Fault Responses Generator

Autocoder User Guide

1 Description

The purpose of this generator is to provide a user friendly way of defining the command responses to system faults handled by the Fault Correction component. The generator takes a YAML model file as input which specifies fault, the command response (and associated command arguments) and some configuration parameters for handling the fault response.

Note the example shown in this documentation is used in the unit test of this component so that the reader of this document can see it being used in context. Please refer to the unit test code for more details on how this generator can be used.

2 Schema

The following pykwalify schema is used to validate the input YAML model. The schema is commented to show what each of the available YAML keys are and what they accomplish. Even without knowing the specifics of pykwalify schemas, you should be able to gleam some knowledge from the file below.

```
1  ---
2  # This schema describes the yaml format for a set of fault responses.
3  type: map
4  mapping:
5    # Many "with" dependencies are automatically deduced and included by
6    # the generator. If you want to manually add a "with" statement, you
7    # can list the names of the packages here. This can be useful if you need
8    # to include a package that declares an enumeration used in a command
9    # argument.
10   with:
11     seq:
12       - type: str
13     required: False
14   # Description of the fault response set.
15   description:
16     type: str
17     required: False
18   # List of fault responses to include
19   fault_responses:
20     seq:
21       - type: map
22         mapping:
23           # The name of the fault in the form
             ↪  Component_Instance_Name.Fault_Name. This gets translated
24           # to a Fault_Id type internally.
25           fault:
26             type: str
27             required: True
28           # Latching flag. When True, an incoming fault will cause a command
             ↪  response to be sent only once
```

```
            # until the latch is cleared via ground command. When False, a command response is sent every time
            # a fault of this ID is received.
            latching:
              type: bool
              required: True
            # Startup state. When enabled, the fault response is enabled at startup. When disabled, the fault
            # response is disabled at startup.
            startup_state:
              type: str
              enum: ['enabled', 'disabled']
              required: True
            # The name of the command response in the form Component_Instance_Name.Command_Name. This gets
            # translated into a Command.T type internally.
            command_response:
              type: str
              required: True
            # The argument to serialize into the command response. This is required if the command response
            # includes command arguments. The argument should be supplied in the form:
            #   "(Name_1 => Value_1, Name_2 => Value_2, etc.)"
            # Where the names and values fill in the command argument packed record type.
            command_arg:
              type: str
              required: False
            # Description of the fault response.
            description:
              type: str
              required: False
    # A  sequence store must have at least one slot.
    range:
      min: 1
    required: True
```

3 Example Input

The following is an example fault responses input yaml file. Model files must be named in the form *optional_name.assembly_name.fault_responses.yaml* where *optional_name* is the specific name of the fault response set and is only necessary if there is more than one Fault Correction component instances in an assembly. The *assembly_name* is the assembly which the fault responses will be used, and the rest of the model file name must remain as shown. Generally this file is created in the same directory or near to the assembly model file. This example adheres to the schema shown in the previous section, and is commented to give clarification.

```
---
# Set the fault responses description.
description: This is an example fault response table for the Fault Correction unit test.
# Below is the set of responses as specified by the user.
fault_responses:
  - fault: Component_A.Fault_1
    latching: False
    startup_state: enabled
```

```
    command_response: Component_A.Command_1
    description: The fault response for Component A Fault 1.
  - fault: Component_A.Fault_2
    latching: True
    startup_state: enabled
    command_response: Component_A.Command_2
    command_arg: "(0, 0)" # Sys_Time.T
    description: The fault response for Component A Fault 2.
  - fault: Component_B.Fault_3
    latching: True
    startup_state: enabled
    command_response: Component_B.Command_3
    command_arg: "(Value => 18)" # Packed_U32.T
    description: The fault response for Component B Fault 3.
  - fault: Component_B.Fault_1
    latching: False
    startup_state: disabled
    command_response: Component_C.Command_1
    description: The fault response for Component B Fault 1.
  - fault: Component_C.Fault_1
    latching: False
    startup_state: disabled
    command_response: Component_B.Command_1
    description: The fault response for Component C Fault 1.
```

As can be seen, specifying the fault responses consists of listing the faults and corresponding command responses. Each fault may be enabled or disable at startup. Each fault may be configured as latching (meaning a command response is only sent out the first time that fault has been received by that component) and non-latching (meaning a command response is sent out each time a fault is received by the component. A latched fault can be cleared by command in the Fault Correction component.

4 Example Output

The example input shown in the previous section produces the following Ada output. The `Fault_Response_List` variable should be passed into the Fault Correction component's `Init` procedure during assembly initialization.

