cadence®

Part Table Editor User Guide

Product Version 23.1
September 2023

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Contents

Preface

PTF Editor is shipped as part of PCB Librarian, PCB Librarian XL, and the latest packaging configurations of Allegro Design Entry HDL.

This guide describes how to create and maintain physical part table files.

This guide assumes familiarity with a system text editor, HDL language concepts, and the following Cadence tools used to create component symbols and models:

- Library Explorer, which lets you manage component libraries
- Part Table Editor, which lets you create part table files
- Allegro Design Entry HDL, which lets you create logic designs by drawing schematics using symbols and functional blocks
- Packager-XL, which lets you prepare your schematic for PCB layout
- Allegro PCB Editor, which lets you create and manage physical layouts

Typographical Conventions

This list describes the syntax conventions used for tools used in the library development and management process. Where applicable, exceptions to these conventions are explicitly indicated.

`literal (LITERAL)`	Nonitalic or (UPPERCASE) words indicate key words that you must enter literally. These keywords represent command (function, routine) or option names.
`argument`	Words in italics indicate user-defined arguments for which you must substitute a value.
|	Vertical bars (OR-bars) separate possible choices for a single argument. They take precedence over any other character. For example, `command` *`argument`* | *`argument`*

[ ]	Brackets denote optional arguments. When used with OR-bars, they enclose a list of choices. You can choose one argument from the list.
{ }	Braces are used with OR-bars and enclose a list of choices. You must choose one argument from the list.
...	An ellipsis indicates that you can repeat the previous argument. If they are used with brackets, you can specify zero or more arguments. If they are used without brackets, you must specify at least one argument, but you can specify more. *argument*...: specify at least one argument, but more are possible [*argument*]...: you can specify zero or more arguments
,...	A comma followed by an ellipsis indicates that if you specify more than one argument, you must separate those arguments by commas.
Courier font	Text in Courier font indicates command-line examples.

Related Documentation

The following manuals give you information about other tools used during the library and part creation and management process:

If you want to know...	Read
How to manage digital component libraries	Library Explorer User Guide
How to create library parts	Part Developer User Guide
How to use Allegro Design Entry HDL to enter schematics	Allegro Design Entry HDL User Guide
More about Allegro Design Entry HDL digital libraries	Allegro Design Entry HDL Libraries Reference

If you want to know...	Read
More about how to create and use physical layouts	Allegro documentation
More about properties supported by Cadence PCB design software	PCB Systems Properties Reference

1

Part Table Files

The Physical Part Table (.ptf) file stores the packaging properties for a part in the library. This file contains part information such as package types, manufacturer names, part numbers, and any custom properties. Each physical part must have an entry in the .ptf file in order to be packaged properly.

For example, displayed below is a typical entry from a .ptf file:

```
FILE_TYPE = MULTI_PHYS_TABLE;
PART '74LVT574'
CLASS = IC

:PACK_TYPE = Part_NUMBER | JEDEC_TYPE | DESCRIPTION;
DIP = CDS123 | DIP20_3 | FLIP_FLOP
SOIC = CDS456 | SOIC20 | FLIP-FLOP
LCC = CDS789 | LCC20 | FLIP-FLOP

END_PART
END.
```

In the above part table file:

- A unique part number is assigned based on the package style.
- A package symbol name is assigned based on the package style.

  Note: The JEDEC_TYPE property may also be defined in the chips.prt file. However, the part_table view has the priority.

- A part description is added.

The PACK_TYPE property is a *key* property (it is on the left of the equal sign). This implies that every 74LVT574 part in the schematic must have the PACK_TYPE property assigned. However, if a 74LVT574 part that does not have a PACK_TYPE property value of DIP, SOIC,

or LCC is found, Packager fails. To set a default PACK_TYPE value, use the OPT statement as follows:

```
:PACK_TYPE (OPT = 'LCC') = PART_NUMBER | JEDEC_TYPE | DESCRIPTION;
```

When a 74LVT574 part in the schematic fits the key property description (has a PACK_TYPE property value of DIP, SOIC, or LCC), the *injected* properties (they are all on the right of the equal sign) are added to the packager netlist files (specifically the pstchip.dat file).

Each cell in a library containing logical parts should have a corresponding .ptf file. You can place all of these files in a single directory, which will be read later by Packager-XL during packaging. You should maintain packaging information, such as footprint (JEDEC_TYPE), VALUE, TOLERANCE, and PWR_RATING, in this file.

In some cases, you may wish to automatically generate .ptf files from an existing MRP (Material-Resource-Planning) system. Preferred parts and user part information could be extracted and used for .ptf file creation. This would ensure accurate and current information. The preferred parts are determined by a property called STATUS in the .ptf file. The values are PREF for preferred parts and NONPREF for the non-preferred ones. This lets you easily view and filter the selection of parts based on preferred parts when placing parts in the schematic. CheckPlus can then be used to flag any parts selected from the NONPREF entries.

