

Product Packets Generator

Autocoder User Guide

1 Description

The purpose of this generator is to provide a user friendly way of creating packets formed from a list of data products. The generator takes a YAML model file as input which specifies the packets to produce, the data products to put in each packet, the period that the packet will be emitted at, and whether the packet is enabled or disabled on startup. From this information, the generator autocodes an Ada specification file which contains a data structure that should be passed to the Product Packetizer 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.product_packets.yaml* where *optional_name* is the specific name of this set of packets and is only necessary if there is more than one Product Packetizer component instance in an assembly. The *assembly_name* is the assembly which these product packets 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 glean some knowledge from the file below.

```
1  ---
2  # This schema describes the yaml format for a data product packet suite.
3  type: map
4  mapping:
5    # Description of the packet suite.
6    description:
7      type: str
8      required: False
9    # Many "with" dependencies are automatically deduced and included by
10    # the generator. If you want to manually add a "with" statement, you
11    # can list the names of the packages here.
12    with:
13      seq:
14        - type: str
15        required: False
16    # List of packets to include in the suite.
17    packets:
18      seq:
19        - type: map
20          mapping:
21            # Name of the packet.
22            name:
```

```

23         type: str
24         required: True
25     # Description of the packet.
26     description:
27         type: str
28         required: False
29     # Identifier for the packet (in CCSDS this would be the APID).
30     id:
31         type: int
32         required: True
33     # Is the packet enabled or disabled upon initialization. By default
34     # packets are enabled if this is not specified. This field can be set
35     ↪ to:
36     #     True - Packet is enabled and sent periodically
37     #     False - Packet is disabled
38     #     On_Change - Packet is only sent when data products have changed
39     ↪ since last emission
40     enabled:
41         type: any
42         required: False
43     # The period (in ticks) in which to build the packet. This is the
44     ↪ value set upon
45     # initialization.
46     period:
47         type: str
48         required: True
49     # The offset (in ticks) at which to stagger the construction of this
50     ↪ packet. An offset of
51     # 5 will cause the packet to be built according to its period, but 5
52     ↪ ticks later than expected.
53     # Note that the offset should be less than the period otherwise it
54     ↪ will be mod'ed by the period
55     # so that it is less than the period. For example, if the period is 3
56     ↪ and the offset is set to
57     # 5, the actual offset used will be 2.
58     #
59     # This field can be used to stagger packet creation, allowing the user
60     ↪ to evenly distribute
61     # the work that this component does, so as to not cause cycle slips
62     ↪ when many packets need to
63     # be built on the same tick.
64     offset:
65         type: str
66         required: False
67     # If set to true then the packet is timestamped with the time found on
68     ↪ the incoming Tick.T
69     # instead of the current time as fetched via the time connector. By
70     ↪ default, if not specified
71     # this value is set to False.
72     use_tick_timestamp:
73         type: bool
74         required: False
75     # List of data products to include in packet
76     data_products:
77         seq:
78             - type: map
79               mapping:
80                 # The name of the data product. The name should be in the
81                 ↪ format
82                 # Component_Name.Data_Product_Name. The name is a required
83                 ↪ field unless

```

```

71         # pad_bytes is specified.
72         name:
73             type: str
74             required: False
75         # Produce an event if the data product is ever not available
76         ↪ when fetched. By default
77         # this is false.
78         event_on_missing:
79             type: bool
80             required: False
81         # Use this data product's timestamp as the packet timestamp.
82         ↪ This may only be set true for
83         # a single data product per packet. By default this value is
84         ↪ false.
85         use_timestamp:
86             type: bool
87             required: False
88         # Include this data product's timestamp just before its value
89         ↪ in the actual packet.
90         # By default this value is false.
91         include_timestamp:
92             type: bool
93             required: False
94         # When the packet's enabled field is set to On_Change, this
95         ↪ field determines whether
96         # this data product's timestamp should be checked to trigger
97         ↪ packet emission. If true,
98         # the packet will be sent if this data product has been
99         ↪ updated since the last packet
100        # emission. If false, this data product's timestamp is ignored
101        ↪ for on_change checks.
102        # By default this value is true if not specified.
103        used_for_on_change:
104            type: bool
105            required: False
106        # Pad bytes can be used to insert a n-number of bytes of
107        ↪ unused data into a packet. This is
108        # also useful to add in spacing for data products that do not
109        ↪ exist yet, but are expected
110        # in the packet. Pad bytes can only be specified if no other
111        ↪ fields are specified.
112        pad_bytes:
113            type: int
114            required: False
115    range:
116        min: 1
117        required: True
118    # A packet suite must have at least one packet.
119    range:
120        min: 1
121        required: True

