunction, though changes are limited to a specific file. This means minimal overhead for missions that do not need custom functions.

3.9 *Task: Creating a Limit Checker Table*

The first step in deploying the LC application is to size and construct default WDT and ADT tables. LC requires these default table images to be on the file system when the application is started or it will fail to load.

Many different personnel need to know how to create a limit checker table. Mission Developers or testers may want to use limit checker itself to verify that LC is working and then initiate checks of other systems. Flight controllers and post-launch software maintenance developers may routinely use tables to initiate or modify limit checker tables for their own purposes.

Missions may maintain their own documentation on how to create Limit Checker tables. Web based ground software tools at NASA Goddard may be used to help with part of the formatting task of creating a limit checker table.

The LC source files lc_def_wdt.c and lc_def_adt.c provide example source code files that can be used to build LC tables and contain some sample table entries in the comment blocks.

Since the CFS LC was based upon the LC implementation for LRO, LRO limit checker tables may also provide guidance though there are some minor differences in syntax between the two implementations.

Step by step instructions of creating a limit checker table from scratch are beyond the scope of this Guide.

See also: CFS Limit Checker Deployment Guide in "Doxygen" html documentation.

Chapter 4. **Understanding LC Operational Constraints**

4.1 ***Startup***

The LC application requires default WDT and ADT table images to be on the file system when the application is started or it will fail to load. These files are required even if using the CDS since LC will fall back to these table images if a table restore from CDS fails (which will happen during a power-on reset). Where LC will look to find these files is dictated by (CFS LC Configuration Parameters LC_WDT_FILENAME and LC_ADT_FILENAME).

Potential causes for LC to terminate prematurely are an error return from the software bus in the application main loop or error return from one of the cFE table services functions when LC tries to do table management (check for updates, dump requests etc.) during each housekeeping cycle.

When using the CDS, the results tables and housekeeping data will only be restored if the application was shut down through cFE Executive Services and exited clean. If this is not the case, default values will be used (see Chapter 5, Frequently Asked Questions (FAQs), starting on page 5-1).

Ideally, LC should require no intervention from the ground on a routine basis. Once monitoring is enabled, it will keep processing data and checking for threshold violations (assuming regular scheduler input).

4.2 ***Unused Entries in WDT***

The entire WDT is NOT searched anytime a message is received that may contain watchpoints. The look up process is optimized to use a hash table instead of searching the entire WDT for entries that reference the received packet.

However, the entire ADT is processed anytime a sample request is received that specifies an actionpoint equal to LC_ALL_ACTIONPOINTS. For this reason, it is important that unused entries are properly marked by setting the ADT parameter DefaultState to Unused. ***See also:*** Table 16, Actionpoint (AP) Results Housekeeping Telemetry, on page A-9.

When either the WDT or ADT are updated, the corresponding results table (WRT or ART) is reset to initialization values. For each entry in the WRT, WatchResult is set to LC_WATCH_STALE and all other values are zeroed. For each entry in the ART, ActionResult is set to LC_ACTION_STALE, the CurrentState is set to the value of the actionpoint's DefaultState specified in the ADT, and all other values are zeroed.

These are also the values used (for the entries specified in the command) when the LC_RESET_AP_STATS_CC [CC stands for Command Code] or LC_RESET_WP_STATS_CC ground command is received.

This page intentionally blank

Chapter 5. **Frequently Asked Questions (FAQs)**

5.1 *FOT Questions*

5.1.1 On cFE power-on, what values does LC use to initialize Actionpoint data?

Upon cFE power-On, LC initializes the following Actionpoint data to the following values:

- The result of the last Actionpoint Sample is set to Stale.
- The current state is set as defined in the ADT.
- The number of times this Actionpoint has crossed from the Fail to Pass state is set to zero.
- The number of times this Actionpoint has crossed from the Pass to Fail state is set to zero.
- The number of consecutive times the equation result equaled Failed is set to zero.
- The cumulative number of times the equation result equaled Failed is set to zero.
- The cumulative count of the RTS executions is set to zero.
- Total number of event messages sent is set to zero.

5.1.2 What happens when I send a reset command?

On receipt of a Reset command, LC resets the following housekeeping variables to a value of zero:

- Valid Command Counter
- Command Rejected Counter
- Passive RTS Execution Counter
- Actionpoint Sample Count
- Telemetry (TLM) Count
- RTS Execution Counter

5.1.3 When updating tables, what should I do with LC?

When updating watchpoint or actionpoint definition tables, consider using the Set LC Application State to Disable command.

Note that changing the LC application state does not reset statistics. However, if the Watchpoint Definition Table is updated, then the Watchpoint Results Table is reset.

Likewise if the Actionpoint Definition Table is updated, the Actionpoint Results Table is reset to initialization values.

5.1.4 What if I want LC to monitor data but not take action?

If as the operator you need LC to monitor telemetry and report statistics but take no actions, use the Set LC Application State to Passive command.

5.1.5 How do I set LC back to normal operating mode?

Send the Set LC Application State to Active command. LC then sets the state of the LC Application to Active Nominal mode.

5.1.6 What is Active Nominal mode?

In Active Nominal mode, LC performs all limit tests defined in the watchpoint definition table and invokes stored command sequences if the results of watchpoint evaluations trigger an actionpoint condition as defined in the actionpoint definition table.

5.1.7 Can I set the state of individual actionpoints to Active, Passive or Disable?

Yes.

5.1.8 Can I set the state of individual watchpoints?

No. There is no capability to set the state of watchpoints, only actionpoints.

5.1.9 What happens when LC receives a no-op command?

When LC receives a no-op command, LC increments the LC Valid Command Counter and generates an event

5.1.10 How is stale data defined?

The goal of tracking stale data by LC is to prevent LC from taking action on old or inaccurate information. LC is designed with stale data defined as datum for which a packet update has not been received and/or has not been received in a requisite interval specified by the age value. In most cases, it indicates that a subsystem is turned off or some other condition has prevented packet generation.

5.1.11 In LC, what value are Watchpoint and Actionpoint Results assigned at Initialization?

For each Watchpoint specified in the Watchpoint Definition Table (WDT) LC specifies an age value which indicates when the data becomes "stale". However, in LC, all watchpoint and actionpoint results are initialized to Stale at startup.

5.1.12 How do I activate or deactivate the Stale process for individual watchpoints?

To activate the stale process on individual watchpoints, use nonzero values for the stale limit. That is, for individual watchpoints, set ResultAgeWhenStale to a nonzero value at the start of the process. *ResultAgeWhenStale* is the age when a Watchpoint comparison result becomes stale. This value is contained in the WDT, as shown in Table 5, **Error! Reference source not found.**, on page A-**Error! Bookmark not defined.**.

To turn off the stale process for individual watchpoints, use zero as the stale limit. That is, for individual watchpoints, set ResultAgeWhenStale to zero (0) at the start of the process. This shuts down the stale process for this watchpoint, because the WatchResult never changes from true or false to stale. Why? Because the CountdownToStale never decrements if ResultAgeWhenStale is already zero at the start of the process.

5.1.13 How does LC track stale data?

For each watchpoint specified in the Watchpoint Definition Table (WDT) LC maintains the age of the data.

5.2 *Questions for Creators of Watchpoints and Actionpoints*

While the questions in this section generally do not apply to the FOT (the FOT generally does not build watchpoint or actionpoint tables), they are included to help the FOT better understand those tables.

5.2.1 Can I send a single actionpoint evaluation message for multiple actionpoints (eg. 1,2,5,6,7)?

No. You can send individual messages for each action point, a single message to evaluate ALL (xFFFF) actionpoints, or a contiguous range. See *AP Sample Request* in Table 3, Internal Messages, on page A-1.

5.2.2 What if most of my actionpoints need to be evaluated at 1HZ but one needs to be evaluated at 10HZ. How would I setup the Scheduler table to do this?

In order to evaluate an actionpoint, an evaluation message must be placed in the scheduler table. Since the evaluation frequency of each entry in the scheduler table is defined in seconds (1 Hz being the highest frequency), the suggested way to accomplish this is to place nine entries for the 10 Hz actionpoint and a single entry to evaluate ALL actionpoints. Note that the ALL actionpoints message needs to be placed 100ms away from the other nine (9).

5.2.3 Watchpoints evaluate to LC_WATCH_TRUE or LC_WATCH_FALSE. When the watchpoint is defined which should be the error condition?

Watchpoints evaluate to a Boolean True or False. So if you setup a comparison < 100 any watchpoint value zero (0) to 99 will evaluate LC_WATCH_TRUE and 100+ will be LC_WATCH_FALSE.

You should construct the watchpoint cases so they evaluate True when you are outside of the acceptable range for a telemetry point. That is why there are cumulative and consecutive true counts in the watchpoint results table, but no corresponding values for False.

For FSSE, some examples can be found in the CFS LC source file lc_def_wdt.c

5.2.4 There is only a single comparison value for each watchpoint. How can I have multiple thresholds for a single telemetry point?

Create additional watchpoint definitions that reference the same telemetry point but have different comparison values. The watchpoints will be evaluated in the order they are listed in the WDT when the monitor point arrives.

Having multiple watchpoints allows different Actionpoints to trigger on each watchpoint state (or combination of states).

5.2.5 How do I calculate the watchpoint offset?

The offset is a zero based byte offset from the beginning of the message (including any headers) to the first byte of the watchpoint data. So for a cFE raw command using CCSDS, the offset has to account for the size of the cFE command header (CFE_SB_CMD_HDR_SIZE).

5.2.6 When do I need to use the bitmask in a watchpoint definition?

All watchpoints are sized to a 32 bit value when extracted from a message. The specified bitmask value is then applied (as a bitwise AND operation) before the comparison is made or the custom function is called.

When the watchpoint data is sized, data types smaller than 32 bits are properly sign or zero extended. For this reason, it is not necessary to define a mask for UWORD, WORD or UBYTE, BYTE data types to compare properly. (Although it is not needed, a properly constructed bit mask will not cause a problem for these evaluations.)

When you really need to use a bitmask is when monitoring odd sized data (such as a 24 bit sensor reading) or testing data to see if certain bits have been set (or cleared).

