Command Rejector

Component Design Document

1 Description

This component is initialized with a list of commands to reject. The component receives commands, and checks their IDs against the reject command list. If a command is found in the list, then it is dropped and reported as an error packet. Commands that are not on the reject list are always forwarded. The reject command list is stored internally as a binary tree data structure that can determine if a command should be rejected or not in O(log(n)) time, where n is the number of commands to reject. Since most systems only manage a handful of commands on the reject list, the performance of this component should be acceptable for most missions. A common application for this component is to actively disallow commands eminating from certain sources, such as an onboard command sequence.

2 Requirements

The requirements for the Command Rejector component are specified below.

1. The component shall be configured with a list of command IDs to reject upon initialization.
2. The component shall reject commands if found in the command reject list.
3. The component shall report the number of commands rejected in telemetry

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** - 6
- **Number of Invokee Connectors** - 1
- **Number of Invoker Connectors** - 5
- **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** - 1
- **Number of Faults** - *None*

- **Number of Data Products** - 1
- **Number of Data Dependencies** - *None*
- **Number of Packets** - 1

3.2 Diagram

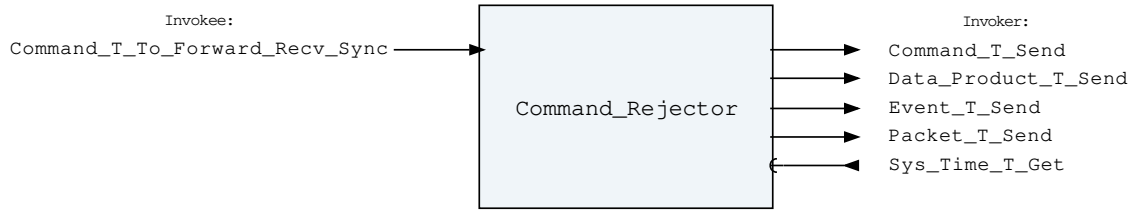

Figure 1: Command Rejector component diagram.

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: Command Rejector Invokee Connectors

Name	Kind	Type	Return_Type	Count
Command_T_To_Forward_Recv_Sync	recv_sync	Command.T	-	1

Connector Descriptions:

- **Command_T_To_Forward_Recv_Sync** - Commands received on this connector will be checked against the command reject list. Commands not found in the command reject list they will be forwarded.

3.3.2 Invoker Connectors

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

Table 2: Command Rejector Invoker Connectors

Name	Kind	Type	Return_Type	Count
Command_T_Send	send	Command.T	-	1
Data_Product_T_Send	send	Data_Product.T	-	1
Event_T_Send	send	Event.T	-	1
Packet_T_Send	send	Packet.T	-	1
Sys_Time_T_Get	get	-	Sys_Time.T	1

Connector Descriptions:

- **Command_T_Send** - The packet send connector

- **`Data_Product_T_Send`** - Data products are sent out of this connector.
- **`Event_T_Send`** - Events are sent out of this connector.
- **`Packet_T_Send`** - Error packets are sent out of this connector.
- **`Sys_Time_T_Get`** - The system time is retrieved via this connector.

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: Command Rejector Set Id Bases Parameters

Name	Type
`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:
- **`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. This component requires a list of command ID to reject at initialization. The `init` subprogram requires the following parameters:

Table 4: Command Rejector Implementation Initialization Parameters

Name	Type	Default Value
`Command_Id_Reject_List`	`Command_Id_List`	*None provided*

Parameter Descriptions:

- **`Command_Id_Reject_List`** - The list of command IDs to reject.

3.6 Commands

The Command Rejector component has no commands.

3.7 Parameters

The Command Rejector component has no parameters.

3.8 Events

Below is a list of the events for the Command Rejector component.

Table 5: Command Rejector Events

Local ID	Event Name	Parameter Type
0	`Rejected_Command`	`Command_Header.T`

Event Descriptions:

- **`Rejected_Command`** - A command was rejected (dropped) because it was found in the reject list.

3.9 Data Products

Data products for the Command Rejector.

Table 6: Command Rejector Data Products

Local ID	Data Product Name	Type
0x0000 (0)	`Rejected_Command_Count`	`Packed_U16.T`

Data Product Descriptions:

- **`Rejected_Command_Count`** - The number of received commands rejected because they were on the reject list.

3.10 Packets

Packets for the Command Rejector component.

Table 7: Command Rejector Packets

Local ID	Packet Name	Type
0x0000 (0)	`Error_Packet`	`Command.T`

Packet Descriptions:

- **`Error_Packet`** - This packet contains a command that was dropped due to being on the reject list.

4 Unit Tests

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

4.1 *Command_ Rejector_ Tests* Test Suite

This is a unit test suite for the Command Rejector.

Test Descriptions:

- **`Test_Initialization`** - This unit test tests all permutations of initializing the component and makes sure improper initialization results in a runtime assertion.
- **`Test_Command_Accept`** - This unit test tests that a command not on the reject list is passed along.
- **`Test_Command_Reject`** - This unit test tests that a command on the reject list is dropped by the component.

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.

```
1 type Command_Id_List is array (Natural range <>) of Command_Types.Command_Id;
```

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 8: 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 9: 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

Data_Product.T:

Generic data product packet for holding arbitrary data types

Table 10: 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 11: 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 12: 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 13: 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

Packed_U16.T:

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

Table 14: 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.

Packet.T:

Generic packet for holding arbitrary data

Table 15: 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 16: 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 17: 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.

5.3 Enumerations

No enumerations found in component.