Event Filter

Component Design Document

1 Description

The Event Filter component is used to filter out event IDs. The component takes in a range of IDs valid for filtering and an initial list of event ID's to filter. The filter utilizes a specific package to hold all of the ID and associated states. The package will determine if a ID needs to be filtered or not and return that to the component. The component will then either forward the event or perform no action if it is a filtered event. The component also has the capability to change the state of a single event ID, a range of event IDs, and a global state for turning off filtering all together. In this case the state of filtering for each ID will be maintained for when the master state is set back to enabled. Lastly, there is a packet that contains all the bit information of each ID's state and can be dumped by command.

2 Requirements

These are the requirements for the Event Filter component.

1. The Event Filter component shall accept all events and pass then on if not filtered or out of the begining set range of IDs.
2. The Event Filter component shall track the number of events that are filtered.
3. The Event Filter component shall not filter events if they are set to unfiltered.
4. The Event Filter component shall have the ability to filter or unfilter events by command.
5. The Event Filter 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** - 7

- **Number of Parameters** - *None*
- **Number of Events** - 12
- **Number of Faults** - *None*
- **Number of Data Products** - 3
- **Number of Data Dependencies** - *None*
- **Number of Packets** - 1

3.2 Diagram

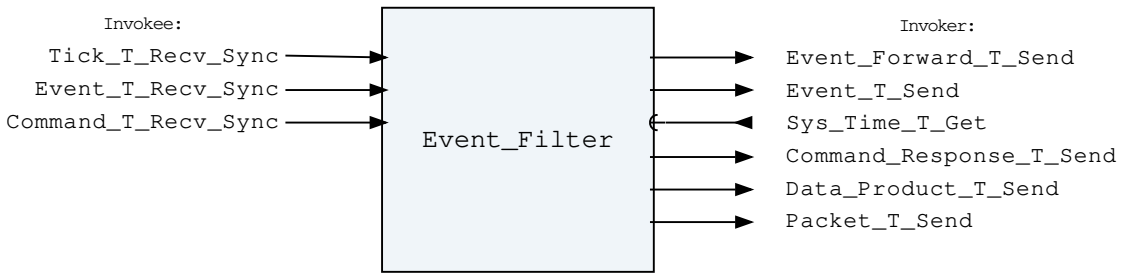

Figure 1: Event Filter component diagram.

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 Filter 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 record the number of events that have been filtered and send the state packet if it was requested.
- **Event_T_Recv_Sync** - Events are received synchronously on this connector and are passed along or filtered.
- **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 Filter 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 filtering is disabled, or if unknown events come in.
- **Event_T_Send** - The Event connector to send the events specifc 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 Filter 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 Filter Implementation Initialization Parameters

Name	Type	Default Value
`Event_Id_Start_Range`	`Event_Types.Event_Id`	*None provided*
`Event_Id_End_Range`	`Event_Types.Event_Id`	*None provided*
`Event_Filter_List`	`Event_Filter_Entry.Event_Id_List`	`(1..0=>0)`

Parameter Descriptions:

- **`Event_Id_Start_Range`** - The event ID that begins the range of ids that the component will include for filtering of events.
- **`Event_Id_End_Range`** - The event ID that ends the range of ids that the component will include for filtering of events.
- **`Event_Filter_List`** - A list of event IDs that are filtered by default

3.6 Commands

These are the commands for the event filter component.

Table 5: Event Filter Commands

Local ID	Command Name	Argument Type
0	`Filter_Event`	`Event_Filter_Single_Event_Cmd_Type.T`
1	`Unfilter_Event`	`Event_Filter_Single_Event_Cmd_Type.T`
2	`Filter_Event_Range`	`Event_Filter_Id_Range.T`
3	`Unfilter_Event_Range`	`Event_Filter_Id_Range.T`
4	`Enable_Event_Filtering`	-
5	`Disable_Event_Filtering`	-
6	`Dump_Event_States`	-

Command Descriptions:

- **`Filter_Event`** - Enable the event filtering for a specific event ID.
- **`Unfilter_Event`** - Disable the event filtering for a specific event ID.
- **`Filter_Event_Range`** - Enable the event filtering for a specific range of event IDs.

- **Unfilter_Event_Range** - Disable the event filtering for a specific range of event IDs.
- **Enable_Event_Filtering** - Enable the component to filter events that have been set to be filtered.
- **Disable_Event_Filtering** - Disable the component so that all events will not be filtered. The event states will be maintained for when re-enabled.
- **Dump_Event_States** - Dump a packet for the state of all events pertaining to if they are filtered or not.

3.7 Parameters

The Event Filter component has no parameters.