When no bitmask is needed, be sure to use the constant LC_NO_BITMASK (or its equivalent value 0xFFFFFFFF) in your WDT tables. Since masking is a bitwise AND, setting the bitmask to zero will have the effect of always clearing the watchpoint data prior to comparison.

5.2.7 Why do I need to specify the byte order (big or little Endean) of the watchpoint data type?

The CFS version of LC will properly byte swap watchpoint data prior to masking and comparison if the byte order of the data is different than the order used by the processor running LC. This allows LC to monitor telemetry data that might be constructed by instrument or other subsystem processors that use a different byte order.

LC determines its byte order at the time the code is compiled, and this cannot be changed by the FOT.

5.2.8 When would a watchpoint evaluate to Stale?

LC_WATCH_STALE is an initialization value for the Watchpoint Results Table. If a watchpoint has this WatchResult then the watchpoint is unused (the DataType in the WDT is set to LC_WATCH_NOT_USED), or a message that contains the watchpoint has not yet been received by LC and evaluated. For more, see the next question below.

5.2.9 How exactly does a watchpoint result transition from True or False to Stale?

To fully understand how an actionpoint evaluates to LC_ACTION_STALE, one needs to understand how a watchpoint result transitions from LC_WATCH_TRUE or LC_WATCH_FALSE to LC_WATCH_STALE.

When any watchpoint evaluates to true or false, the corresponding WatchResult is set to LC_WATCH_TRUE or LC_WATCH_FALSE and CountdownToStale is set to ResultAgeWhenStale.

Thereafter, each time a Sample Actionpoints command is processed, each non-zero watchpoint CountdownToStale is decremented.

If the decremented CountdownToStale becomes zero, then that WatchResult is set to LC_WATCH_STALE.

5.2.10 When would a watchpoint evaluate to LC_WATCH_ERROR?

LC_WATCH_ERROR is a runtime error indicator for watchpoint processing. Such a result should be rare since most of the causes are invalid watchpoint parameters that should be caught during validation of the Watchpoint Definition Table.

One notable exception is for floating point watchpoints when the message data is detected to be a floating point NAN (Not A Number) that cannot be relationally compared to any value.

In all cases, an error event detailing the cause of the problem will be issued when a WatchResult is set to LC_WATCH_ERROR.

5.2.11 How is the WPResults array in the housekeeping packet (LC_HkPacket_t) interpreted?

The WPResults array is a byte array (aligned to the nearest longword boundary) that contains a packed subset of the current contents of the Watchpoint Results Table.

The WPResults array allocates 2 bits per watchpoint for the most recent watchpoint comparison result. The numerical 2 bit values are defined using the following constants:

LC_HKWR_FALSE

LC_HKWR_TRUE

LC_HKWR_ERROR

LC_HKWR_STALE.

Ordering [up to (CFS LC Configuration Parameter LC_MAX_WATCHPOINTS)] is as follows:

Byte 0:(Rwp3, Rwp2, Rwp1, Rwp0), Byte 1:(Rwp7, Rwp6, Rwp5, Rwp4), etc...

The WPResults array is constructed every housekeeping cycle and is not affected by the reset counters (LC_RESET_CC) ground command. It will only get cleared if the Watchpoint Results Table is reset (via a new WDT table load or with a LC_RESET_WP_STATS_CC ground command).

5.2.12 Actionpoints evaluate to LC_ACTION_PASS or LC_ACTION_FAIL. How should the RPN expression get constructed so it will evaluate to the proper result?

Actionpoint Reverse Polish Notation (RPN) expressions are combinations of watchpoint states and logical operators that evaluate to a boolean True or False.

Just like a watchpoint evaluation of True indicates a parameter outside acceptable limits, an AP expression that evaluates True is considered to have Failed and will have its ActionResult set to LC_ACTION_FAIL.

While the terminology may seem confusing, the key point is to construct both watchpoint and actionpoint expressions to define the error condition LC is looking for and NOT the normal condition of the spacecraft data stream.

For some examples, see the CFS LC source file lc_def_adt.c

5.2.13 When would an actionpoint evaluate to Stale?

LC_ACTION_STALE is an initialization value for the Actionpoint Results Table. If an actionpoint has this ActionResult then one of three possible conditions are true:

1. The actionpoint is unused (the DefaultState in the ADT is set to LC_ACTION_NOT_USED).
2. An actionpoint sample request (LC_SAMPLE_AP_MID) targeting the AP has not yet been received by LC so the AP has not yet been evaluated.
5. One or more of the watchpoints that this AP depends on (as defined by the RPN expression) has a current WatchResult of LC_WATCH_STALE so the AP can't be evaluated.

5.2.14 When would an actionpoint evalute to LC_ACTION_ERROR?

LC_ACTION_ERROR is a runtime error indicator for actionpoint processing. Such a result should be rare since most of the causes are invalid actionpoint parameters or improperly constructed RPN expressions that should be caught during validation of the Actionpoint Definition Table.

However, an exception is the case where one or more watchpoints that this AP depends on (as defined by the RPN expression) has a current WatchResult of LC_WATCH_ERROR. Since the AP cannot be evaluated, this will cause the ActionResult to be set to LC_ACTION_ERROR.

In all cases, an error event detailing the cause of the problem will be issued when an ActionResult is set to LC_ACTION_ERROR.

5.2.15 How does the Actionpoint state LC_APSTATE_PERMOFF differ from LC_APSTATE_DISABLED?

The AP state LC_APSTATE_PERMOFF is intended to provide a way to disable an AP so it cannot easily be turned back on by mistake. Such actionpoints may not be needed after a separation sequence or only apply to certain mission phases.

While the two states are treated the same way during actionpoint processing (the AP is not evaluated), there are a few differences.

An AP cannot be set to LC_APSTATE_PERMOFF with the LC_SET_AP_STATE_CC command; it must be done with the LC_SET_AP_PERMOFF_CC command.

To set an AP to LC_APSTATE_PERMOFF with the LC_SET_AP_PERMOFF_CC command, the current AP state must be LC_APSTATE_DISABLED.

The LC_SET_AP_PERMOFF_CC command can only be issued for a single actionpoint, LC_ALL_ACTIONPOINTS is not valid as an argument for this command.

Once an AP is set to LC_APSTATE_PERMOFF, it can only be changed with a new ADT table load.

5.2.16 How is the APResults array in the housekeeping packet (LC_HkPacket_t) interpreted?

The APResults array is a byte array (aligned to the nearest longword boundary) that contains a packed subset of the current contents of the Actionpoint Results Table (see the Actionpoint Results Table Structure).

It allocates 4 bits per actionpoint, with 2 bits representing the current state, and 2 bits for the most recent evaluation result.

The numerical 2 bit values for current state are defined using the following constants:

- LC_HKAR_STATE_NOT_USED
- LC_HKAR_STATE_ACTIVE
- LC_HKAR_STATE_PASSIVE
- LC_HKAR_STATE_DISABLED

An actionpoint whose current state is LC_APSTATE_PERMOFF will have its state reported in the APResults as LC_HKAR_STATE_NOT_USED.

The numerical two (2) bit values for evaluation results are defined using the following constants:

- LC_HKAR_PASS
- LC_HKAR_FAIL
- LC_HKAR_ERROR
- LC_HKAR_NOT_MEASURED

Ordering (up to LC_MAX_ACTIONPOINTS) is as follows :

- Byte 0: (Sap1, Rap1, Sap0, Rap0)
- Byte 1:(Sap3, Rap3, Sap2, Rap2)
- etc...

The APResults array is constructed every housekeeping cycle and is not affected by the reset counters (LC_RESET_CC) ground command. It will only get cleared if the Actionpoint Results Table is reset (via a new ADT table load or with a LC_RESET_AP_STATS_CC ground command)

5.2.17 Will an RTS get requested more than once if an AP stays in the LC_ACTION_FAIL state?

No. Assuming the current state of an actionpoint is LC_APSTATE_ACTIVE, then when the actionpoint fails enough times to trigger an RTS, the state is set to LC_APSTATE_PASSIVE.

In the passive state, the AP will continue to be sampled and statistics updated, but no RTS requests will be initiated.

5.2.18 Can we filter event messages per actionpoint?

Not in the current implementation. Transition event messages for actionpoints are of type CFE_EVS_DEBUG and enabling them will turn on events for all actionpoint transitions.

5.2.19 When do results tables get cleared?

When either the WDT or ADT are updated, the corresponding results table (WRT or ART) is reset to initialization values. For each entry in the WRT, WatchResult is set to LC_WATCH_STALE and all other values are zeroed. For each entry in the ART, ActionResult is set to LC_ACTION_STALE, the CurrentState is set to the value of the actionpoint's DefaultState specified in the ADT, and all other values are zeroed.

These are also the values used (for the entries specified in the command) when the LC_RESET_AP_STATS_CC or LC_RESET_WP_STATS_CC ground command is received.

This page intentionally blank

Appendix A CFS LC Reference

A.1 Internal Messaging

Table 3 below shows Internal Messages.

Table 3 Internal Messages

Message	Flow	Description
Housekeeping Request	Input to LC	Send housekeeping data. (See LC_SEND_HK_MID in Table 4 below.)
AP Sample Request	Input to LC	Sample one, all, or a range of actionpoints. Source of request is transparent to LC. Normally sent internally but can also be sent from the ground. [Optionally update age of WP results. ***See also:*** **Error! Reference source not found.**, **Error! ference source not found.**, **Error! Reference source not found.**.] ***See also:*** LC_SAMPLE_AP_MID in Table 4 below.
Start RTS	Output from LC	Sent when an actionpoint failure needs to initiate an RTS.

Table 4 below shows Message IDs.

Table 4 Message IDs

Name	Value	Description
LC_HK_TLM_MID	0x08A7	LC Housekeeping Telemetry.
LC_SAMPLE_AP_MID	0x18A6	Msg ID to request actionpoint sample.
LC_SEND_HK_MID	0x18A5	Msg ID to request LC housekeeping.

A.2 Watchpoints and Actionpoints

A.2.1 Watchpoint Definitions

Table 5 below shows a full definition of a single watchpoint. (The Watchpoint Definitions Table (WDT) is a series of watchpoint definitions. There is only one WDT.)

