Event Text Logger

Component Design Document

1 Description

The Event Text Logger component receives events on an asynchronous queue and prints them either to the terminal or to a file as it receives them. The print statements for events are generated per the assembly that the Event Text Logger is attached to. By pointing the generator to a particular assembly model, you enable it to recognize and print certain events that are present in that assembly.

2 Requirements

No requirements have been specified for this component.

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

3.2 Diagram

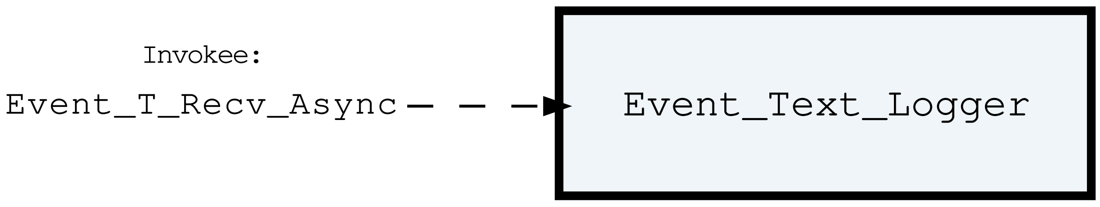

Figure 1: Event Text Logger component diagram.

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 Text Logger Invokee Connectors

Name	Kind	Type	Return_Type	Count
`Event_T_Recv_ Async`	`recv_async`	`Event.T`	-	1

Connector Descriptions:

- **`Event_T_Recv_Async`** - Events are received asynchronously on this connector.

3.3.2 Internal Queue

This component contains an internal first-in-first-out (FIFO) queue to handle asynchronous messages. This queue is sized at initialization as a configurable number of bytes. Determining the size of the component queue can be difficult. The following table lists the connectors that will put asynchronous messages onto the queue, and the maximum sizes of each of those messages on the queue. Note that each message put onto the queue also incurs an overhead on the queue of 5 additional bytes, which is included in the max message size below:

Table 2: Event Text Logger Asynchronous Connectors

Name	Type	Max Size (bytes)
`Event_T_Recv_Async`	`Event.T`	48

If you are unsure how to size the queue of this component, it is recommended that you make the queue size a multiple of the largest size found above.

3.3.3 Invoker Connectors

None

3.4 Initialization

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

3.4.1 Component Instantiation

The discriminant for this component takes an access of type Event_To_Text_Function. The function provided should translate an event to a string for any given event in an assembly. A package with this function implementation is autocoded for each assembly, and can be passed into the Event Text Logger to configure it for a given assembly. This component contains the following instantiation parameters in its discriminant:

Table 3: Event Text Logger Instantiation Parameters

Name	Type
`Event_To_Text`	`Event_To_Text_Function_Access`

Parameter Descriptions:

- **`Event_To_Text`** - An access to an event to text function.

3.4.2 Component Base Initialization

This component achieves base class initialization using the `init_Base` subprogram. This subprogram requires the following parameters:

Table 4: Event Text Logger Base Initialization Parameters

Name	Type
`Queue_Size`	`Natural`

Parameter Descriptions:

- **`Queue_Size`** - The number of bytes that can be stored in the component's internal queue.

3.4.3 Component Set ID Bases

This component contains no commands, events, packets, faults or data products that need base indentifiers.

3.4.4 Component Map Data Dependencies

This component contains no data dependencies.

3.4.5 Component Implementation Initialization

This component contains no implementation class initialization, meaning there is no `init` subprogram for this component.

4 Unit Tests

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

4.1 *Tests* Test Suite

This is a unit test suite for the Event Text Logger component

Test Descriptions:

- **Test_Event_Printing** - This unit test sends events to the event text logger and expects them to be printed to the screen.

5 Appendix

5.1 Preamble

This component contains the following preamble code. This is inline Ada code included in the component model that is usually used to define types or instantiate generic packages used by the component. Preamble code is inserted as the top line of the component base package specification.

```
type Event_To_Text_Function_Access is not null access function (Evt : in
↪  Event.T) return String;
```

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..

Event.T:

Generic event packet for holding arbitrary events

Table 5: 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_Header.T:

Generic event packet for holding arbitrary events

Table 6: 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

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 7: 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.