3.8 Events

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

Table 6: Event Filter Events

Local ID	Event Name	Parameter Type
0	Invalid_Command_Received	Invalid_Command_Info.T
1	Filtered_Event	Event_Filter_Single_Event_Cmd_Type.T
2	Unfiltered_Event	Event_Filter_Single_Event_Cmd_Type.T
3	Filtered_Event_Range	Event_Filter_Id_Range.T
4	Unfiltered_Event_Range	Event_Filter_Id_Range.T
5	Enable_Event_Filter	-
6	Disable_Event_Filter	-
7	Filter_Event_Invalid_Id	Event_Filter_Single_Event_Cmd_Type.T
8	Unfilter_Event_Invalid_Id	Event_Filter_Single_Event_Cmd_Type.T
9	Filter_Event_Range_Invalid_Id	Event_Filter_Id_Range.T
10	Unfilter_Event_Range_Invalid_Id	Event_Filter_Id_Range.T
11	Dump_Event_States_Recieved	-

Event Descriptions:

- **Invalid_Command_Received** - A command was received with invalid parameters.
- **Filtered_Event** - This event indicates that the state of an event was set to enabled for the filter.
- **Unfiltered_Event** - This event indicates that the state of an event was set to disabled for the filter.
- **Filtered_Event_Range** - This event indicates that the state of a range of events were set to enabled for the filter.
- **Unfiltered_Event_Range** - This event indicates that the state of a range of events were set to disabled for the filter.
- **Enable_Event_Filter** - This event indicates that the state of all events were set to enabled for the filter, but kept the internal state.
- **Disable_Event_Filter** - This event indicates that the state of all events were set to disabled for the filter, but kept the internal state.
- **Filter_Event_Invalid_Id** - This event indicates that the command to change the event

state to enabled failed since the event ID was out of range.

- **Unfilter_Event_Invalid_Id** - This event indicates that the command to change the event state to disable failed since the event ID was out of range.
- **Filter_Event_Range_Invalid_Id** - This event indicates that changing the state for the range to enabled, failed due to an invalid id.
- **Unfilter_Event_Range_Invalid_Id** - This event indicates that changing the state for the range to disabled, failed due to an invalid id.
- **Dump_Event_States_Recieved** - 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 filter component.

Table 7: Event Filter Data Products

Local ID	Data Product Name	Type
0x0000 (0)	Total_Events_Filtered	Packed_U32.T
0x0001 (1)	Total_Events_Unfiltered	Packed_U32.T
0x0002 (2)	Component_Filter_State	Event_Component_State_Type.T

Data Product Descriptions:

- **Total_Events_Filtered** - The total number of events that have been filtered for the components lifetime.
- **Total_Events_Unfiltered** - The total number of events that have been passed through for the components lifetime. Does not include out of range IDs.
- **Component_Filter_State** - The state of the master switch for filtering events.

3.10 Packets

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

Table 8: Event Filter Packets

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

Packet Descriptions:

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

4 Unit Tests

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

4.1 *Event_ Filter_ Tests* Test Suite

This is a unit test suite for the Event Filter component

Test Descriptions:

- **`Test_Received_Event`** - This unit test is to test an event that was received over the event connector and was either filtered or not.
- **`Test_Data_Products`** - This unit test is to test the data products for the component state and total filtered count
- **`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_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 : 808 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	-	768	40	807	Header.Arg_ Buffer_Length

Field Descriptions:

- **`Header`** - The command header
- **`Arg_Buffer`** - A buffer to 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 96	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_Component_State_Type.T:

This record is for the data product that indicates if the event filter component is enabled for filtering or disabled all together.

Table 15: Event_Component_State_Type Packed Record : 8 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Component_Filter_State	Event_Filter_Entry_Enums.Global_Filter_State.E	0 => Disabled 1 => Enabled	8	0	7

Field Descriptions:

- **Component_Filter_State** - Flag to indicate if the component is enabled or disabled at a overriding level from the individal event states

Event_Filter_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 16: Event_Filter_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_Filter_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_Filter_Single_Event_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 17: Event_Filter_Single_Event_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_Filter_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

Event_Header.T:

Generic event packet for holding arbitrary events

Table 18: 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 19: 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

Invalid_Command_Info.T:

Record for holding information about an invalid command

Table 20: 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 unkown 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_U32.T:

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

Table 21: 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 22: 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 23: 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 24: 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 25: 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 enumerations provides information on the success/failure of a command through the command response connector.

Table 26: 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_Filter_Entry_Enums.Global_Filter_State.E:

This is an enumeration indicating if the filter component is enabled/disabled

Table 27: Global_Filter_State Literals:

Name	Value	Description
Disabled	0	Individual event states will be ignored and no events will be filtered
Enabled	1	Individual event states will be used to determine if the event needs to be filtered

Event_Filter_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 28: Issue_Packet_Type Literals:

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