```

3 Example Input

The following is an example product packet input yaml file. Model files must be named in the form *optional_name.assembly_name.product_packets.yaml* where *optional_name* is the specific name of the product packets and is only necessary if there is more than one Product Packetizer component instance in an assembly. The *assembly_name* is the assembly which these packets 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.

```
1  ---
2  description: This is an example set of packets.
3  # starting id...
4  # assuming 1 hz tick
5  packets:
6  - name: Packet_1 # must be unique, enforce by autocoder
7    description: This is packet 1.
8    id: 7
9    data_products:
10     - name: Test_Component_1_Instance.Data_Product_A
11       use_timestamp: False
12       include_timestamp: True
13       event_on_missing: True
14     - name: Test_Component_2_Instance.Data_Product_C
15       event_on_missing: False
16       use_timestamp: False
17       include_timestamp: False
18   period: "3" # create every 3 ticks
19   enabled: True
20 - name: Packet_2
21   id: 9
22   data_products:
23     - name: Test_Component_2_Instance.Data_Product_D
24     - name: Test_Component_1_Instance.Data_Product_B
25       use_timestamp: True
26   period: "1" # create every tick
27   offset: "0"
28   enabled: False
29 - name: Packet_3 # must be unique, enforce by autocoder
30   description: This is packet 1.
31   id: 8
32   use_tick_timestamp: False
33   data_products:
34     - name: Test_Component_1_Instance.Data_Product_A
35       use_timestamp: False
36       include_timestamp: True
37       event_on_missing: True
38     - name: Test_Component_2_Instance.Data_Product_C
39       event_on_missing: False
40       use_timestamp: False
41       include_timestamp: False
42   period: "3" # create every 3 ticks
43   offset: "5" # This should act like an offset of 2, but we are testing that
44     ↪ feature here.
45   enabled: False
46 - name: Packet_4
47   description: This packet tests padding
48   id: 12
49   use_tick_timestamp: True
50   data_products:
51     - pad_bytes: 5
52     - name: Test_Component_1_Instance.Data_Product_A
53       use_timestamp: False
54       include_timestamp: False
55       event_on_missing: False
56     - pad_bytes: 3
57   period: "1" # create every tick
58   offset: "0"
59   enabled: False
```

```

59 - name: Packet_5
60   id: 15
61   data_products:
62     - name: Product_Packetizer_Instance.Packet_4_Period
63     - name: Product_Packetizer_Instance.Packet_5_Period
64     - name: Product_Packetizer_Instance.Packet_3_Period
65   period: "2"
66   enabled: False

```

4 Example Output

The example input shown in the previous section produces the following Ada output. The `Packet_List` variable should be passed into the Product Packetizer component's discriminant during assembly initialization.

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

```

1  -- Standard includes:
2  with Product_Packet_Types; use Product_Packet_Types;
3  with Packet_Types;
4  with Sys_Time.Arithmetic;
5
6  -- This is an example set of packets.
7  package Test_Assembly_Product_Packets_Test_Packets is
8
9      -- Packet_1:
10     -- This is packet 1.
11
12     -- Packet_1 data product items:
13     -- Total packet buffer size: 160 bits
14     Packet_1_Items : aliased Packet_Items_Type := [
15         -- Item entry for Test_Component_1_Instance.Data_Product_A:
16         1 => (Data_Product_Id => 1, Use_Timestamp => False, Include_Timestamp =>
17             => True, Event_On_Missing => True, Used_For_On_Change => True,
18             => Packet_Period_Item => False, Size => 4),
19         -- Item entry for Test_Component_2_Instance.Data_Product_C:
20         2 => (Data_Product_Id => 3, Use_Timestamp => False, Include_Timestamp =>
21             => False, Event_On_Missing => False, Used_For_On_Change => True,
22             => Packet_Period_Item => False, Size => 10)
23     ];
24
25     -- Packet_1 packet description:
26     Packet_1_Description : Packet_Description_Type := (
27         Id => 7,
28         Items => Packet_1_Items'Access,
29         Period => 3,
30         Offset => 0,
31         Enabled => Product_Packet_Types.Enabled,
32         Use_Tick_Timestamp => False,
33         Count => Packet_Types.Sequence_Count_Mod_Type'First,
34         Send_Now => False,
35         Last_Emission_Time => Sys_Time.Arithmetic.Sys_Time_Zero
36     );
37
38     -- Packet_2:
39
40     -- Packet_2 data product items:
41     -- Total packet buffer size: 96 bits