Table 5 Watchpoint Definition Table (WDT) Entry

Element	**Type**	**Name**	**Description**
Watchpoint Data Type	uint8	DataType	Watchpoint Data Types: • Unused • Byte • UByte (8 bit unsigned byte) • Word • UWord (16 bit signed word) • DWord (32 bit signed double word) • UDWord • Float
Watchpoint Operator ID	uint8	OperatorID	Comparison type (enumerated). ***See also:*** Table 6 below for Operator ID Comparison Types.
Watchpoint Message ID	uint16	MessageID	Message ID for the message containing the watchpoint
Watchpoint Data Offset	uint32	WatchpointOffset	Zero based byte offset from the beginning of the message (including any headers) to the first byte of the watchpoint data
Watchpoint Bit Mask	uint32	BitMask	Value to be masked with watchpoint data prior to comparison. Use the constant LC_NO_BITMASK when no masking is desired. ***See also:*** Table 7 below for Watchpoint Definition Table (WDT) Enumerated Types.
Watchpoint Comparison Value	28 byte struct	ComparisonValue	Value against which watchpoint data is compared. This field uses the LC_MultiType_t union to store different data types in a fixed 32-bit field. See lc_def_wdt.c in the CFS LC source code for examples of how to set this value.
Age When Watchpoint Comparison Result Becomes Stale	uint32	ResultAgeWhenStale	Units are the number of LC "sample actionpoints" commands since comparison.
Custom Function Argument	uint32	CustomFuncArgument	Optional 32 bit data to be passed to the custom function. Can be used for any mission-defined purpose. Must be set up before program is compiled. Generally not changeable by the FOT.

Table 6 below lists the possible values for the operator ID comparison type, which is one of the elements in a watchpoint definition.

Table 6 Operator ID Comparison Types

Element	CFS LC Value	Description	
LC_NO_OPER	0xFF	Can be used for unused entries (optional)	Unused (optional)
LC_OPER_LT	1	Less Than	<
LC_OPER_LE	2	Less Than or Equal	<=
LC_OPER_NE	3	Not Equal	!=
LC_OPER_EQ	4	Equal	=
LC_OPER_GE	5	Greater Than or Equal	>=
LC_OPER_GT	6	Greater Than	>
LC_OPER_CUSTOM	7	No compare, call custom function	Custom

Table 7 Watchpoint Definition Table (WDT) Enumerated Types

Internal Name	CFS LC Default Value	Description
LC_NO_BITMASK	0xFFFFFFFF	Use for no bitmasking.
LC_NO_OPER	0xFF	Use for empty entries.
LC_OPER_CUSTOM	7	Use custom function.

A.2.2 Actionpoint Definitions

Table 8 below shows descriptions of elements in Actionpoint definitions tables. The structure of a single actionpoint definition is defined by the Actionpoint Definition Table Entry (see table below). The ADT is an array of these entries sized by the configuration parameter LC_MAX_ACTIONPOINTS. The zero based index into this table is used by LC as the Actionpoint ID.

Table 8 Actionpoint Definition Table (ADT) Entry

Element	Description
Default State	Default state for this AP: Unused (0xFF) Active (1) Passive (2) Disabled (3) Permanently off (4) *To set this state to permanently off requires using the LC_SET_AP_PERMOFF_CC command.*

Element	Description
Max Passive Events	Max number of events before filter RTS not started because AP is passive.
Max Pass to Fail Events	Max number of events before filter AP result transition from pass to fail.
Max Fail to Pass Events	Max number of events before filter AP result transition from fail to pass.
RTS ID	RTS to request if this AP fails
Max Fails Before RTS	How may consecutive failures before an RTS request is issued
RPN Equation	Expression in Reverse Polish Notation that specifies when this actionpoint should fail
	One (or more) watchpoint IDs combined with "and," "or," "xor," "not," or "equals". ***See also:*** Table 9 below for RPN Operators.
Event Type	Event type to use for event msg if AP fails: Informational Debug Error Critical
Event ID	Event ID used for the event msg if the AP fails.
Event Text	Text used for the event msg when this AP fails.

Table 9 RPN Operators

Name	Value
LC_RPN_EQUAL	0xFFF5
LC_RPN_NOT	0xFFF4
LC_RPN_OR	0xFFF2
LC_RPN_XOR	0xFFF3

A.2.3 Watchpoint Results Structure

Table 10 below shows a full definition of a single watchpoint results table entry; a WRT is a series of watchpoint results definitions. The WRT is an array of single watchpoint results table entries sized by (CFS LC Configuration Parameter LC_MAX_WATCHPOINTS). The index into this table is the same Watchpoint ID used for the corresponding definition table entry.

Table 10 Watchpoint Results Table (WRT) Entry

WRT Field Name	Description
Watchpoint Evaluation Result	Result for the last evaluation of this watchpoint. Possible Values: True (1) False (0) Error (2) Stale (0xFF)
Age Countdown Until Watchpoint Result Becomes Stale	Age units are the number of LC "sample actionpoints" commands
Cumulative Evaluation Count	Total number of times this watchpoint has been evaluated
Cumulative False To True Count	Total number of times this watchpoint has transitioned from False to True
Consecutive True Count	Number of consecutive times this watchpoint has evaluated to True
Cumulative True Count	Total number of times this watchpoint has evaluated to True
Most Recent Watchpoint Result Transition From False to True	Spacecraft time and watchpoint data when transition occurred. • Watchpoint value and timestamp at the last transition from 0 (False) or Stale (0xFF) to 1 (True) • (The timestamp used for the LastFalseToTrue field is taken from the header of the message that contained the watchpoint. If the message timestamp is zero, LC will use the time returned by the CFE_TIME_GetTime function instead.)
Most Recent Watchpoint Result Transition From True to False	Spacecraft time and watchpoint data when transition occurred • Watchpoint value and timestamp at the last transition from 1 (True) to 0 (False). • (The timestamp used for the LastTrueToFalse field is taken from the header of the message that contained the watchpoint. If the message timestamp is zero, LC will use the time returned by the CFE_TIME_GetTime function instead.)

Table 11 Watchpoint Definition Table (WDT) Validation Error Enumerated Types

Name	CFS LC Value	Description
LC_WDTVAL_NO_ERR	0	No error.
LC_WDTVAL_ERR_DATATYPE	1	Invalid DataType.
LC_WDTVAL_ERR_OPER	2	Invalid OperatorID.
LC_WDTVAL_ERR_MID	3	Invalid MessageID.
LC_WDTVAL_ERR_FPNAN	4	ComparisonValue is NAN float.
LC_WDTVAL_ERR_FPINF	5	ComparisonValue is infinite float.

A.2.4 Actionpoint Results

Table 12 below shows the descriptions for ARTs.

Table 12 Actionpoint Results Table (ART) Entry

Element	Description
Actionpoint Evaluation Result	Most recent evaluation result Pass (0) Fail (1) Error (2) Stale (0xFF)
Actionpoint Current State	Current state of this actionpoint Unused Active Passive Disabled Permanently Off
Cumulative Fail To Pass Count	Total number of times this actionpoint has transitioned from Fail to Pass
Cumulative Pass To Fail Count	Total number of times this actionpoint has transitioned from Pass to Fail
Consecutive Fail Count	Number of consecutive times this actionpoint has evaluated to Fail
Cumulative Fail Count	Total number of times this actionpoint has evaluated to Fail
Cumulative RTS Exec Count	Total number of times an RTS request has been sent for this actionpoint

A.2.5 Housekeeping Telemetry

Table 13 below shows the Housekeeping (HK) Packet Telemetry Structure.

Table 13 Housekeeping (HK) Packet Telemetry Structure

Type	Name	Description	CFS LC Default Telemetry Mnemonic(s)	Mission-Specific Telemetry Mnemonic
uint16	ActiveAPs	How many actionpoints are currently set active.	$sc_$cpu_LC_ActiveAPs	Mission Defined
uint8	APResults [LC_HKAR_NUM_BYTES]	Packed actionpoint results data, 4 bits per actionpoint. ***See also:*** Table 16 below (on Page A-9).	$sc_$cpu_LC_APResults	Mission Defined
uint32	APSampleCount	Total count of Actionpoints sampled.	$sc_$cpu_LC_APSample Cnt	Mission Defined
uint16	CmdCount	LC Application Command Counter. Number of accepted ground commands	$sc_$cpu_LC_CMDPC	Mission Defined
uint16	CmdErrCount	LC Application Command Error Counter. Number of rejected ground commands	$sc_$cpu_LC_CMDEC	Mission Defined
uint8	CurrentLCState	Current LC application operating state (Active, Passive, Disabled)	$sc_$cpu_LC_CurrentLCS tate	Mission Defined
uint32	MonitoredMsgCount	Total count of messages monitored for watchpoints.	$sc_$cpu_LC_MonMsgCn t	Mission Defined
uint16	Pad16			
uint8	Pad8[3]			

Type	Name	Description	CFS LC Default Telemetry Mnemonic(s)	Mission-Specific Telemetry Mnemonic
uint16	PassiveRTSExecCount	Total count of RTS sequences not initiated because either the LC application state or the state of the actionpoint that failed is set to Passive.	$sc_$cpu_LC_PassRTSCnt	Mission Defined
uint32	RTSExecCount	Total count of RTS sequences initiated.	$sc_$cpu_LC_RTSCnt	Mission Defined
uint8	TlmHeader [CFE_SB_TLM_HDR_SIZE]	cFE SB Tlm Msg Hdr		
uint8	WPResults [LC_HKWR_NUM_BYTES]	Packed watchpoint results data, 2 bits per watchpoint. ***See also:*** Table 14 below.	$sc_$cpu_LC_WPResults	Mission Defined
uint16	WPsInUse	How many watchpoints are currently in effect, that is, how many watchpoints are currently defined.	$sc_$cpu_LC_WPsInUse	Mission Defined

Table 14 below shows descriptions of the packed subset of the **Watchpoint** Results Table.

Table 14 Watchpoint (WP) Results Housekeeping Telemetry

WPResults	Description
Byte array	Two bits per watchpoint (aligned to nearest longword boundary)
Most recent watchpoint comparison result (2 bits)	0 = False (0x00) 1 = True (0x01) 2 = Error (0x02) 3 = Stale (0x03)
Ordering	(Rwp3, Rwp2, Rwp1, Rwp0), (Rwp7, Rwp6, Rwp5, Rwp4), etc...

