Ccsds Downsampler Generator

Autocoder User Guide

1 Description

The purpose of this generator is to provide the user a way to define the packets that they would like downsampled. The generator takes a YAML model file as an input from which it constructs an Ada specification file that contains a list as the input parameter for initialization in the component. The component is required to know the Apid of the desired packets to be downsampled as well as the filtering factor to downsample the packets. The generator will also create data products of the filter factor for each of the specified Apids in the list.

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 *assembly_name.ccsds_downsampler_filters.yaml*. The *assembly_name* is the assembly which these packets will be downsampled 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.

```
---
# This schema describes the yaml format for the ccsds downsampler list of
↪  filter factors with the associated APID.
type: map
mapping:
  description:
    type: str
    required: False
  # List of data products to include in the suite.
  downsampled_packets:
    seq:
      - type: map
        mapping:
          # APID of the packet that we are setting a filter factor for
          apid:
            type: int
            required: True
          # Filter factor of the packet
          filter_factor:
            type: int
            required: True
          # Name of the packet.
          packet_name:
            type: str
```

```
            required: False
          # Description if desired
          description:
            type: str
            required: False
    # A ccsds downsampler must have at least one packet to downsample.
    range:
      min: 1
    required: True
```

3 Example Input

The following shows an example input YAML model that is used in the unit testing for the downsampler component. The example attempts to use all the variations of the optional inputs so that the user can see how they can use the options to generate their own desired input. 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.

```
---
description: Data products for the ccsds downsampler component.
downsampled_packets:
  - apid: 100
    filter_factor: 0
    packet_name: First_Packet
    description: The first filter packet
  - apid: 200
    filter_factor: 3
    packet_name: Second_Packet
    description: The second filter packet
  - apid: 300
    filter_factor: 1
    description: The unnamed filter packet
  - apid: 400
    filter_factor: 4
    packet_name: Last_Packet
  - apid: 500
    filter_factor: 2
```

As can be seen, specifying a packet to downsample simply consists of listing the Apid for that packet and the corresponding filter factor. There is also the capability to include a name which will be used as the name of the generated data product. If no name is specified, the generator will use a autogenerated name using the Apid.

4 Example Output

The example input shown in the previous section produces the following Ada output. The `Downsample_List` variable should be passed into the Ccsds Downsampler 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. The generator will also create data products as was mentioned before in this document

```
--------------------------------------------------------------------------------
--  This file was autogenerated from
    /vagrant/adamant/src/components/ccsds_downsampler/test/test_assembly/test_downsample_list.ccsds_downsampler_filters.yaml on 2022-04-01
    19:36.
--
-- Copyright: The University of Colorado, Laboratory for Atmospheric and Space
    Physics (LASP)
--------------------------------------------------------------------------------

-- Standard includes:
with Ccsds_Downsampler_Types; use Ccsds_Downsampler_Types;

-- Data products for the ccsds downsampler component.
package Test_Downsample_List is
  -- Size of the initial list of the downsampler
  Downsample_List_Size_In_Bytes : constant Natural := 20;
  -- The initial list for the downsampler
  Downsample_List : aliased Ccsds_Downsample_Packet_List := (
    -- First_Packet
    -- The first filter packet
    0 => (
      Apid => 100,
      Filter_Factor => 0
    ),
    -- Second_Packet
    -- The second filter packet
    1 => (
      Apid => 200,
      Filter_Factor => 3
    ),
    -- Apid_300
    -- The unnamed filter packet
    2 => (
      Apid => 300,
      Filter_Factor => 1
    ),
    -- Last_Packet
    3 => (
      Apid => 400,
      Filter_Factor => 4
    ),
    -- Apid_500
    4 => (
      Apid => 500,
      Filter_Factor => 2
    )
  );

  -- Access for the downsample list
  Downsample_List_Access : Ccsds_Downsample_Packet_List_Access :=
    Downsample_List'Access;

end Test_Downsample_List;
```