```

```

38 Packet_2_Items : aliased Packet_Items_Type := [
39   -- Item entry for Test_Component_2_Instance.Data_Product_D:
40   1 => (Data_Product_Id => 4, Use_Timestamp => False, Include_Timestamp =>
41     ↳ False, Event_On_Missing => False, Used_For_On_Change => True,
42     ↳ Packet_Period_Item => False, Size => 2),
43   -- Item entry for Test_Component_1_Instance.Data_Product_B:
44   2 => (Data_Product_Id => 2, Use_Timestamp => True, Include_Timestamp =>
45     ↳ False, Event_On_Missing => False, Used_For_On_Change => True,
46     ↳ Packet_Period_Item => False, Size => 10)
47 ];
48
49 -- Packet_2 packet description:
50 Packet_2_Description : Packet_Description_Type := (
51   Id => 9,
52   Items => Packet_2_Items'Access,
53   Period => 1,
54   Offset => 0,
55   Enabled => Product_Packet_Types.Disabled,
56   Use_Tick_Timestamp => False,
57   Count => Packet_Types.Sequence_Count_Mod_Type'First,
58   Send_Now => False,
59   Last_Emission_Time => Sys_Time.Arithmetic.Sys_Time_Zero
60 );
61
62 -- Packet_3:
63 -- This is packet 1.
64
65 -- Packet_3 data product items:
66 -- Total packet buffer size: 160 bits
67 Packet_3_Items : aliased Packet_Items_Type := [
68   -- Item entry for Test_Component_1_Instance.Data_Product_A:
69   1 => (Data_Product_Id => 1, Use_Timestamp => False, Include_Timestamp =>
70     ↳ True, Event_On_Missing => True, Used_For_On_Change => True,
71     ↳ Packet_Period_Item => False, Size => 4),
72   -- Item entry for Test_Component_2_Instance.Data_Product_C:
73   2 => (Data_Product_Id => 3, Use_Timestamp => False, Include_Timestamp =>
74     ↳ False, Event_On_Missing => False, Used_For_On_Change => True,
75     ↳ Packet_Period_Item => False, Size => 10)
76 ];
77
78 -- Packet_3 packet description:
79 Packet_3_Description : Packet_Description_Type := (
80   Id => 8,
81   Items => Packet_3_Items'Access,
82   Period => 3,
83   Offset => 5,
84   Enabled => Product_Packet_Types.Disabled,
85   Use_Tick_Timestamp => False,
86   Count => Packet_Types.Sequence_Count_Mod_Type'First,
87   Send_Now => False,
88   Last_Emission_Time => Sys_Time.Arithmetic.Sys_Time_Zero
89 );
90
91 -- Packet_4:
92 -- This packet tests padding
93
94 -- Packet_4 data product items:
95 -- Total packet buffer size: 96 bits
96 Packet_4_Items : aliased Packet_Items_Type := [
97   -- Item entry for :
98   1 => (Data_Product_Id => 0, Use_Timestamp => False, Include_Timestamp =>
99     ↳ False, Event_On_Missing => False, Used_For_On_Change => True,
100    ↳ Packet_Period_Item => False, Size => 5),