Table 15 Watchpoint Definition Table (WDT) Validation Error Enumerated Types

Name	CFS LC Value	Description
LC_WDTVAL_NO_ERR	0	No error.
LC_WDTVAL_ERR_DATATYPE	1	Invalid DataType.
LC_WDTVAL_ERR_OPER	2	Invalid OperatorID.
LC_WDTVAL_ERR_MID	3	Invalid MessageID.
LC_WDTVAL_ERR_FPNAN	4	ComparisonValue is NAN float.
LC_WDTVAL_ERR_FPINF	5	ComparisonValue is infinite float.

Table 16 below shows descriptions of the packed subset of the **Actionpoint** Results Table.

Table 16 Actionpoint (AP) Results Housekeeping Telemetry

APResults	Description
Byte array	Four bits per actionpoint (aligned to nearest longword boundary)
Actionpoint current state (2 bits)	0 = Unused or Permanently Off (0x00) 1 = Active (0x01) 2 = Passive (0x02) 3 = Disabled (0x03)
Most recent actionpoint analysis result (2 bits)	0 = Pass (0x00) 1 = Fail (0x01) 2 = Error (0x02) 3 = Stale (0x03)
Ordering	(Sap1, Rap1, Sap0, Rap0), (Sap3, Rap3, Sap2, Rap2), etc...

Table 17 Actionpoint Definition Table (ADT) Validation Error Enumerated Types

Name	Value	Description
LC_ADTVAL_NO_ERR	0	No error.
LC_ADTVAL_ERR_DEFSTATE	1	Invalid DefaultState.
LC_ADTVAL_ERR_RTSID	2	Invalid RTS Id.
LC_ADTVAL_ERR_FAILCNT	3	Max Fails Before RTS is zero.
LC_ADTVAL_ERR_EVTTYPE	4	Invalid EventType.
LC_ADTVAL_ERR_RPN	5	Invalid Reverse Polish (RPN) Expression.

A.3 Configuration Parameters

This section shows the LC Configuration Parameters provided as a default by the CFS LC application. Space is provided below for these values to be updated by the mission. If the mission has configured non-default values, they will be documented in the flight software RDL database.

While configuration parameters cannot be changed by the FOT, and are generally never changed after launch except by FSSE, the FOT need to know the mission-specific values that have been incorporated into the software at the time the software was finalized and compiled.

(From an FSSE standpoint, configuration parameters are defined in the source code in lc_platform_cfg.h)

Table 18 Configuration Parameter - Actionpoint Definition Table (ADT) Filename

Configuration Parameter	LC_ADT_FILENAME
CFS LC Default	"/boot/lc_def_adt.tbl"
Mission-Specific Default	Mission Defined
Purpose	Actionpoint Definition Table (ADT) filename
Description	Default file to load the actionpoint definition table from during a power-on reset sequence
Limits	This string should not be longer than OS_MAX_PATH_LEN for the target platform in question

Table 19 Configuration Parameter - Application Name

Configuration Parameter	LC_APP_NAME
CFS LC Default	"LC"
Mission-Specific Default	Mission Defined
Purpose	Application Name
Description	This definition must match the name used at startup by the cFE Executive Services when creating the LC application. Note that application names are also an argument to certain cFE commands. For example, the application name is needed to access tables via cFE Table Services commands.

Limits	LC requires that this name be defined, but otherwise places no limits on the definition. Refer to cFE Executive Services for specific information on limits related to application names.

Table 20 Configuration Parameter - Floating Point Compare Tolerance

Configuration Parameter	LC_FLOAT_TOLERANCE
CFS LC Default	(1.0e-25)
Mission-Specific Default	Mission Defined
Purpose	Floating Point Compare Tolerance
Description	Difference between two (2) floats that will still compare as equal. The default value of (1.0e-25) was taken from the GNC file mathconstants.h.
Limits	The LC app does not place a limit on this parameter.

Table 21 Configuration Parameter - Maximum Actionpoint Event Text String Size

Configuration Parameter	LC_MAX_ACTION_TEXT
CFS LC Default	32
Mission-Specific Default	Mission Defined
Purpose	Maximum actionpoint event text string size
Description	Maximum length of the event message string that can specified in an actionpoint definition (including NUL terminator)
Limits	LC appends the trailer text LC_AP_EVENT_TAIL_STR to this string when reporting actionpoint failures. The size of this string is LC_AP_EVENT_TAIL_LEN. The total value of LC_MAX_ACTION_TEXT + LC_AP_EVENT_TAIL_LEN should be less than CFE_EVS_MAX_MESSAGE_LENGTH to avoid event message truncation. Raising this value will also increase the size of the Actionpoint Definition Table (ADT).

Table 22 Configuration Parameter - Maximum Number of Actionpoints

Configuration Parameter	LC_MAX_ACTIONPOINTS

CFS LC Default	176
Mission-Specific Default	Mission Defined
Purpose	Maximum number of actionpoints
Description	Maximum number of actionpoints that can be defined in the Actionpoint Definition Table (ADT)
Limits	• This parameter cannot be larger than an unsigned 16 bit integer (65535). It must be a multiple of two to avoid indexing past the end of the array as LC indexes ahead to build the packed status bytes. • This parameter will dictate the size of the Actionpoint Definition Table: ADT Size = (Configuration Parameter Maximum Number of Actionpoints) * size of (Actionpoint Definition Table). • The total size of this table should not exceed the cFE size limit for a single buffered table set by the CFE_TBL_MAX_SNGL_TABLE_SIZE parameter.

Table 23 Configuration Parameter - Maximum Reverse Polish (RPN) Equation Size

Configuration Parameter	LC_MAX_RPN_EQU_SIZE
CFS LC Default	20
Mission-Specific Default	Mission Defined
Purpose	Maximum RPN equation size
Description	Maximum combined number of operators and operands that may exist in an actionpoint definition's RPN equation.
Limits	The LC app does not place a limit on this parameter. However, raising this value will increase the size of the Actionpoint Definition Table (ADT).

Table 24 Configuration Parameter - Maximum Valid ADT RTS ID

Configuration Parameter	LC_MAX_VALID_ADT_RTSID

CFS LC Default	256
Mission-Specific Default	Mission Defined
Purpose	Maximum valid ADT RTS ID.
Description	The maximum RTS ID that LC will allow during table validation in an Actionpoint Definition Table (ADT) entry.
Limits	This parameter cannot be larger than an unsigned 16 bit integer (65535).

Table 25 Configuration Parameter - Maximum Number of Watchpoints

Configuration Parameter	LC_MAX_WATCHPOINTS
CFS LC Default	176
Mission-Specific Default	Mission Defined
Purpose	Maximum number of watchpoints
Description	Maximum number of watchpoints that can be defined in the Watchpoint Definition Table (WDT).
Limits	This parameter cannot be larger than 65520 (0xFFF0) because higher values are reserved for use as RPN operators. It must be a multiple of four to avoid indexing past the end of the array as LC indexes ahead to build the packed status bytes. This parameter will dictate the size of the Watchpoint Definition Table: WDT Size = LC_MAX_WATCHPOINTS * sizeof (Watchpoint Definition Table) The total size of this table should not exceed the cFE size limit for a single buffered table set by the CFE_TBL_MAX_SNGL_TABLE_SIZE parameter.

Table 26 Configuration Parameter - Mission Specific Version Number for LC Application

Configuration Parameter	LC_MISSION_REV
CFS LC Default Value	1

Mission-Specific Default	Mission Defined
Purpose	Mission specific version number for LC application
Description	An application version number consists of four parts: • major version number, • minor version number, • revision number and • mission specific revision number. From an FSSE standpoint, the mission specific revision number is defined here and the other parts are defined in "lc_version.h".
Limits	Must be defined as a numeric value that is greater than or equal to zero.

Table 27 Configuration Parameter - Command Pipe Depth

Configuration Parameter	LC_PIPE_DEPTH
CFS LC Default Value	36
Mission-Specific Default	Mission Defined
Purpose	Command Pipe Depth
Description	Maximum number of messages that will be allowed in the LC command pipe at one time. Used during initialization in the call to CFE_SB_CreatePipe
Limits	This parameter cannot be larger than an unsigned 16 bit integer (65535).

Table 28 Configuration Parameter - LC State after Power-On Reset

Configuration Parameter	LC_STATE_POWER_ON_RESET
CFS LC Default Value	LC_STATE_DISABLED
Mission-Specific Default	Mission Defined
Purpose	LC state after power-on reset

Description	To what operating state LC should initialize after a power-on reset.
Limits	This parameter must be one of the following: LC_STATE_ACTIVE (1) LC_STATE_PASSIVE (2) LC_STATE_DISABLED (3) LC_STATE_FROM_CDS (4) [Available only if mission is using Critical Data Store (CDS)].

Table 29 Configuration Parameter - Save to CDS Compiler Switch/LC State When CDS Restored

Configuration Parameter	LC_STATE_WHEN_CDS_RESTORED
CFS LC Default Value	LC_STATE_FROM_CDS
Mission-Specific Default	Mission Defined
Purpose	1. Save data to CDS compiler switch. 2. LC state when CDS is restored.
Description	1. Save data to CDS compiler switch: Compile switch that tells LC that we should save data over a processor or application reset by using the Critical Data Store (CDS). This may be commented out or #undef by the mission in order to force LC to do a default (power-on) initialization sequence on all restarts (this is the default case). 2. LC state when CDS is restored: What operating state LC should initialize to after successfully restoring information from the CDS after a processor or application reset. This is only used when LC_SAVE_TO_CDS is set to TRUE, and provides a way to override any state LC may have been operating in prior to the reset occurring.
Limits	1. Save data to CDS compiler switch: N/A 2. LC state when CDS is restored: This parameter must be one of the following: LC_STATE_ACTIVE LC_STATE_PASSIVE LC_STATE_DISABLED LC_STATE_FROM_CDS

Table 30 Configuration Parameter - Watchpoint Definition Table (WDT) filename

Configuration Parameter	LC_WDT_FILENAME

CFS LC Default Value	"/boot/lc_def_wdt.tbl"
Mission-Specific Default	Mission Defined
Purpose	Watchpoint Definition Table (WDT) filename
Description	Default file from which to load the watchpoint definition table during a power-on reset sequence.
Limits	This string should be no longer than OS_MAX_PATH_LEN for the target platform in question.

