Event Limiter

Component Design Document

1 Description

The Event Limiter is used to limit the number of events in the system for prevention of flooding the system with a single event. The component takes in a range of event IDs which sets up a 1-bit enable/disable state and 3-bit counter for each ID, for a total of 4-bits. Any IDs out of that range will not be limited. Each event counter monitors how many events have been sent through the component, before it limits from a given persistence value that can be set from 1 to 7. Each tick of the system will decrement each of the event counters by 1. At this point, there will be an event issued if any events are at the event limit and the number of events that were dropped. An event listed at the limit does not necessarily mean that any events of that ID were dropped. The component itself can be disabled to allow all events through with no limiting, but will maintain the state of each event for when it is re-enabled. The user also has the ability to dump all the states of the ID range in a packet. Each ID also has the ability to be disabled or enabled, where disabled will not limit events of that ID.

2 Requirements

These are the requirements for the Event Limiter component.

1. The Event Limiter component shall accept all events from the system and pass them on if not limited.
2. The Event Limiter component shall track the number of each particular event and limit if more than a set persistence, between 1 and 7, is seen in 1 tick.
3. The Event Limiter component shall allow one event through each tick if the event is at the limit and still coming through the system.
4. The Event Limiter component shall not limit event messages if the event is not set to enabled.
5. The Event Limiter component shall have the ability to enable or disable event limiting by command.
6. The Event Limiter component shall not limit and simply pass any events not initialized in the event ID range.
7. The Event Limiter component shall send a packet of all event states when issued by command.

3 Design

3.1 At a Glance

Below is a list of useful parameters and statistics that give a quick look into the makeup of the component.

- **Execution** - *passive*
- **Number of Connectors** - 9

- **Number of Invokee Connectors** - 3
- **Number of Invoker Connectors** - 6
- **Number of Generic Connectors** - *None*
- **Number of Generic Types** - *None*
- **Number of Unconstrained Arrayed Connectors** - *None*
- **Number of Commands** - 8
- **Number of Parameters** - *None*
- **Number of Events** - 14
- **Number of Faults** - *None*
- **Number of Data Products** - 3
- **Number of Data Dependencies** - *None*
- **Number of Packets** - 1

3.2 Diagram

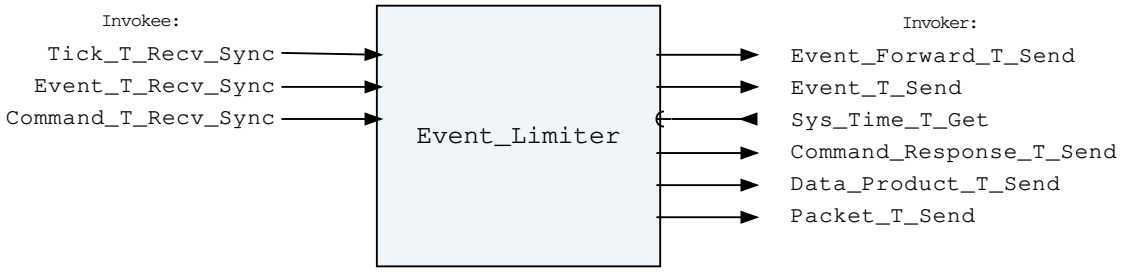

Figure 1: Event Limiter component diagram.

3.3 Connectors

Below are tables listing the component's connectors.

3.3.1 Invokee Connectors

The following is a list of the component's *invokee* connectors:

Table 1: Event Limiter Invokee Connectors

Name	Kind	Type	Return_Type	Count
Tick_T_Recv_Sync	recv_sync	Tick.T	-	1
Event_T_Recv_Sync	recv_sync	Event.T	-	1
Command_T_Recv_Sync	recv_sync	Command.T	-	1

Connector Descriptions:

- **Tick_T_Recv_Sync** - This is the base tick for the component. Upon reception the component will decrement the count of each ID unless it is already 0. Every tick, an event of what has been filtered will be sent.
- **Event_T_Recv_Sync** - Events are received synchronously on this connector and checked for the number of events of that ID.

- **Command_T_Recv_Sync** - This is the command receive connector.