The main job of the generator in this case was to verify the input YAML for validity and then to translate the data to an Ada data structure for use by the component.

```
--------------------------------------------------------------------------------
-- This file was autogenerated from /vagrant/adamant/src/components/fault_correction/test/test_assembly/test_assembly.fault_responses.yaml on 2022-04-01 19:53.
--
-- Copyright: The University of Colorado, Laboratory for Atmospheric and Space Physics (LASP)
--------------------------------------------------------------------------------

-- Standard includes:
with Fault_Correction_Types;
with Basic_Types; use Basic_Types;
with Command_Types;
with Fault_Correction_Enums; use Fault_Correction_Enums.Startup_Status_Type; use Fault_Correction_Enums.Latching_Type;
with Sys_Time;
```

```
13  with Packed_U32;
14
15  -- This is an example fault response table for the Fault Correction unit test.
16  package Test_Assembly_Fault_Responses is
17
18    --
19    -- The fault responses.
20    --
21
22    -- Fault response configuration definition for Component_A.Fault_1:
23    -- The fault response for Component A Fault 1.
24    Component_A_Fault_1_Response : constant
       ↪  Fault_Correction_Types.Fault_Response_Config := (
25      -- Set fault ID for Component_A.Fault_1:
26      Id => 1,
27      -- Set latching configuration:
28      Latching => Non_Latching,
29      -- Set startup state:
30      Startup_State => Enabled,
31      -- Set command response as Component_A.Command_1:
32      Command_Response => (
33        Header => (
34          Source_Id => 0,
35          Id => 1,
36          Arg_Buffer_Length => 0
37        ),
38        Arg_Buffer => (
39          others => 0
40        )
41      )
42    );
43
44    -- Fault response configuration definition for Component_A.Fault_2:
45    -- The fault response for Component A Fault 2.
46    Component_A_Fault_2_Response : constant
       ↪  Fault_Correction_Types.Fault_Response_Config := (
47      -- Set fault ID for Component_A.Fault_2:
48      Id => 2,
49      -- Set latching configuration:
50      Latching => Latching,
51      -- Set startup state:
52      Startup_State => Enabled,
53      -- Set command response as Component_A.Command_2:
54      Command_Response => (
55        Header => (
56          Source_Id => 0,
57          Id => 2,
58          Arg_Buffer_Length => 8
59        ),
60        Arg_Buffer => (
61          Sys_Time.Serialization.To_Byte_Array ((0, 0)) &
62          (0 .. Command_Types.Command_Arg_Buffer_Type'Length -
             ↪  Sys_Time.Size_In_Bytes - 1 => 0)
63        )
64      )
65    );
66
67    -- Fault response configuration definition for Component_B.Fault_3:
68    -- The fault response for Component B Fault 3.
69    Component_B_Fault_3_Response : constant
       ↪  Fault_Correction_Types.Fault_Response_Config := (
```

```
    -- Set fault ID for Component_B.Fault_3:
    Id => 6,
    -- Set latching configuration:
    Latching => Latching,
    -- Set startup state:
    Startup_State => Enabled,
    -- Set command response as Component_B.Command_3:
    Command_Response => (
      Header => (
        Source_Id => 0,
        Id => 6,
        Arg_Buffer_Length => 4
      ),
      Arg_Buffer => (
        Packed_U32.Serialization.To_Byte_Array ((Value => 18)) &
        (0 .. Command_Types.Command_Arg_Buffer_Type'Length -
         ↪  Packed_U32.Size_In_Bytes - 1 => 0)
      )
    )
  );

  -- Fault response configuration definition for Component_B.Fault_1:
  -- The fault response for Component B Fault 1.
  Component_B_Fault_1_Response : constant
   ↪  Fault_Correction_Types.Fault_Response_Config := (
    -- Set fault ID for Component_B.Fault_1:
    Id => 4,
    -- Set latching configuration:
    Latching => Non_Latching,
    -- Set startup state:
    Startup_State => Disabled,
    -- Set command response as Component_C.Command_1:
    Command_Response => (
      Header => (
        Source_Id => 0,
        Id => 7,
        Arg_Buffer_Length => 0
      ),
      Arg_Buffer => (
        others => 0
      )
    )
  );

  -- Fault response configuration definition for Component_C.Fault_1:
  -- The fault response for Component C Fault 1.
  Component_C_Fault_1_Response : constant
   ↪  Fault_Correction_Types.Fault_Response_Config := (
    -- Set fault ID for Component_C.Fault_1:
    Id => 7,
    -- Set latching configuration:
    Latching => Non_Latching,
    -- Set startup state:
    Startup_State => Disabled,
    -- Set command response as Component_B.Command_1:
    Command_Response => (
      Header => (
        Source_Id => 0,
        Id => 4,
        Arg_Buffer_Length => 0
      ),
```

```
      Arg_Buffer => (
        others => 0
      )
    )
  );

  --
  -- The fault response configuration list:
  --

  Fault_Response_List : constant
   ↪  Fault_Correction_Types.Fault_Response_Config_List := (
    0 => Component_A_Fault_1_Response,
    1 => Component_A_Fault_2_Response,
    2 => Component_B_Fault_3_Response,
    3 => Component_B_Fault_1_Response,
    4 => Component_C_Fault_1_Response
  );

end Test_Assembly_Fault_Responses;
```