A.4 LC Commands

The tables in this section show detailed descriptions of all the standard commands available to ground controllers for use with the LC application.

Table 31 Command 0 – Noop

CFS LC Command Code	0
Command Name	Noop
Description	Implements the Noop command that insures the LC task is alive; increments the Command Accepted Counter and sends an event message with application version information.
CFS LC Default Command Mnemonic(s)	$sc_$cpu_LC_NOOP
Mission Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_LC_CMDPC (or as defined by the mission) - command counter will increment. • The Event ID 3 informational event message will be generated when the command is received.
Error Conditions	This command may fail for the following reason(s): • Command packet length is not as expected.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_LC_CMDEC (or as defined by the mission); command error counter will increment. • Error specific event message Event ID 44

Criticality	None
See also:	Table 32 (Command 1 - Reset Counters) below.

Table 32 Command 1 - Reset Counters

CFS LC Command Code	1
Command Name	Reset Counters
Description	Resets the LC housekeeping counters.
CFS LC Default Command Mnemonic	$sc_$cpu_LC_ResetCtrs
Mission Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_LC_CMDPC (or as defined by the mission); command counter will be cleared. • $sc_$cpu_LC_CMDEC (or as defined by the mission); command error counter will be cleared. • $sc_$cpu_LC_APSampleCnt (or as defined by the mission); actionpoint sample counter will be cleared. • $sc_$cpu_LC_MonMsgCnt (or as defined by the mission); monitored message counter will be cleared. • $sc_$cpu_LC_RTSCnt (or as defined by the mission); RTS execution counter will be cleared. • $sc_$cpu_LC_PassRTSCnt (or as defined by the mission); passive RTS execution counter will be cleared. • The Event ID 4 debug event message will be generated when the command is executed.
Error Conditions	This command may fail for the following reason(s): Command packet length is not as expected.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_LC_CMDEC (or as defined by the mission); command error counter will increment. • Error specific event message Event ID 44
Criticality	None
See also:	Table 31 (Command 0 – Noop) above on page A-16.

Table 33 Command 2 – Set LC Application State

CFS LC Command Code	2
Command Name	Set LC Application State
Description	Sets the operational state of the LC application (Active, Passive, Disabled).
CFS LC Default Command Mnemonic	$sc_$cpu_LC_SetLCState
Mission Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_LC_CMDPC (or as defined by the mission); command counter will increment. • $sc_$cpu_LC_CurrentLCState (or as defined by the mission) - will be set to the new state. • The Event ID 49 informational event message will be generated when the command is executed.
Error Conditions	This command may fail for the following reason(s): • Command packet length is not as expected. • Invalid new state is specified in command message.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_LC_CMDEC (or as defined by the mission); command error counter will increment. • Error specific event message Event ID 44 • Error specific event message Event ID 50
Criticality	None

Table 34 Command 3 – Set AP State

CFS LC Command Code	3
Command Name	Set AP State
Description	Sets the state of one or all actionpoints (Active, Passive, Disabled).

CFS LC Default Command Mnemonic	$sc_$cpu_LC_SetAPState
Mission Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_LC_CMDPC (or as defined by the mission); command counter will increment. • The Event ID 51 informational event message will be generated when the command is executed.
Error Conditions	This command may fail for the following reason(s): • Command packet length is not as expected. • Invalid actionpoint state is specified in command message. • Actionpoint number specified in command message is out of range. • Actionpoint current state is either LC_ACTION_NOT_USED or LC_APSTATE_PERMOFF.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_LC_CMDEC (or as defined by the mission); command error counter will increment. • Error specific event message Event ID 44 • Error specific event message Event ID 52 • Error specific event message Event ID 54 • Error specific event message Event ID 53
CFS Criticality	None
Mission Specific Criticality	Mission Defined

Table 35 Command 4 – Set AP Permanently Off

CFS LC Command Code	4
Command Name	Set AP Permanently Off
Description	Sets the state of a single actionpoint to permanently off (requires table load to restore).
CFS LC Default Command Mnemonic	$sc_$cpu_LC_SetAPPermOff

Mission Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_LC_CMDPC (or as defined by the mission); command counter will increment. • The Event ID 55 informational event message will be generated when the command is executed.
Error Conditions	This command may fail for the following reason(s): • Command packet length is not as expected. • Actionpoint number specified in command message is out of range • Actionpoint current state is not LC_APSTATE_DISABLED.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_LC_CMDEC (or as defined by the mission); command error counter will increment. • Error specific event message Event ID 44 • Error specific event message Event ID 56 • Error specific event message Event ID 57
Criticality	None

Table 36 Command 5 – Reset AP Statistics

CFS LC Command Code	5
Command Name	Reset AP Statistics
Description	Reset statistics in the Actionpoint Results Table (ART) for one or all actionpoints.
CFS LC Default Command Mnemonic	$sc_$cpu_LC_ResetAPStats
Mission Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_LC_CMDPC (or as defined by the mission); command counter will increment. • The Event ID 58 informational event message will be generated when the command is executed.

Error Conditions	This command may fail for the following reason(s): • Command packet length is not as expected. • Actionpoint number specified in command message is out of range.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_LC_CMDEC (or as defined by the mission) - command error counter will increment • Error specific event message Event ID 44 • Error specific event message Event ID 59
Criticality	None
See also:	Table 37 (Command 6 – Reset WP Statistics) below, on page A-21.

Table 37 Command 6 – Reset WP Statistics

CFS LC Command Code	6
Command Name	Reset WP Statistics
Description	Reset statistics in the Watchpoint Results Table (WRT) for one or all watchpoints.
CFS LC Default Command Mnemonic	$sc_$cpu_LC_ResetWPStats
Mission Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_LC_CMDPC (or as defined by the mission); command counter will increment. • The Event ID 60 informational event message will be generated when the command is executed.
Error Conditions	This command may fail for the following reason(s): • Command packet length is not as expected. • Watchpoint number specified in command message is out of range.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_LC_CMDEC (or as defined by the mission) - command error counter will increment • Error specific event message Event ID 44 • Error specific event message Event ID 61

Criticality	None
See also:	Table 36 (Command 5 – Reset AP Statistics) above, on page A-20.

A.5 Event Messages

This section shows event messages. Event messages are not mission specific. Generally the Event ID numbers shown below should appear in telemetry on ground flight software systems (e.g., ASIST), no matter the mission.

A.5.1 Event Messages - CRITICAL

The tables in this section show **critical** event messages for the LC application.

Table 38 Event ID 1 (Critical)

Event ID Number:	1
Event Message:	'Task terminating, err = 0x%08X'
Type:	CRITICAL
Cause:	• This event message is issued when the CFS Limit Checker exits due to a fatal error condition.
	• The **`err`** field contains the return status from the cFE call that caused the task to terminate.

A.5.2 Event Messages - ERROR

The tables in this section show **error** event messages for the LC application.

Table 39 Event ID 5 (Error)

Event ID Number:	5
Event Message:	'Error Creating LC Pipe, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the CFS Limit Checker is unable to create its command pipe via the CFE_SB_CreatePipe API. • The **`RC`** field contains the return status from the CFE_SB_CreatePipe call that generated the error.

Table 40 Event ID 6 (Error)

Event ID Number:	6
Event Message:	'Error Subscribing to HK Request, MID=0x%04X, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the CFS Limit Checker is unable to subscribe to its Housekeeping Request message via the CFE_SB_Subscribe API. • The **`MID`** field contains the Message ID that LC was attempting to subscribe to. • The **`RC`** field contains the return status from the CFE_SB_Subscribe call that generated the error.

Table 41 Event ID 7 (Error)

Event ID Number:	7
Event Message:	'Error Subscribing to GND CMD, MID=0x%04X, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the CFS Limit Checker is unable to subscribe to its ground commands via the CFE_SB_Subscribe API. • The **`MID`** field contains the Message ID that LC was attempting to subscribe to. • The **`RC`** field contains the return status from the CFE_SB_Subscribe call that generated the error.

Table 42 Event ID 8 (Error)

Event ID Number:	8
Event Message:	'Error Subscribing to Sample CMD, MID=0x%04X, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the CFS Limit Checker is unable to subscribe to its actionpoint sample command via the CFE_SB_Subscribe API. • The **`MID`** field contains the Message ID that LC was attempting to subscribe to. • The **`RC`** field contains the return status from the CFE_SB_Subscribe call that generated the error.

Table 43 Event ID 11 (Error)

Event ID Number:	11
Event Message:	'Error registering WDT, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the watchpoint definition table (WDT) could not be registered.
	• The **RC** field is the return code from the CFE_TBL_Register function call that generated the error.

Table 44 Event ID 14 (Error)

Event ID Number:	14
Event Message:	'Error registering ADT, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the actionpoint definition table (ADT) could not be registered.
	• The **RC** field is the return code from the CFE_TBL_Register function call that generated the error.

Table 45 Event ID 15 (Error)

Event ID Number:	15
Event Message:	'Error registering WRT, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the watchpoint results table (WRT) could not be registered.
	• The **RC** field is the return code from the CFE_TBL_Register function call that generated the error.

Table 46 Event ID 16 (Error)

Event ID Number:	16
Event Message:	'Error registering ART, RC=0x%08X'
Type:	ERROR

Cause:	• This event message is issued when the actionpoint results table (ART) could not be registered.
	• The `RC` field is the return code from the CFE_TBL_Register function call that generated the error.

Table 47 Event ID 17 (Error)

Event ID Number:	17
Event Message:	'Error registering WRT CDS Area, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the CDS area for the watchpoint results table (WRT) data could not be registered.
	• The `RC field` is the return code from the CFE_ES_RegisterCDS function call that generated the error.

Table 48 Event ID 18 (Error)

Event ID Number:	18
Event Message:	'Error registering ART CDS Area, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the CDS area for the actionpont restuls table (ART) data could not be registered.
	• The `RC` field is the return code from the CFE_ES_RegisterCDS function call that generated the error.

Table 49 Event ID 19 (Error)

Event ID Number:	19
Event Message:	'Error registering application data CDS Area, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the CDS area for the LC application data could not be registered.
	• The `RC` field is the return code from the CFE_ES_RegisterCDS function call that generated the error.