3.3.2 Invoker Connectors

The following is a list of the component's *invoker* connectors:

Table 2: Event Limiter Invoker Connectors

Name	Kind	Type	Return_Type	Count
Event_Forward_T_Send	send	Event.T	-	1
Event_T_Send	send	Event.T	-	1
Sys_Time_T_Get	get	-	Sys_Time.T	1
Command_Response_T_Send	send	Command_Response.T	-	1
Data_Product_T_Send	send	Data_Product.T	-	1
Packet_T_Send	send	Packet.T	-	1

Connector Descriptions:

- **Event_Forward_T_Send** - The Event connector to forward on events when the limiting is disabled, unknown events come in, or if the event count is less than the persistence and is enabled for limiting.
- **Event_T_Send** - The Event connector to send the events specific to the component.
- **Sys_Time_T_Get** - The system time is retrieved via this connector.
- **Command_Response_T_Send** - This connector is used to register and respond to the component's commands.
- **Data_Product_T_Send** - The connector for data products
- **Packet_T_Send** - Packet for sending a packet for all the event states.

3.4 Interrupts

This component contains no interrupts.

3.5 Initialization

Below are details on how the component should be initialized in an assembly.

3.5.1 Component Instantiation

This component contains no instantiation parameters in its discriminant.

3.5.2 Component Base Initialization

This component contains no base class initialization, meaning there is no `init_Base` subprogram for this component.

3.5.3 Component Set ID Bases

This component contains commands, events, packets, faults, or data products that require a base identifier to be set at initialization. The `set_Id_Bases` procedure must be called with the following parameters:

Table 3: Event Limiter Set Id Bases Parameters

Name	Type
Command_Id_Base	Command_Types.Command_Id_Base
Data_Product_Id_Base	Data_Product_Types.Data_Product_Id_Base
Event_Id_Base	Event_Types.Event_Id_Base
Packet_Id_Base	Packet_Types.Packet_Id_Base

Parameter Descriptions:

- **Command_Id_Base** - The value at which the component's command identifiers begin.
- **Data_Product_Id_Base** - The value at which the component's data product identifiers begin.
- **Event_Id_Base** - The value at which the component's event identifiers begin.
- **Packet_Id_Base** - The value at which the component's unresolved packet identifiers begin.

3.5.4 Component Map Data Dependencies

This component contains no data dependencies.

3.5.5 Component Implementation Initialization

The calling of this implementation class initialization procedure is mandatory. The component achieves implementation class initialization using the `init` subprogram. The `init` subprogram requires the following parameters:

Table 4: Event Limiter Implementation Initialization Parameters

Name	Type	Default Value
Event_Id_Start	Event_Types.Event_Id	*None provided*
Event_Id_Stop	Event_Types.Event_Id	*None provided*
Event_Disable_List	Two_Counter_Entry. Event_Id_List	[1..0=>0]
Event_Limit_Persistence	Two_Counter_Entry. Persistence_Type	*None provided*

Parameter Descriptions:

- **Event_Id_Start** - The event ID that begins the range of ids that the component will include for potential limiting of events.
- **Event_Id_Stop** - The event ID that ends the range of ids that the component will include for potential limiting of events.
- **Event_Disable_List** - A list of event IDs that are disabled by default
- **Event_Limit_Persistence** - The initial persistence of the number of events to allow before limiting them between ticks (1 to 7)

3.6 Commands

These are the commands for the event limiter component.

Table 5: Event Limiter Commands

Local ID	Command Name	Argument Type
0	Enable_Event_Limit	Event_Single_State_Cmd_Type.T
1	Disable_Event_Limit	Event_Single_State_Cmd_Type.T
2	Enable_Event_Limit_Range	Event_Limiter_Id_Range.T
3	Disable_Event_Limit_Range	Event_Limiter_Id_Range.T
4	Enable_Event_Limiting	-
5	Disable_Event_Limiting	-
6	Set_Event_Limit_Persistence	Event_Limiter_Persistence_Type.T
7	Dump_Event_States	-

Command Descriptions:

- **Enable_Event_Limit** - Enable the event limiter for a specific event ID.
- **Disable_Event_Limit** - Disable the event limiter for a specific event ID.
- **Enable_Event_Limit_Range** - Enable the event limiter for a specific range of event IDs.
- **Disable_Event_Limit_Range** - Disable the event limiter for a specific range of event IDs.
- **Enable_Event_Limiting** - Enable the component to limit events that have been set to be limited.
- **Disable_Event_Limiting** - Disable the component so that all events will not be limited. The event states and counts will be maintained for when re-enabled.
- **Set_Event_Limit_Persistence** - Change the persistence of the event limiter for all events that are limited. Value must be between 1 and 7.
- **Dump_Event_States** - Dump a packet for the state of all events on if they are limited or not.

