Ccsds Router Table Generator

Autocoder User Guide

1 Description

The purpose of this generator is to provide a user friendly way of creating a static CCSDS router table for use within a CCSDS Router component. The generator takes a YAML model file as input which specifies which destination component(s) each CCSDS packet, identified via APID, should be routed to. A CCSDS packet of a single APID can be routed to zero or many destination components. The table also specifies, for each APID, whether or not to track the sequence numbers of that packet, report out of order sequence numbers, and/or drop subsequent packets received with the same sequence number. From this information, the generator autocodes an Ada specification file which contains a data structure that should be passed to the CCSDS Router component upon initialization.

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. Model files must be named in the form *optional_name.assembly_name.ccsds_router_table.yaml* where *optional_name* is the specific name of this router table and is only necessary if there is more than one CCSDS Router component instance in an assembly. The *assemble_name* is the assembly which this router table will be used in, 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. 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 ccsds router table.
3  type: map
4  mapping:
5    # The name of the component that this router table is being constructed for.
     ↪  This component
6    # must exist in the assembly specified by this file's name otherwise an error
     ↪  will be thrown.
7    # The component's name must be specified so that the generator can verify
     ↪  that any routed
8    # connections in the table actually exist in the assembly.
9    ccsds_router_instance_name:
10     type: str
11     required: True
12   # Description of the table.
13   description:
14     type: str
15     required: False
16   # List of table entries. Each entry is a map between a CCSDS packet's APID
     ↪  and a list of destination
17   # of which to route that packet. At least one destination is required, but
     ↪  "ignore" can be used to
```

```yaml
  # not route a certain packet.
  table:
    seq:
      - type: map
        mapping:
          # The APID of the CCSDS packet we intend to route.
          apid:
            type: int
            required: True
          # The names of the components to route this packet to. Note, this uses the actual name
          # of the component as found in the assembly. If the component name does not exist, an
          # error will be thrown. This is a list, and at least one item MUST be specified. If you
          # do not want to route packets of this APID, then you may list "ignore" here and the
          # packet will not be routed.
          destinations:
            seq:
              - type: str
            required: True
          # Description of this router table entry.
          description:
            type: str
            required: False
          # The sequence count check mode to perform on this APID. The options are as follows:
          #
          #  no_check   - CCSDS packets of this APID will not be checked for sequence count
          #      warn   - CCSDS packets of this APID will produce a warning event if a packet
          #               is received with an unexpected (non-incrementing) sequence count
          #  drop_dupes - the second CCSDS packet of this APID will be dropped if two packets
          #               in a row are seen with the same APID. Note that 'drop_dupes' also
          #               implies the same behavior as 'warn' listed above.
          sequence_check:
            type: str
            enum: ['no_check', 'warn', 'drop_dupes']
            required: False
    range:
      min: 1
    required: True
```

3 Example Input

The following is an example routing table input yaml file. Model files must be named in the form *optional_name.assembly_name.ccsds_router_table.yaml* where *optional_name* is the specific name of this router table and is only necessary if there is more than one CCSDS Router component instance in an assembly. The *assemble_name* is the assembly which this router table will be used in, 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.

```yaml
---
# Set the router table description.
description: This is an example router table for the CCSDS Router unit test.
# Below, we specify the name of the CCSDS Router component in the
# assembly that we want to associate this table with. This allows
# the generator to check out routing connections to make sure
# they actually exist.
ccsds_router_instance_name: Ccsds_Router_Instance
# Below is the actual router table as specified by the user. We
# map CCSDS packet APIDs to a list of corresponding destinations.
# In this table, we also specify the sequence check mode of each
# APID. Sequence check mode options are as follows:
#
#  no_check   - CCSDS packets of this APID will not be checked for sequence count
#      warn   - CCSDS packets of this APID will produce a warning event if a packet
#               is received with an unexpected (non-incrementing) sequence count
#  drop_dupes - the second CCSDS packet of this APID will be dropped if two packets
#               in a row are seen with the same APID. Note that 'drop_dupes' also
#               implies the same behavior as 'warn' listed above.
table:
  # For APID 1 we only want to route it to Component_A and nowhere else.
  # By default, if we do not specify a "sequence_check" then it will be
  # assumed that we want "no_check" on this APID.
  - apid: 1
    destinations:
      - Component_A
  # For APID 2 we want to route to two components, Component_A and
  # Component_C. Here we specifically specify that we don't want
  # the CCSDS Router to keep track of sequence numbers for packets
  # of this APID
  - apid: 2
    destinations:
      - Component_A
      - Component_C
    sequence_check: no_check
  # The following 2 APIDs specify 'warn' for sequence check so that
  # an error event is produced when a non-subsequent sequence number is
  # received. Note, when this happens, the CCSDS packets are still routed
  # according to the table. They are NOT dropped.
  - apid: 3
    destinations:
      - Component_A
    sequence_check: warn
  - apid: 4
    destinations:
      - Component_B
    sequence_check: warn
  # The following 2 APIDs specify 'drop_dupes' for sequence check so
  # that if two packets are received in a row with identical sequence
  # counts, the second packet is dropped and an error is produced.
  # Note that 'drop_dupes' also implies the behavior found in 'warn'.
  - apid: 5
    destinations:
      - Component_B
    sequence_check: drop_dupes
  - apid: 6
    destinations:
```