Table 50 Event ID 31 (Error)

Event ID Number:	31
Event Message:	'Error subscribing watchpoint: MID=0x%04X, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when an error is encountered when subscribing to a watchpoint message ID.
	• The **`MID`** field is the message ID. • The **`RC`** field is the return code from the CFE_SB_Subscribe call that generated the error.

Table 51 Event ID 32 (Error)

Event ID Number:	32
Event Message:	'Error unsubscribing watchpoint: MID=0x%04X, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when an error is encountered unsubscribing to a watchpoint message ID.
	• The **`MID`** field is the message ID. • The **`RC`** field is the return code from the CFE_SB_Unsubscribe call that generated the error.

Table 52 Event ID 33 (Error)

Event ID Number:	33
Event Message:	'Error (RC=0x%08X) Loading WDT with '%s''
Type:	ERROR
Cause:	• This event message is issued when an error is encountered loading the watchpoint definition table (WDT) from the default file image.
	• The **`RC`** field is the return code from the CFE_TBL_Load call that generated the error. • The **`with`** field is the name of the load file.

Table 53 Event ID 34 (Error)

Event ID Number:	34
Event Message:	'Error (RC=0x%08X) Loading ADT with '%s''

Type:	ERROR
Cause:	• This event message is issued when an error is encountered loading the actionpoint definition table (ADT) from the default file image.
	• The **`RC`** field is the return code from the CFE_TBL_Load call that generated the error. • The **`with`** field is the name of the load file.

Table 54 Event ID 36 (Error)

Event ID Number:	36
Event Message:	'WRT data NOT saved to CDS on exit, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the watchpoint results table (WRT) data could not be saved to the CDS on application exit.
	• The **`RC`** field is the return code from the CFE_ES_CopyToCDS call that generated the error.

Table 55 Event ID 38 (Error)

Event ID Number:	38
Event Message:	'ART data NOT saved to CDS on exit, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the actionpoint results table (ART) data could not be saved to the CDS on application exit.
	• The **`RC`** field is the return code from the CFE_ES_CopyToCDS call that generated the error.

Table 56 Event ID 40 (Error)

Event ID Number:	40
Event Message:	'Application data NOT saved to CDS on exit, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the LC application data could not be saved to the CDS on application exit.
	• The **`RC`** field is the return code from the CFE_ES_CopyToCDS call that generated the error.

Table 57 Event ID 41 (Error)

Event ID Number:	41
Event Message:	'Invalid command code: ID = 0x%04X, CC = %d'
Type:	ERROR
Cause:	• This event message is issued when a software bus message is received with an invalid command code.
	• The **`ID`** field contains the message ID. • The **`CC`** field contains the command code that generated the error.

Table 58 Event ID 42 (Error)

Event ID Number:	42
Event Message:	'Invalid HK request msg length: ID = 0x%04X, CC = %d, Len = %d, Expected = %d'
Type:	ERROR
Cause:	• This event message is issued when a housekeeping request is received with a message length that does not match the expected value.
	• The **`ID`** field contains the message ID. • The **`CC`** field contains the command code. • The **`Len`** field is the actual length returned by the CFE_SB_GetTotalMsgLength call. • The **`Expected`** field is the expected length for the message.

Table 59 Event ID 43 (Error)

Event ID Number:	43
Event Message:	'Invalid AP sample msg length: ID = 0x%04X, CC = %d, Len = %d, Expected = %d'
Type:	ERROR
Cause:	• This event message is issued when a actionpoint sample request is received with a message length that does not match the expected value.
	• The **`ID`** field contains the message ID. • The **`CC`** field contains the command code. • The **`Len`** field is the actual length returned by the CFE_SB_GetTotalMsgLength call. • The **`Expected`** field is the expected length for the message.

Table 60 Event ID 44 (Error)

Event ID Number:	44
Event Message:	'Invalid msg length: ID = 0x%04X, CC = %d, Len = %d, Expected = %d'
Type:	ERROR
Cause:	• This event message is issued when a ground command message is received with a message length that does not match the expected value.
	• The **`ID`** field contains the message ID. • The **`CC`** field contains the command code. • The **`Len`** field is the actual length returned by the CFE_SB_GetTotalMsgLength call. • The **`Expected`** field is the expected length for a message with that command code.

Table 61 Event ID 45 (Error)

Event ID Number:	45
Event Message:	'Error getting WRT address, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the address cannot be obtained from table services for the watchpoint results table (WRT).
	• The **`RC`** field is the return code from the CFE_TBL_GetAddress function call that generated the error.

Table 62 Event ID 46 (Error)

Event ID Number:	46
Event Message:	'Error getting ART address, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the address cannot be obtained from table services for the actionpoint results table (ART).
	• The **`RC`** field is the return code from the CFE_TBL_GetAddress function call that generated the error.

Table 63 Event ID 47 (Error)

Event ID Number:	47
Event Message:	'Error getting WDT address, RC=0x%08X'

Type:	ERROR
Cause:	• This event message is issued when the address cannot be obtained from table services for the watchpoint definition table (WDT).
	• The **`RC`** field is the return code from the CFE_TBL_GetAddress function call that generated the error.

Table 64 Event ID 48 (Error)

Event ID Number:	48
Event Message:	'Error getting ADT address, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the address cannot be obtained from table services for the actionpoint definition table (ADT).
	• The **`RC`** field is the return code from the CFE_TBL_GetAddress function call that generated the error.

Table 65 Event ID 50 (Error)

Event ID Number:	50
Event Message:	'Set LC state error: invalid state = %d'
Type:	ERROR
Cause:	• This event message is issued when the LC_SET_LC_STATE_CC command has been received with an invalid state argument specified.
	• The **`invalid state`** field is the state specified in the command message that triggered the error.

Table 66 Event ID 52 (Error)

Event ID Number:	52
Event Message:	'Set AP state error: AP = %d, Invalid new state = %d'
Type:	ERROR
Cause:	• This event message is issued when the LC_SET_AP_STATE_CC command has been received with an invalid state argument specified.
	• The **`AP`** field is the specified actionpoint number. • The **`Invalid new state`** field is the state specified in the command message that triggered the error.

Table 67 Event ID 53 (Error)

Event ID Number:	53
Event Message:	'Set AP state error: AP = %d, Invalid current AP state = %d'
Type:	ERROR
Cause:	• This event message is issued when the LC_SET_AP_STATE_CC command has been received and the current actionpoint state is either LC_ACTION_NOT_USED or LC_APSTATE_PERMOFF (which can only be changed with a table load).
	• The **`AP`** field is the specified actionpoint number. • The **`Invalid current AP state`** field is the current state that was determined invalid.

Table 68 Event ID 54 (Error)

Event ID Number:	54
Event Message:	'Set AP state error: Invalid AP number = %d'
Type:	ERROR
Cause:	• This event message is issued when the LC_SET_AP_STATE_CC command has been received with an invalid actionpoint number specified.
	• The **`Invalid AP number`** field is the number specified in the command message that triggered the error.

Table 69 Event ID 56 (Error)

Event ID Number:	56
Event Message:	'Set AP perm off error: Invalid AP number = %d'
Type:	ERROR
Cause:	• This event message is issued when the LC_SET_AP_PERMOFF_CC command has been received with an invalid actionpoint number specified.
	• The **`Invalid AP number`** field is the number specified in the command message that triggered the error.

Table 70 Event ID 57 (Error)

Event ID Number:	57

Event Message:	'Set AP perm off error, AP NOT Disabled: AP = %d, Current state = %d'
Type:	ERROR
Cause:	• This event message is issued when the LC_SET_AP_PERMOFF_CC command has been received and the current actionpoint state is not LC_APSTATE_DISABLED.
	• The **`AP`** field is the specified actionpoint number. • The **`Current state`** field is the current state of the actionpoint.

Table 71 Event ID 59 (Error)

Event ID Number:	59
Event Message:	'Reset AP stats error: invalid AP number = %d'
Type:	ERROR
Cause:	• This event message is issued when the LC_RESET_AP_STATS_CC command has been received with an invalid actionpoint number specified.
	• The **`invalid AP number`** field is the number specified in the command message that triggered the error.

Table 72 Event ID 61 (Error)

Event ID Number:	61
Event Message:	'Reset WP stats error: invalid WP number = %d'
Type:	ERROR
Cause:	• This event message is issued when the LC_RESET_WP_STATS_CC command has been received with an invalid watchpoint number specified.
	• The **`invalid WP number`** field is the number specified in the command message that triggered the error.

Table 73 Event ID 63 (Error)

Event ID Number:	63
Event Message:	'WP has undefined data type: WP = %d, DataType = %d'
Type:	ERROR
Cause:	• This event message is issued whenever an undefined watchpoint data type identifier is detected.

	• The **`WP`** field is the watchpoint number. • The **`DataType`** field is the data type value that triggered the error.

Table 74 Event ID 64 (Error)

Event ID Number:	64
Event Message:	'WP offset error: MID = %d, WP = %d, Offset = %d, DataSize = %d, MsgLen = %d'
Type:	ERROR
Cause:	• This event message is issued when a watchpoint offset value extends past the end of the message as reported by the CFE_SB_GetTotalMsgLength function.
	• The **`MID`** field is the message ID. • The **`WP`** field is the watchpoint number. • The **`Offset`** field is the watchpoint offset. • The **`DataSize`** field is the size of the watchpoint data in bytes. • The **`MsgLen`** field is the reported message length from CFE_SB_GetTotalMsgLength.

Table 75 Event ID 65 (Error)

Event ID Number:	65
Event Message:	'WP has invalid operator ID: WP = %d, OperID = %d'
Type:	ERROR
Cause:	• This event message is issued whenever an undefined watchpoint operator identifier is detected.
	• The **`WP`** field is the watchpoint number. • The **`OperID`** field is the operator ID value that triggered the error.

Table 76 Event ID 66 (Error)

Event ID Number:	66
Event Message:	'WP data value is a float NAN: WP = %d, Value = 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when a watchpoint is defined as a float type, but the extracted value would equate to a floating point NAN (not-a-number) value.

	• The `WP` field is the watchpoint number. • The `Value` field is the watchpoint value that triggered the error displayed as a 32 bit hexadecimal number.