3.7 Parameters

The Event Limiter component has no parameters.

3.8 Events

Below is a list of the events for the Event Limiter component.

Table 6: Event Limiter Events

Local ID	Event Name	Parameter Type
0	Invalid_Command_Received	Invalid_Command_Info.T
1	Events_Limited_Since_Last_Tick	Event_Limiter_Num_Events_Type.T
2	Event_Limit_Enabled	Event_Single_State_Cmd_Type.T
3	Event_Limit_Disabled	Event_Single_State_Cmd_Type.T
4	Event_Limit_Range_Enabled	Event_Limiter_Id_Range.T
5	Event_Limit_Range_Disabled	Event_Limiter_Id_Range.T
6	Event_Limiting_Enabled	-
7	Event_Limiting_Disabled	-
8	Event_Limit_Enable_Invalid_Id	Event_Single_State_Cmd_Type.T

9	Event_Limit_Disable_Invalid_Id	Event_Single_State_ Cmd_Type.T
10	Event_Limit_Range_Enabled_Invalid_Id	Event_Limiter_Id_ Range.T
11	Event_Limit_Range_Disabled_Invalid_Id	Event_Limiter_Id_ Range.T
12	Set_New_Persistence	Event_Limiter_ Persistence_Type.T
13	Dump_Event_States_Received	-

Event Descriptions:

- **Invalid_Command_Received** - A command was received with invalid parameters.
- **Events_Limited_Since_Last_Tick** - An event that indicates how many events have been limited as well as up to the first 10 ids of those events. The event ids listed may not have been dropped, but are at listed since they are at the current max limit.
- **Event_Limit_Enabled** - This event indicates that the state of an event was set to enabled for the limiter.
- **Event_Limit_Disabled** - This event indicates that the state of an event was set to disabled for the limiter.
- **Event_Limit_Range_Enabled** - This event indicates that the state of a range of events were set to enabled for the limiter.
- **Event_Limit_Range_Disabled** - This event indicates that the state of a range of events were set to disabled for the limiter.
- **Event_Limiting_Enabled** - This event indicates that the state of all events were set to enabled for the limiter.
- **Event_Limiting_Disabled** - This event indicates that the state of all events were set to disabled for the limiter.
- **Event_Limit_Enable_Invalid_Id** - This event indicates that the command to change the event state to enabled failed since the event ID was out of range.
- **Event_Limit_Disable_Invalid_Id** - This event indicates that the command to change the event state to disabled failed since the event ID was out of range.
- **Event_Limit_Range_Enabled_Invalid_Id** - This event indicates that changing the state for the range to enabled, failed due to an invalid id.
- **Event_Limit_Range_Disabled_Invalid_Id** - This event indicates that changing the state for the range to disabled, failed due to an invalid id.
- **Set_New_Persistence** - Indicates that the persistence of the number of events until we limit was changed to a new value between 1 and 7.
- **Dump_Event_States_Received** - Event that indicates the process of building the packet that stores the event states has started and will send the packet once we go through a decrement cycle.

3.9 Data Products

Data products for the event limiter component.

Table 7: Event Limiter Data Products

Local ID	Data Product Name	Type
0x0000 (0)	Limited_Events_Since_Tick	Packed_U16.T
0x0001 (1)	Total_Events_Limited	Packed_U32.T
0x0002 (2)	Component_Limiting_Enabled_Status	Event_Enable_State_ Type.T

Data Product Descriptions:

- **Limited_Events_Since_Tick** - The number of events that were limited since the last tick.
- **Total_Events_Limited** - The total number of events that have been limited for the components lifetime.
- **Component_Limiting_Enabled_Status** - The current state of the component master switch.

3.10 Packets

Packet to dump the event state of all events. Each event state takes one bit.

Table 8: Event Limiter Packets

Local ID	Packet Name	Type
0x0000 (0)	Event_Limiter_State_Packet	*Undefined*

Packet Descriptions:

- **Event_Limiter_State_Packet** - The packet used to dump all the state information for which events are limited and which are not. Each event takes a bit and any extra bits beyond the event range will show as disabled.