```
      # Because Component_D is attached to the CCSDS router twice, on two different connectors,
      # we need to specify which connector to route this APID to. Otherwise a modeling error will result.
      - Component_D.Ccsds_Space_Packet_T_Recv_Sync
    sequence_check: drop_dupes
  # The following 2 APID's use "ignore" as a destination for the CCSDS
  # packet. "ignore" is a special keyword that tells the autocoder to
  # still recognize the CCSDS packet (ie. not drop it as an unrecognized
  # packet) but do not route it. This can be useful if you still want to
  # check sequence counts for a certain packet, without actually routing
  # it to a destination.
  - apid: 7
    destinations:
      - ignore
      - Component_A
      # Because Component_D is attached to the CCSDS router twice, on two different connectors,
      # we need to specify which connector to route this APID to. Otherwise a modeling error will result.
      - Component_D.Ccsds_Space_Packet_T_Recv_Sync_2
    sequence_check: no_check
  # The tactic used here is to check the sequence counts of CCSDS packets
  # with APID 8, and produce warnings when necessary, but to not route
  # them to any destination. "ignore" is used to achieve this.
  - apid: 8
    destinations:
      - ignore
    sequence_check: warn
```

4 Example Output

The example input shown in the previous section produces the following Ada output. The `Router_Table` variable should be passed into the CCSDS Router component's `Init` function during assembly initialization.

The main job of the generator in this case was to verify the input YAML router table for validity and then to translate component names into connector output indexes, which the CCSDS Router then uses directly.

```
--------------------------------------------------------------------------------
-- This file was autogenerated from /vagrant/adamant/src/components/ccsds_router/test/test_assembly/test_assembly.ccsds_router_table.yaml on 2022-04-01 19:33.
--
-- Copyright: The University of Colorado, Laboratory for Atmospheric and Space Physics (LASP)
--------------------------------------------------------------------------------

-- Standard includes:
with Ccsds_Router_Types; use Ccsds_Router_Types;

-- Ccsds Router Table for component instance: Ccsds_Router_Instance
-- This is an example router table for the CCSDS Router unit test.
package Test_Assembly_Ccsds_Router_Table is

```

```
14    -- APID destination tables:
15    -- Destination table for APID: 1
16    destination_Table_1 : aliased Destination_Table := (0 => 1);
17    -- Destination table for APID: 2
18    destination_Table_2 : aliased Destination_Table := (0 => 1, 1 => 3);
19    -- Destination table for APID: 3
20    destination_Table_3 : aliased Destination_Table := (0 => 1);
21    -- Destination table for APID: 4
22    destination_Table_4 : aliased Destination_Table := (0 => 2);
23    -- Destination table for APID: 5
24    destination_Table_5 : aliased Destination_Table := (0 => 2);
25    -- Destination table for APID: 6
26    destination_Table_6 : aliased Destination_Table := (0 => 4);
27    -- Destination table for APID: 7
28    destination_Table_7 : aliased Destination_Table := (0 => 1, 1 => 5);
29    -- Destination table for APID: 8
30    -- Ignore this APID, there is no destination for it.
31
32    -- Router table entries:
33    Router_Table : constant Router_Table_Entry_Array := (
34      -- Table entry for APID: 1
35      0 => (Apid => 1, Destinations => destination_Table_1'Access,
        ↪  Sequence_Count_Mode => No_Check),
36      -- Table entry for APID: 2
37      1 => (Apid => 2, Destinations => destination_Table_2'Access,
        ↪  Sequence_Count_Mode => No_Check),
38      -- Table entry for APID: 3
39      2 => (Apid => 3, Destinations => destination_Table_3'Access,
        ↪  Sequence_Count_Mode => Warn),
40      -- Table entry for APID: 4
41      3 => (Apid => 4, Destinations => destination_Table_4'Access,
        ↪  Sequence_Count_Mode => Warn),
42      -- Table entry for APID: 5
43      4 => (Apid => 5, Destinations => destination_Table_5'Access,
        ↪  Sequence_Count_Mode => Drop_Dupes),
44      -- Table entry for APID: 6
45      5 => (Apid => 6, Destinations => destination_Table_6'Access,
        ↪  Sequence_Count_Mode => Drop_Dupes),
46      -- Table entry for APID: 7
47      6 => (Apid => 7, Destinations => destination_Table_7'Access,
        ↪  Sequence_Count_Mode => No_Check),
48      -- Table entry for APID: 8
49      7 => (Apid => 8, Destinations => null, Sequence_Count_Mode => Warn)
50    );
51
52  end Test_Assembly_Ccsds_Router_Table;
```