Table 77 Event ID 67 (Error)

Event ID Number:	67
Event Message:	'Sample AP error: invalid AP number = %d'
Type:	ERROR
Cause:	• This event message is issued when the LC_SAMPLE_AP_MID message has been received with an invalid actionpoint number specified.
	• The `invalid AP number` field is the number specified in the command message that triggered the error.

Table 78 Event ID 68 (Error)

Event ID Number:	68
Event Message:	'Sample AP error, invalid current AP state: AP = %d, State = %d'
Type:	ERROR
Cause:	• This event message is issued when the LC_SAMPLE_AP_MID message has been received and the current state for the specified actionpoint state is either LC_ACTION_NOT_USED or LC_APSTATE_PERMOFF.
	• The `AP` field is the actionpoint number. • The `state` field is the current state of the actionpoint.

Table 79 Event ID 74 (Error)

Event ID Number:	74
Event Message:	'AP has illegal RPN expression: AP = %d, LastOperand = %d, StackPtr = %d'
Type:	ERROR
Cause:	• This event message is issued when an illegal Reverse Polish Notation (RPN) expression is detected during an actionpoint evaluation.
	• The `AP` field is the actionpoint number. • The `LastOperand` field is the operand when the error occurred. • The `StackPtr` field is the value of the equation stack pointer when the error occurred.

Table 80 Event ID 76 (Error)

Event ID Number:	76
Event Message:	'WDT verify err: WP = %d, Err = %d, DType = %d, Oper = %d, MID = %d'
Type:	ERROR
Cause:	• This event message is issued on the first error when a table validation fails for a watchpoint definition table (WDT) load and the error is NOT a failed floating point check.
	• The **`WP`** field is the watchpoint number. • The other fields are from the watchpoint's definition table entry that failed validation.

Table 81 Event ID 77 (Error)

Event ID Number:	77
Event Message:	'WDT verify float err: WP = %d, Err = %d, ComparisonValue = 0x%08X'
Type:	ERROR
Cause:	• This event message is issued on the first error when a table validation fails for a watchpoint definition table (WDT) load and the error is a failed floating point check. • This error is caused when the data type for a watchpoint definition is floating point and the comparison value equates to a floating point NAN (not-a-number) or infinite value.
	• The **`WP`** field is the watchpoint number.
	• The **`Err`** field is an error identifier.
	• The **`ComparisonValue`** field contains the data that triggered the error displayed as a 32 bit hexadecimal number.

Table 82 Event ID 79 (Error)

Event ID Number:	79
Event Message:	'ADT verify err: AP = %d, Err = %d, State = %d, RTS = %d, FailCnt = %d, EvtType = %d'
Type:	ERROR
Cause:	• This event message is issued on the first error when a table validation fails for an actionpoint definition table (ADT) load and the error is NOT a failed RPN equation check.

	• The **AP** field is the actionpoint number. • The other fields are from that actionpoint's definition table entry that failed validation.

Table 83 Event ID 80 (Error)

Event ID Number:	80
Event Message:	'ADT verify RPN err: AP = %d, Index = %d, StackDepth = %d'
Type:	ERROR
Cause:	• This event message is issued on the first error when a table validation fails for a actionpoint definition table (ADT) load and the error is a failed RPN equation check.
	• The **AP** field is the watchpoint number. • The **Index** field is the index into the equation where the error occurred, • The **StackDepth** field contains the RPN stack index when the error occurred.

Table 84 Event ID 81 (Error)

Event ID Number:	81
Event Message:	'Error dumping WRT to buffer, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when an error is returned by table services when dumping the watchpoint results table (WRT).
	• The **RC** field is the return code from the CFE_TBL_DumpToBuffer function call that generated the error.

Table 85 Event ID 82 (Error)

Event ID Number:	82
Event Message:	'Error getting WRT status, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the current status cannot be obtained from table services for the watchpoint results table (WRT).
	• The **RC** field is the return code from the CFE_TBL_GetStatus function call that generated the error.

Table 86 Event ID 83 (Error)

Event ID Number:	83
Event Message:	'Error dumping ART to buffer, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when an error is returned by table services when dumping the actionpoint results table (ART).
	• The **RC** field is the return code from the CFE_TBL_DumpToBuffer function call that generated the error.

Table 87 Event ID 84 (Error)

Event ID Number:	84
Event Message:	'Error getting ART status, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the current status cannot be obtained from table services for the actionpoint results table (ART).
	• The **RC** field is the return code from the CFE_TBL_GetStatus function call that generated the error.

Table 88 Event ID 85 (Error)

Event ID Number:	85
Event Message:	'Unexpected LC_CustomFunction call: WP = %d'
Type:	ERROR
Cause:	• This event message is issued when the mission specific custom function /LC_CustomFunction is called with an unexpected watchpoint ID.
	• The **WP** field is the watchpoint number that generated the call.

Table 89 Event ID 87 (Error)

Event ID Number:	87
Event Message:	'Application data NOT saved to CDS on startup, RC=0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the LC application data could not be saved to the CDS on application startup.

	• The **RC** field is the return code from the CFE_ES_CopyToCDS call that generated the error.

A.5.3 Event Messages - INFORMATION

The tables in this section show **information** type event messages for the LC application.

Table 90 Event ID 2 (Informational)

Event ID Number:	2
Event Message:	'LC Initialized. Version %d.%d.%d.%d'
Type:	INFORMATION
Cause:	• This event message is issued when the CFS Limit Checker has completed initialization.
	• The first **d** field contains the Application's Major Version Number. • The second **d** field contains the Application's Minor Version Number. • The third **d** field contains the Application's Revision Number. • The fourth **d** field contains the Application's Mission Revision Number.

Table 91 Event ID 3 (Informational)

Event ID Number:	3
Event Message:	'No-op command: Version %d.%d.%d.%d'
Type:	INFORMATION
Cause:	• This event message is issued when a NOOP command has been received.
	• The first **d** field contains the Application's Major Version Number. • The second **d** field contains the Application's Minor Version Number. • The third **d** field contains the Application's Revision Number. • The fourth **d** field contains the Application's Mission Revision Number.

Table 92 Event ID 27 (Informational)

Event ID Number:	27
Event Message:	'WDT and ADT CDS restore NOT complete, loading defaults'
Type:	INFORMATION

Cause:	This event message is issued when either the watchpoint or actionpoint definition table data could not be recovered from the CDS, forcing a default initialization sequence.

Table 93 Event ID 28 (Informational)

Event ID Number:	28
Event Message:	'Stats CDS restore NOT complete, resetting stats and counters'
Type:	INFORMATION
Cause:	This event message is issued when all the application and results table data could not be recovered from the CDS, forcing a reset of the results tables and application data on restart.

Table 94 Event ID 29 (Informational)

Event ID Number:	29
Event Message:	'CDS data not saved on last exit, resetting stats and counters'
Type:	INFORMATION
Cause:	This event message is issued when the results tables and application data was not updated in the CDS on exit, forcing a reset of the results tables and application data on restart.

Table 95 Event ID 30 (Informational)

Event ID Number:	30
Event Message:	'CDS restore complete'
Type:	INFORMATION
Cause:	This event message is issued when all the table and application data was successfully recovered from the CDS.

Table 96 Event ID 35 (Informational)

Event ID Number:	35
Event Message:	'WRT data saved to CDS on exit'
Type:	INFORMATION

Cause:	This event message is issued when the watchpoint results table (WRT) data has been successfully saved to the CDS on application exit.

Table 97 Event ID 37 (Informational)

Event ID Number:	37
Event Message:	'ART data saved to CDS on exit'
Type:	INFORMATION
Cause:	This event message is issued when the actionpoint results table (ART) data has been successfully saved to the CDS on application exit.

Table 98 Event ID 39 (Informational)

Event ID Number:	39
Event Message:	'Application data saved to CDS on exit'
Type:	INFORMATION
Cause:	This event message is issued when the LC application data has been successfully saved to the CDS on application exit.

Table 99 Event ID 49 (Informational)

Event ID Number:	49
Event Message:	'Set LC state command: new state = %d'
Type:	INFORMATION
Cause:	• This event message is issued when the LC_SET_LC_STATE_CC command has been successfully executed.
	• The **`new state`** field is the state specified in the command message that the LC operating state has been set to.

Table 100 Event ID 51 (Informational)

Event ID Number:	51
Event Message:	'Set AP state command: AP = %d, New state = %d'
Type:	INFORMATION
Cause:	• This event message is issued when the LC_SET_AP_STATE_CC command has been successfully executed.

	• The **`AP`** field is the actionpoint number. • The **`New state`** field is the state specified in the command message that the actionpoint state has been set to.

Table 101 Event ID 55 (Informational)

Event ID Number:	55
Event Message:	'Set AP permanently off command: AP = %d'
Type:	INFORMATION
Cause:	• This event message is issued when the LC_SET_AP_PERMOFF_CC command has been successfully executed.
	• The **`AP`** field is the actionpoint number that has been set to permanently off.

Table 102 Event ID 58 (Informational)

Event ID Number:	58
Event Message:	'Reset AP stats command: AP = %d'
Type:	INFORMATION
Cause:	• This event message is issued when the LC_RESET_AP_STATS_CC command has been successfully executed.
	• The **`AP`** field is the actionpoint number whose stats have been cleared.

Table 103 Event ID 60 (Informational)

Event ID Number:	60
Event Message:	'Reset WP stats command: WP = %d'
Type:	INFORMATION
Cause:	• This event message is issued when the LC_RESET_WP_STATS_CC command has been successfully executed.
	• The **`WP`** field is the watchpoint number whose stats have been cleared.

Table 104 Event ID 62 (Informational)

Event ID Number:	62

Event Message:	'Msg with unreferenced message ID rcvd: ID = 0x%04X'
Type:	INFORMATION
Cause:	• This event message is issued when a software bus message has been received that is not a recognized LC message and has no defined watchpoints referencing its message ID.
	• The `ID` field is the message ID.

Table 105 Event ID 71 (Informational)

Event ID Number:	71
Event Message:	'AP state change from PASS to FAIL: AP = %d'
Type:	INFORMATION
Cause:	• This event message is issued when an actionpoint evaluation transitions from LC_ACTION_PASS to LC_ACTION_FAIL.
	• The `AP` field is the actionpoint number that transitioned.