```

```

91      -- Item entry for Test_Component_1_Instance.Data_Product_A:
92      2 => (Data_Product_Id => 1, Use_Timestamp => False, Include_Timestamp =>
93      ↪ False, Event_On_Missing => False, Used_For_On_Change => True,
94      ↪ Packet_Period_Item => False, Size => 4),
95      -- Item entry for :
96      3 => (Data_Product_Id => 0, Use_Timestamp => False, Include_Timestamp =>
97      ↪ False, Event_On_Missing => False, Used_For_On_Change => True,
98      ↪ Packet_Period_Item => False, Size => 3)
99  ];
100
101  -- Packet_4 packet description:
102  Packet_4_Description : Packet_Description_Type := (
103      Id => 12,
104      Items => Packet_4_Items'Access,
105      Period => 1,
106      Offset => 0,
107      Enabled => Product_Packet_Types.Disabled,
108      Use_Tick_Timestamp => True,
109      Count => Packet_Types.Sequence_Count_Mod_Type'First,
110      Send_Now => False,
111      Last_Emission_Time => Sys_Time.Arithmetic.Sys_Time_Zero
112  );
113
114  -- Packet_5:
115
116  -- Packet_5 data product items:
117  -- Total packet buffer size: 96 bits
118  Packet_5_Items : aliased Packet_Items_Type := [
119      -- Item entry for Product_Packetizer_Instance.Packet_4_Period:
120      1 => (Data_Product_Id => 4, Use_Timestamp => False, Include_Timestamp =>
121      ↪ False, Event_On_Missing => False, Used_For_On_Change => True,
122      ↪ Packet_Period_Item => True, Size => 4),
123      -- Item entry for Product_Packetizer_Instance.Packet_5_Period:
124      2 => (Data_Product_Id => 5, Use_Timestamp => False, Include_Timestamp =>
125      ↪ False, Event_On_Missing => False, Used_For_On_Change => True,
126      ↪ Packet_Period_Item => True, Size => 4),
127      -- Item entry for Product_Packetizer_Instance.Packet_3_Period:
128      3 => (Data_Product_Id => 3, Use_Timestamp => False, Include_Timestamp =>
129      ↪ False, Event_On_Missing => False, Used_For_On_Change => True,
130      ↪ Packet_Period_Item => True, Size => 4)
131  ];
132
133  -- Packet_5 packet description:
134  Packet_5_Description : Packet_Description_Type := (
135      Id => 15,
136      Items => Packet_5_Items'Access,
137      Period => 2,
138      Offset => 0,
139      Enabled => Product_Packet_Types.Disabled,
140      Use_Tick_Timestamp => False,
141      Count => Packet_Types.Sequence_Count_Mod_Type'First,
142      Send_Now => False,
143      Last_Emission_Time => Sys_Time.Arithmetic.Sys_Time_Zero
144  );
145
146  -- List of packets for the packetizer to build:
147  Packet_List : aliased Packet_Description_List_Type := [
148      1 => Packet_1_Description,
149      2 => Packet_2_Description,
150      3 => Packet_3_Description,
151      4 => Packet_4_Description,

```

```

142     5 => Packet_5_Description
143 ];
144
145 end Test_Assembly_Product_Packets_Test_Packets;

```

5 On-Change Packet Example

The Product Packetizer also supports “on-change” packets that emit only when one or more tracked data products have fresh timestamps relative to the last emission. The enabled enumerual value `On_Change` puts a packet into this mode, while the optional `used_for_on_change` flag attached to each item allows you to include data products that do not participate in the change detection. Any data product with `used_for_on_change = True` (the default) will cause the packet to fire once its timestamp advances; data products tagged with `False` are still copied into the packet but will not on their own schedule a send. Note that the configured packet period still matters—the component evaluates on-change packets on those interval boundaries, but suppresses the transmission if no tracked data product reports a timestamp newer than the last emission.

The dedicated on-change unit tests under `src/components/product_packetizer/test_on_change` exercise this behavior in isolation. The YAML below matches the model compiled in those tests and is intentionally minimal so it can double as documentation for configuring the feature.

5.1 On-Change YAML Model

```

1  ---
2  description: This is an example set of packets.
3  # starting id...
4  # assuming 1 hz tick
5  packets:
6    - name: Packet_1 # must be unique, enforce by autocoder
7      description: Baseline periodic packet used by the on-change tests for
8        ↳ reference behavior.
9      id: 7
10     data_products:
11       - name: Test_Component_1_Instance.Data_Product_A
12         use_timestamp: False
13         include_timestamp: True
14         event_on_missing: True
15       - name: Test_Component_2_Instance.Data_Product_C
16         event_on_missing: False
17         use_timestamp: False
18         include_timestamp: False
19     period: "3" # create every 3 ticks
20     enabled: False
21   - name: Packet_6
22     description: Minimal packet dedicated to validating the On_Change feature.
23     id: 16
24     data_products:
25       - name: Test_Component_2_Instance.Data_Product_D
26         used_for_on_change: True # When True, this data product's timestamp
27           ↳ drives the on-change decision.
28       - name: Test_Component_1_Instance.Data_Product_B
29         used_for_on_change: False # When False, the data is included but does
30           ↳ not trigger on-change sends.
31     period: "1"
32     enabled: On_Change # Packets configured this way emit only when a tracked
33       ↳ data product timestamp advances.