4 Unit Tests

The following section describes the unit test suites written to test the component.

4.1 *Event_ Limiter_ Tests* Test Suite

This is a unit test suite for the Event Limiter component

Test Descriptions:

- **Test_Received_Event** - This unit test is to test incrementing counters for the range events as they come in with both enabled and disabled states.
- **Test_Decrement_Event_Count** - This unit test is to test the decrement of all the event counters in the tick and the construction of the event if events were limited.
- **Test_Issue_State_Packet** - This unit test sends the issue command and test that the appropriate values are received in that packet.
- **Test_Command_Single_State_Change** - This unit test is used to test enabling and disabling a single event and testing if the packet is sent if issued by the command.
- **Test_Command_Range_State_Change** - This unit test is used to test enabling and disabling a range of events and testing if the packet is sent if issued by the command.

- **Test_Command_Component_State_Change** - This unit test is used to test enabling and disabling all events and testing if the packet is sent if issued by the command.
- **Test_Persistence_Change** - This unit test exercises updating the persistence and ensures it was set correctly.
- **Test_Invalid_Command** - This unit test exercises that an invalid command throws the appropriate event.

5 Appendix

5.1 Preamble

This component contains no preamble code.

5.2 Packed Types

The following section outlines any complex data types used in the component in alphabetical order. This includes packed records and packed arrays that might be used as connector types, command arguments, event parameters, etc..

Command.T:

Generic command packet for holding arbitrary commands

Table 9: Command Packed Record : 2080 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Command_ Header.T	-	40	0	39	-
Arg_Buffer	Command_ Types. Command_Arg_ Buffer_Type	-	2040	40	2079	Header.Arg_ Buffer_Length

Field Descriptions:
- **Header** - The command header
- **Arg_Buffer** - A buffer that contains the command arguments

Command_Header.T:

Generic command header for holding arbitrary commands

Table 10: Command_Header Packed Record : 40 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Source_Id	Command_Types. Command_Source_Id	0 to 65535	16	0	15
Id	Command_Types. Command_Id	0 to 65535	16	16	31
Arg_Buffer_Length	Command_Types. Command_Arg_Buffer_ Length_Type	0 to 255	8	32	39

Field Descriptions:

- **`Source_Id`** - The source ID. An ID assigned to a command sending component.
- **`Id`** - The command identifier
- **`Arg_Buffer_Length`** - The number of bytes used in the command argument buffer

Command_Response.T:

Record for holding command response data.

Table 11: Command_Response Packed Record : 56 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Source_Id	Command_ Types.Command_ Source_Id	0 to 65535	16	0	15
Registration_ Id	Command_ Types.Command_ Registration_ Id	0 to 65535	16	16	31
Command_Id	Command_Types. Command_Id	0 to 65535	16	32	47
Status	Command_Enums. Command_ Response_ Status.E	0 => Success 1 => Failure 2 => Id_Error 3 => Validation_Error 4 => Length_Error 5 => Dropped 6 => Register 7 => Register_Source	8	48	55

Field Descriptions:

- **`Source_Id`** - The source ID. An ID assigned to a command sending component.
- **`Registration_Id`** - The registration ID. An ID assigned to each registered component at initialization.
- **`Command_Id`** - The command ID for the command response.
- **`Status`** - The command execution status.

Data_Product.T:

Generic data product packet for holding arbitrary data types

Table 12: Data_Product Packed Record : 344 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Data_Product_ Header.T	-	88	0	87	-

Buffer	Data_Product_ Types.Data_ Product_ Buffer_Type	-	256	88	343	Header.Buffer_ Length

Field Descriptions:

- **Header** - The data product header
- **Buffer** - A buffer that contains the data product type

Data_Product_Header.T:

Generic data_product packet for holding arbitrary data_product types

Table 13: Data_Product_Header Packed Record : 88 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Data_Product_Types. Data_Product_Id	0 to 65535	16	64	79
Buffer_Length	Data_Product_ Types.Data_Product_ Buffer_Length_Type	0 to 32	8	80	87

Field Descriptions:

- **Time** - The timestamp for the data product item.
- **Id** - The data product identifier
- **Buffer_Length** - The number of bytes used in the data product buffer

Event.T:

Generic event packet for holding arbitrary events

Table 14: Event Packed Record : 344 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Event_Header.T	-	88	0	87	-
Param_Buffer	Event_Types. Parameter_ Buffer_Type	-	256	88	343	Header.Param_ Buffer_Length

Field Descriptions:

- **Header** - The event header
- **Param_Buffer** - A buffer that contains the event parameters

Event_Enable_State_Type.T:

This record contains the definition for a two event ID type for ranges in the event limiter commands as well as an issue packet type for issuing packets

Table 15: Event_Enable_State_Type Packed Record : 8 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Component_Enable_State	Two_Counter_Entry_Enums.Event_State_Type.E	0 => Disabled 1 => Enabled	8	0	7

Field Descriptions:

- **Component_Enable_State** - Flag to indicate if the component is enabled or disabled at a overriding level

Event_Header.T:

Generic event packet for holding arbitrary events

Table 16: Event_Header Packed Record : 88 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Event_Types.Event_Id	0 to 65535	16	64	79
Param_Buffer_Length	Event_Types.Parameter_Buffer_Length_Type	0 to 32	8	80	87

Field Descriptions:

- **Time** - The timestamp for the event.
- **Id** - The event identifier
- **Param_Buffer_Length** - The number of bytes used in the param buffer

Event_Id.T:

A packed record which holds an event identifier.

Table 17: Event_Id Packed Record : 16 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Id	Event_Types.Event_Id	0 to 65535	16	0	15

Field Descriptions:

- **Id** - The event identifier

Event_Id_Array.T:

Packed array of events so that it can be used with the event for which ids were limited

Table 18: Event_Id_Array Packed Array : 160 bits

Type	Range	Element Size (Bits)	Length	Total Size (Bits)

Event_Types.Event_Id	0 to 65535	16	10	160

Event_Limiter_Id_Range.T:

This record contains the definition for a two event ID type for ranges in the event limiter commands as well as an issue packet type for issuing packets

Table 19: Event_Limiter_Id_Range Packed Record : 40 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Start_Event_Id	Event_Id.T	-	16	0	15
Stop_Event_Id	Event_Id.T	-	16	16	31
Issue_State_Packet	Event_Limiter_ Enums.Issue_ Packet_Type.E	0 => No_Issue 1 => Issue	8	32	39

Field Descriptions:

- **Start_Event_Id** - The starting event ID to begin the range
- **Stop_Event_Id** - The last event ID to end the range
- **Issue_State_Packet** - Flag to indicate if we dump the states after the change is complete

Event_Limiter_Num_Events_Type.T:

This record contains the definition for the format of the event for this component that contains up to 10 event IDs of those that were limited.

Table 20: Event_Limiter_Num_Events_Type Packed Record : 184 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Num_Events_Limited	Interfaces. Unsigned_16	0 to 65535	16	0	15
Num_Event_Ids	Interfaces. Unsigned_8	0 to 255	8	16	23
Event_Id_Limited_ Array	Event_Id_Array.T	-	160	24	183

Field Descriptions:

- **Num_Events_Limited** - The number of events limited since last issuing the event.
- **Num_Event_Ids** - The number of event IDs contained in the event that indicate that event was limited.
- **Event_Id_Limited_Array** - The Array that list the event IDs that were limited since the last event message.

Event_Limiter_Persistence_Type.T:

This record contains the definition for a packed persistence type

Table 21: Event_Limiter_Persistence_Type Packed Record : 8 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Persistence	Two_Counter_Entry. Persistence_Type	1 to 7	8	0	7

Field Descriptions:

- **Persistence** - The persistence that is used when limiting events.

Event_Single_State_Cmd_Type.T:

This record contains the definition for a two event ID type for ranges in the event limiter commands as well as an issue packet type for issuing packets

Table 22: Event_Single_State_Cmd_Type Packed Record : 24 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Event_To_Update	Event_Id.T	-	16	0	15
Issue_State_Packet	Event_Limiter_ Enums.Issue_ Packet_Type.E	0 => No_Issue 1 => Issue	8	16	23

Field Descriptions:

- **Event_To_Update** - The starting event ID to begin the range
- **Issue_State_Packet** - Flag to indicate if we dump the states after the change is complete

Invalid_Command_Info.T:

Record for holding information about an invalid command

Table 23: Invalid_Command_Info Packed Record : 112 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Id	Command_Types. Command_Id	0 to 65535	16	0	15
Errant_Field_ Number	Interfaces. Unsigned_32	0 to 4294967295	32	16	47
Errant_Field	Basic_Types.Poly_ Type	-	64	48	111

Field Descriptions:

- **Id** - The command Id received.
- **Errant_Field_Number** - The field that was invalid. 1 is the first field, 0 means unknown field, 2**32 means that the length field of the command was invalid.
- **Errant_Field** - A polymorphic type containing the bad field data, or length when Errant_Field_Number is 2**32.