Table 106 Event ID 72 (Informational)

Event ID Number:	72
Event Message:	'AP state change from FAIL to PASS: AP = %d'
Type:	INFORMATION
Cause:	• This event message is issued when an actionpoint evaluation transitions from LC_ACTION_FAIL to LC_ACTION_PASS.
	• The `AP` field is the actionpoint number that transitioned.

Table 107 Event ID 73 (Informational)

Event ID Number:	73
Event Message:	'AP evaluated to error: AP = %d, Result = %d'
Type:	INFORMATION
Cause:	• This event message is issued when an actionpoint evaluation results in LC_ACTION_ERROR
	• The `AP` field is the actionpoint number. • The `Result` field is the evaluation value that triggered the error

Table 108 Event ID 75 (Informational)

Event ID Number:	75
Event Message:	'WDT verify results: good = %d, bad = %d, unused = %d'
Type:	INFORMATION
Cause:	• This event message is issued when a table validation has been completed for a watchpoint definition table (WDT) load.
	• The `good` field is number of entries that passed. • The `bad` field is number of entries that failed. • The `unused` field is the number of entries that were not checked because they were marked unused.

Table 109 Event ID 78 (Informational)

Event ID Number:	78
Event Message:	'ADT verify results: good = %d, bad = %d, unused = %d'
Type:	INFORMATION
Cause:	• This event message is issued when a table validation has been completed for an actionpoint definition table (ADT) load.
	• The `good` field is number of entries that passed. • The `bad` field is number of entries that failed. • The `unused` field is the number of entries that were not checked because they were marked unused.

A.5.4 Event Messages - DEBUG

The tables in this section show **debug** event messages for the LC application.

Table 110 Event ID 4 (Debug)

Event ID Number:	4
Event Message:	'Reset counters command'
Type:	DEBUG
Cause:	This event message is issued when a 'reset counters' command has been received.

Table 111 Event ID 9 (Debug)

Event ID Number:	9
Event Message:	'WDT recovered from CDS'
Type:	DEBUG
Cause:	This event message is issued when the watchpoint definition table (WDT) has been successfully recovered from the CDS.

Table 112 Event ID 10 (Debug)

Event ID Number:	10
Event Message:	'WDT NOT recovered from CDS'
Type:	DEBUG
Cause:	This event message is issued when the watchpoint definition table (WDT) could not be recovered from the CDS.

Table 113 Event ID 12 (Debug)

Event ID Number:	12
Event Message:	'ADT recovered from CDS'
Type:	DEBUG
Cause:	This event message is issued when the actionpoint definition table (ADT) has been successfully recovered from the CDS.

Table 114 Event ID 13 (Debug)

Event ID Number:	13
Event Message:	'ADT NOT recovered from CDS'
Type:	DEBUG
Cause:	This event message is issued when the actionpoint definition table (ADT) could not be recovered from the CDS.

Table 115 Event ID 20 (Debug)

Event ID Number:	20
Event Message:	'WRT data recovered from CDS'
Type:	DEBUG
Cause:	This event message is issued when the watchpoint results table (WRT) data has been successfully recovered from the CDS.

Table 116 Event ID 21 (Debug)

Event ID Number:	21
Event Message:	'WRT data NOT recovered from CDS'
Type:	DEBUG
Cause:	This event message is issued when the watchpoint results table (WRT) data could not be recovered from the CDS.

Table 117 Event ID 22 (Debug)

Event ID Number:	22
Event Message:	'ART data recovered from CDS'
Type:	DEBUG
Cause:	This event message is issued when the actionpoint results table (ART) data has been successfully recovered from the CDS.

Table 118 Event ID 23 (Debug)

Event ID Number:	23
Event Message:	'ART data NOT recovered from CDS'
Type:	DEBUG
Cause:	This event message is issued when the actionpoint results table (WRT) data could not be recovered from the CDS.

Table 119 Event ID 24 (Debug)

Event ID Number:	24

Event Message:	'Application data recovered from CDS'
Type:	DEBUG
Cause:	This event message is issued when the LC application data has been successfully recovered from the CDS.

Table 120 Event ID 25 (Debug)

Event ID Number:	25
Event Message:	'Application data NOT recovered from CDS'
Type:	DEBUG
Cause:	This event message is issued when the LC application data could not be recovered from the CDS.

Table 121 Event ID 26 (Debug)

Event ID Number:	26
Event Message:	'LC State recovered from CDS overridden, state = %d'
Type:	DEBUG
Cause:	• This event message is issued when the LC application state recovered from the CDS is overwritten with the LC_STATE_WHEN_CDS_RESTORED configuration parameter setting.
	• The **`state`** field is the operating state that the LC task will be set to.

Table 122 Event ID 69 (Debug)

Event ID Number:	69
Event Message:	'AP failed while LC App passive: AP = %d, FailCount = %d, RTS = %d'
Type:	DEBUG
Cause:	• This event message is issued when an actionpoint fails evaluation while the LC task operating state is LC_STATE_PASSIVE.
	• The **`AP`** field is the actionpoint number. • The **`FailCount`** field is how many times this actionpoint has failed.. • The **`RTS`** field is the RTS that was not initiated because LC was passive.

Table 123 Event ID 70 (Debug)

Event ID Number:	70
Event Message:	'AP failed while passive: AP = %d, FailCount = %d, RTS = %d'
Type:	DEBUG
Cause:	• This event message is issued when an actionpoint fails evaluation while the actionpoint state is LC_APSTATE_PASSIVE.
	• The **`AP`** field is the actionpoint number.
	• The **`FailCount`** field is how many times this actionpoint has failed.
	• The **`RTS`** field is the RTS that was not initiated because the actionpoint was passive.

Table 124 Event ID 86 (Debug)

Event ID Number:	86
Event Message:	'Application data saved to CDS on startup'
Type:	DEBUG
Cause:	• This event message is issued when the LC application data has been successfully saved to the CDS on application startup.

A.5.5 Event Messages – USER DEFINED

The tables in this section show **user defined** event messages for the LC application.

Table 125 Event ID 1000 (User Defined)

Event ID Number:	1000
Type:	User defined in Actionpoint Definition Table

Cause:	• The actionpoint base event ID is designed to avoid conflicts between the event ID's defined above for use by the LC application and the user defined event ID's in the actionpoint table. • These events are generated when the evaluation of an actionpoint results in sending a command to the stored command (SC) processor to start a real time command sequence (RTS). The event text is user defined and specific to the particular actionpoint. • Note that user defined event IDs can be easily recognized if the base number is easily recognizable. For example, using the value 1000 for the base event ID and using the actionpoint table index as the offset portion creates an obvious correlation. Thus, if an LC event ID is 1025 then it is immediately apparent that the event is the user defined event for actionpoint table index 25.

This page intentionally blank.

Appendix B Mission-Specific Reference

Mission specific LC-related documentation is located at (mission defined).

B.1 Watchpoint Definition Tables

Documentation of Watchpoint Definition Tables is located at (mission defined).

B.2 Actionpoint Definition Tables

Documentation of Actionpoint Definition Tables is located at (mission defined).

B.3 LC Configuration Table

Documentation of LC Configuration Table is located at (mission defined).

This page intentionally blank.

Appendix C Document Notes

C.1 Mission-Specific Conventions

- *This document presents selected information that should be removed when tailoring this document for a mission in this italic dark orange Times Roman font.*
- *Command and Telemetry mnemonics are mission-specific. This document as delivered has "suggested" names that may or may not be used by the mission when the MOT creates the ground system database. In particular, the suggested names start with $sc_$cpu_LC which indicate a global setting for spacecraft, processor selection, and the LC subsystem. This has meaning if the mission has multiple spacecraft, each with a copy of cFE/CFS apps being executed, and/or multiple processors per spacecraft, each with a copy of cFE/CFS apps being executed. Most missions have neither and they do not prepend a $sc_$cpu_ selection to the front of the command name. However it is common for missions to differentiate the spacecraft subsystem commands from instrument commands by prepending a couple of characters (e.g. pw for power system) to all the command and telemetry names for that subsystem.*
- *The nomenclature of command and telemetry mnemonics is highly mission-specific, so this document does not attempt to include the actual command and telemetry database names in advance. For example, for MMS the telemetry mnemonics are defined in a Record Definition Language (RDL) file for ASIST, and they will not exactly match the mnemonics in this Guide.*

C.2 Updating This Document

This section is for anyone updating this Guide in the future.

When tailoring this document for a particular mission, remove text appearing in this italic dark orange Times Roman font.

- *Review figures to be sure there is no conflict with mission configurations. Edit figures with MS Visio if necessary.*
- *Add Mission Defined values in Appendix A.*
- *Add the location of mission-specific documentation in Appendix B.*
- *Regenerate the table of contents (TOC); add Appendix page prefixes to TOC as needed.*

This Guide is formatted using Microsoft Word styles. When adding new sections to the Guide, assign paragraphs to the styles shown in the table below. Center tables and figures horizontally on the page. Use 15% grayscale in new table headings.

Table of Contents, Figures, and Tables can be updated automatically, but the letter prefix for Appendix pages must be added manually after update. To update all figure and table references in the document, when using the PC version of Word, select all, then choose F9.

Table 126 Internal Document Styles

Type	Style to be Used	Justification
Chapter titles, subtitles, and subsections.	Heading 1 through 6	Left
First level bullets	"List Bullet 1" style.	Left
Second level bullets	"List Bullet 2" style	Left
Numbered lists	"Style List Enum 0"	Left
Names of code modules	Code	N/A
All text not otherwise tagged	"Body Text"	Full justification
Tables & Figures		
First row of tables	Table Header	Center
All other cells of tables	Table Cells	Left
Figure captions	Caption Figure	Center
Table captions	Caption Table	Center

C.3 Providing Feedback about this Document

For CM reasons, if you find any item in this Guide that is in error, or want to be informed of any updates, please email feedback to the cFE/CFS PDL for validation and routing. As of the date of publication, the cFE/CFS PDL is Susie Strege (susanne.L.strege@nasa.gov).

Besides corrections of any errors, the CFS team is interested in your ideas on improvements that help understanding, such as, perhaps, improvements to flow charts to show better how LC juggles incoming commands with other program responsibilities, for example, or any other area.