```


5.2 Generated Output

Just like the periodic example, the generator produces an Ada structure that is supplied to the Product Packetizer instance. The output derived from the on-change YAML is shown below.

```
1  -- Standard includes:
2  with Product_Packet_Types; use Product_Packet_Types;
3  with Packet_Types;
4  with Sys_Time.Arithmetic;
5
6  -- This is an example set of packets.
7  package Test_Assembly_Product_Packets_Test_Packets is
8
9      -- Packet_1:
10     -- Baseline periodic packet used by the on-change tests for reference
11     ↪ behavior.
12
13     -- Packet_1 data product items:
14     -- Total packet buffer size: 160 bits
15     Packet_1_Items : aliased Packet_Items_Type := [
16         -- Item entry for Test_Component_1_Instance.Data_Product_A:
17         1 => (Data_Product_Id => 1, Use_Timestamp => False, Include_Timestamp =>
18             ↪ True, Event_On_Missing => True, Used_For_On_Change => True,
19             ↪ Packet_Period_Item => False, Size => 4),
20         -- Item entry for Test_Component_2_Instance.Data_Product_C:
21         2 => (Data_Product_Id => 3, Use_Timestamp => False, Include_Timestamp =>
22             ↪ False, Event_On_Missing => False, Used_For_On_Change => True,
23             ↪ Packet_Period_Item => False, Size => 10)
24     ];
25
26     -- Packet_1 packet description:
27     Packet_1_Description : Packet_Description_Type := (
28         Id => 7,
29         Items => Packet_1_Items'Access,
30         Period => 3,
31         Offset => 0,
32         Enabled => Product_Packet_Types.Disabled,
33         Use_Tick_Timestamp => False,
34         Count => Packet_Types.Sequence_Count_Mod_Type'First,
35         Send_Now => False,
36         Last_Emission_Time => Sys_Time.Arithmetic.Sys_Time_Zero
37     );
38
39     -- Packet_6:
40     -- Minimal packet dedicated to validating the On_Change feature.
41
42     -- Packet_6 data product items:
43     -- Total packet buffer size: 96 bits
44     Packet_6_Items : aliased Packet_Items_Type := [
45         -- Item entry for Test_Component_2_Instance.Data_Product_D:
46         1 => (Data_Product_Id => 4, Use_Timestamp => False, Include_Timestamp =>
47             ↪ False, Event_On_Missing => False, Used_For_On_Change => True,
48             ↪ Packet_Period_Item => False, Size => 2),
49         -- Item entry for Test_Component_1_Instance.Data_Product_B:
50         2 => (Data_Product_Id => 2, Use_Timestamp => False, Include_Timestamp =>
51             ↪ False, Event_On_Missing => False, Used_For_On_Change => False,
52             ↪ Packet_Period_Item => False, Size => 10)
53     ];
54
55     -- Packet_6 packet description:
56     Packet_6_Description : Packet_Description_Type := (
57         Id => 16,
```

```

49     Items => Packet_6_Items'Access,
50     Period => 1,
51     Offset => 0,
52     Enabled => Product_Packet_Types.On_Change,
53     Use_Tick_Timestamp => False,
54     Count => Packet_Types.Sequence_Count_Mod_Type'First,
55     Send_Now => False,
56     Last_Emission_Time => Sys_Time.Arithmetic.Sys_Time_Zero
57 );
58
59 -- List of packets for the packetizer to build:
60 Packet_List : aliased Packet_Description_List_Type := [
61     1 => Packet_1_Description,
62     2 => Packet_6_Description
63 ];
64
65 end Test_Assembly_Product_Packets_Test_Packets;

```

6 Special Items

The Product Packetizer allows you to specify “special” items to include in a packet that reflect internal data of the Product Packetizer component itself. Currently, the only supported “special” items are packet periods of the packets produced by the Product Packetizer. Packet 5, specified above, includes these items by specifying a data product within the Product Packetizer, ie. `Product_Packetizer_Instance.Packet_4_Period`. The Product Packetizer doesn’t actually have any data products, so this nomenclature instead denotes a special item. In this case, we want to include the current packet period value (a 4 byte unsigned integer) for Packet 4 into the packet. A period can be specified for any packet included in the YAML model using this pattern. Error checking at the modeling level will prevent you from specifying a packet period for a packet that does not exist.