# Parameter Table Generator
*Autocoder User Guide*

## 1   Description

The purpose of this generator is to provide a user friendly way of defining the layout of a parameter table that will be used within the Parameters component. The generator takes a YAML model file as input which specifies the component instance parameters to include in the table, in the order specified. From this information, the generator autocodes an Ada specification file which contains a data structure that should be passed to the Parameters component upon initialization. This data structure provides the necessary information to the Parameters component to properly decode a parameter table upload and push new parameter values to the correct downstream component where those parameters will be used.

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 parameter table.
3   type: map
4   mapping:
5     # Description of the parameter table.
6     description:
7       type: str
8       required: False
9     # The name of the component that this parameter table is being constructed
       ↪  for. This component
10    # must exist in the assembly specified by this file's name otherwise an error
       ↪  will be thrown.
11    # The component's name must be specified so that the generator can verify
       ↪  that any routed
12    # parameters in the table actually exist in the assembly.
13    parameters_instance_name:
14      type: str
15      required: True
16    # List of parameters to include in the suite.
17    parameters:
18      seq:
19        # Each parameter name must be in the format
         ↪  "Component_Instance_Name"."Parameter_Name" or
20        # simpy "Component_Instance_Name" to use all of the component's
         ↪  parameters.
```

```
21       - type: str
22         required: False
23     # A parameter table must have at least one parameter.
24     range:
25       min: 1
26     required: True
```

# 3 Example Input

The following is an example parameter tabl input yaml file. Model files must be named in the form *optional_name.assembly_name.parameter_table.yaml* where *optional_name* is the specific name of the parameter table and is only necessary if there is more than one Parameters component instance in an assembly. The *assembly_name* is the assembly which this table 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.

```
1    ---
2    # Optional - Description of parameter table
3    description: This is an example parameter table layout.
4    # Required - The name of the Parameters component instance that
5    # this table is being constructed for.
6    parameters_instance_name: Parameters_Instance
7    # Required - A list of parameters to include in the table in order
8    parameters:
9      # You may specify a parameter name in the form:
10     #   "component_Instance_Name"."Parameter_Name"
11     - component_A.Parameter_I32
12     - component_C.The_Tick
13     - component_A.Parameter_U16
14     # Or add ALL the parameters for a component on order
15     # by simply specifying the "Component_Instance_Name"
16     - component_B
```

As can be seen, specifying the parameter table layout simply consists of listing the parameters to include. You can either list just the component instance name, in which case all the parameters for that component will be included in the table in the order specified within the component's parameter model, or you can list individual component parameters, which provides fine grain control of which parameters are included and in what order.

# 4 Example Output

The example input shown in the previous section produces the following Ada output. The `parameter_Table_Entries` variable should be passed into the Parameters 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.

```
1    --------------------------------------------------------------------------------↩
     ↪ --------------
2    -- This file was autogenerated from /vagrant/adamant/src/components/parameters/↩
     ↪ test/test_assembly/test_parameter_table.test_assembly.parameter_table.yaml
     ↪ on 2022-04-01 19:37.
```

```ada
3    --
4    -- Copyright: The University of Colorado, Laboratory for Atmospheric and Space
     ↪   Physics (LASP)
5    --------------------------------------------------------------------------------⌋
     ↪   -------------
6
7    -- Standard includes:
8    with Parameters_Component_Types; use Parameters_Component_Types;
9
10   -- This is an example parameter table layout.
11   package Test_Parameter_Table is
12
13      -- The size of the parameter table (in bytes). This includes the size of the
        ↪   parameter
14      -- table header (Parameter_Table_Header.T) plus the size of all the parameter
        ↪   table data.
15      Parameter_Table_Size_In_Bytes : constant Natural := 30;
16
17      -- A list of the parameter table entries for use by the Parameters_Instance
        ↪   component.
18      Parameter_Table_Entries : aliased Parameter_Table_Entry_List := (
19         -- Parameter Component_A.Parameter_I32, size of 4 byte(s).
20         0 => (
21            Id => 2,
22            Component_Id => 1,
23            Start_Index => 0,
24            End_Index => 3
25         ),
26         -- Parameter Component_C.The_Tick, size of 12 byte(s).
27         1 => (
28            Id => 5,
29            Component_Id => 3,
30            Start_Index => 4,
31            End_Index => 15
32         ),
33         -- Parameter Component_A.Parameter_U16, size of 2 byte(s).
34         2 => (
35            Id => 1,
36            Component_Id => 1,
37            Start_Index => 16,
38            End_Index => 17
39         ),
40         -- Parameter Component_B.Parameter_U16, size of 2 byte(s).
41         3 => (
42            Id => 3,
43            Component_Id => 2,
44            Start_Index => 18,
45            End_Index => 19
46         ),
47         -- Parameter Component_B.Parameter_I32, size of 4 byte(s).
48         4 => (
49            Id => 4,
50            Component_Id => 2,
51            Start_Index => 20,
52            End_Index => 23
53         )
54      );
55
56   end Test_Parameter_Table;
```