Packed_U16.T:

Single component record for holding packed unsigned 16-bit value.

Table 24: Packed_U16 Packed Record : 16 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Value	Interfaces. Unsigned_16	0 to 65535	16	0	15

Field Descriptions:

- **Value** - The 16-bit unsigned integer.

Packed_U32.T:

Single component record for holding packed unsigned 32-bit value.

Table 25: Packed_U32 Packed Record : 32 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Value	Interfaces. Unsigned_32	0 to 4294967295	32	0	31

Field Descriptions:

- **Value** - The 32-bit unsigned integer.

Packet.T:

Generic packet for holding arbitrary data

Table 26: Packet Packed Record : 10080 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Packet_ Header.T	-	112	0	111	-
Buffer	Packet_ Types.Packet_ Buffer_Type	-	9968	112	10079	Header. Buffer_Length

Field Descriptions:

- **Header** - The packet header
- **Buffer** - A buffer that contains the packet data

Packet_Header.T:

Generic packet header for holding arbitrary data

Table 27: Packet_Header Packed Record : 112 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Packet_Types. Packet_Id	0 to 65535	16	64	79

Sequence_Count	Packet_Types. Sequence_Count_Mod_ Type	0 to 16383	16	80	95
Buffer_Length	Packet_Types. Packet_Buffer_ Length_Type	0 to 1246	16	96	111

Field Descriptions:

- **Time** - The timestamp for the packet item.
- **Id** - The packet identifier
- **Sequence_Count** - Packet Sequence Count
- **Buffer_Length** - The number of bytes used in the packet buffer

Sys_Time.T:

A record which holds a time stamp using GPS format including seconds and subseconds since epoch (1-5-1980 to 1-6-1980 midnight).

Table 28: Sys_Time Packed Record : 64 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Seconds	Interfaces. Unsigned_32	0 to 4294967295	32	0	31
Subseconds	Interfaces. Unsigned_32	0 to 4294967295	32	32	63

Field Descriptions:

- **Seconds** - The number of seconds elapsed since epoch.
- **Subseconds** - The number of 1/(2^32) sub-seconds.

Tick.T:

The tick datatype used for periodic scheduling. Included in this type is the Time associated with a tick and a count.

Table 29: Tick Packed Record : 96 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Count	Interfaces. Unsigned_32	0 to 4294967295	32	64	95

Field Descriptions:

- **Time** - The timestamp associated with the tick.
- **Count** - The cycle number of the tick.

5.3 Enumerations

The following section outlines any enumerations used in the component.

Command_Enums.Command_Response_Status.E:

This status enumeration provides information on the success/failure of a command through the command response connector.

Table 30: Command_Response_Status Literals:

Name	Value	Description
Success	0	Command was passed to the handler and successfully executed.
Failure	1	Command was passed to the handler not successfully executed.
Id_Error	2	Command id was not valid.
Validation_Error	3	Command parameters were not successfully validated.
Length_Error	4	Command length was not correct.
Dropped	5	Command overflowed a component queue and was dropped.
Register	6	This status is used to register a command with the command routing system.
Register_Source	7	This status is used to register command sender's source id with the command router for command response forwarding.

Event_Limiter_Enums.Issue_Packet_Type.E:

An enumeration for commands once the state is change for ground to determine if they want to send the state packet or not

Table 31: Issue_Packet_Type Literals:

Name	Value	Description
No_Issue	0	Packet will not be issued
Issue	1	Packet will be issued

Two_Counter_Entry_Enums.Event_State_Type.E:

This is a single bit identifying if the event limiter is enabled for a particular ID.

Table 32: Event_State_Type Literals:

Name	Value	Description
Disabled	0	Event will not be limited
Enabled	1	Event will be limited