Physical Part Table File Format

You can create a physical part table using any text editor or the Part Table Editor from Cadence. These files are kept in tabular format and can easily be read and updated. A physical part table file can contain information for one or more part types. A part table file begins with a line that identifies the type of file it is

FILE_TYPE = MULTI_PHYS_TABLE;

and ends with the keyword

END.

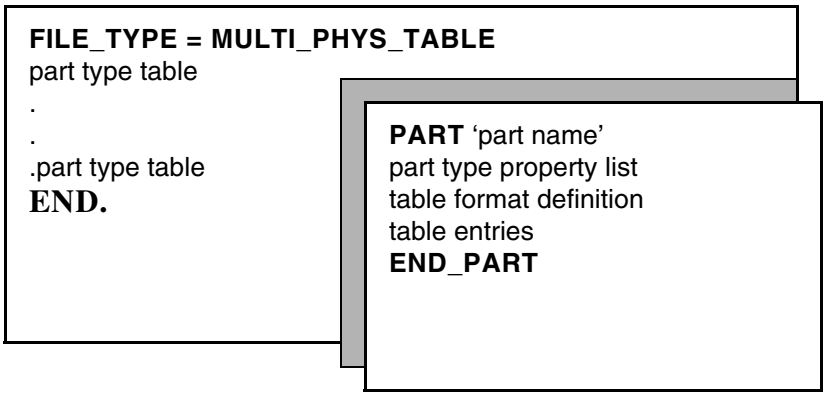
Between these two lines, you can include information for more than one part type. Each part type definition is a separate part type table. Each table begins with a line with the keyword PART followed by the name of the part type being redefined by the table entries and ends with the keyword END_PART (notice the absence of a period).

Physical Part Table File Syntax

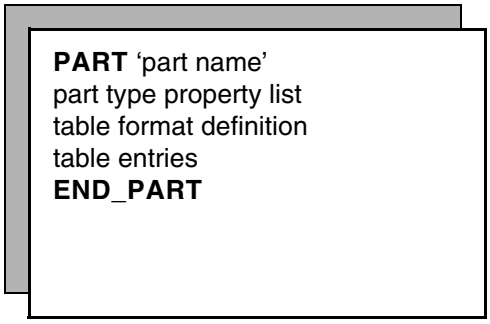
The physical part table file has the following general format:

```
FILE_TYPE = MULTI_PHYS_TABLE;
PART 'part_name'
[ part_type_ prop_list ]
table_format_definition
table_entry
END_PART
PART 'part_name'
     .
     .
     .
END_PART
END.
```

The figure below shows a generalized picture of a physical part table file along with the format of an individual part type table.



The subsections that follow provide detailed syntax information on the format of a part type table. Each line marked with a bullet in the part type table outline below corresponds to a subsection that follows.



PART 'part_name'

The physical part name of the component being redefined by the table entries. The part_name must be enclosed in single quotes.

Part_Type Property_List

You use this section of the part tables to add new properties to all instances of a part type without having to modify the physical information files or library drawings.

This is useful when you want to add properties independent of any set of properties attached to a logical part. The part_type_ property_list follows the format:

```
property_name = property_value
```

There can be any number of property name/value entries, but there can be only one entry per line.

property_name — Specify the name of the property. It is a string of no more than 16 alphanumeric characters, beginning with an alphabetic character. The underscore (_) is considered an alphanumeric character.

property_value — Specify the value of the property along with its unit. For example, if you have added `Height` as the property in `property_name`, then you will specify its value as `10mm`. You can also specify any other unit of measurement. Supported values are `micron`, `microns`, `millimeters`, `mm`, `centimeters`, `cm`, `in`, `inches`, `meter`, `um`, `pm`, `nm`, `ps`, `us`, `ms`, `min`, `sec`, and `hour`. These units are also passed to Allegro PCB Editor for processing. However, the units that are supported by Allegro PCB Editor are `microns`, `mils`, `inches`, `millimeter`, and `centimeter`.

If a property does not fit on one line, use a tilde (~) as a continuation character. The tilde can appear between any two characters but must be the last character in the line.

For example, this entry is read as if it were all on one line:

```
CLASS = DIS~

CRETE
```

Multiple spaces in a line are read as one space. Leading and trailing spaces around property values are removed. If leading or trailing spaces are required, surround the property values with single or double quotes:

```
CLASS = ' DISCRETE'
```

You can include a quotation mark in a quoted string by doubling it when used:

```
HOW_ARE_YOU = 'I''m OK’
```

Table Format Definition

This line defines the format of each table entry that follows.

The `table_ format_definition` uses the format:

```
: prop_name [(OPT=’def’ )] [separator  prop_name ...] = prop_name [separator
prop_name ...] ;
```

The left-hand side of the `table_format_definition` describes the key property names that are attached to an instance of the part on the schematic. These properties control the selection and customization of the part. More than one property can be specified and property definitions can span several lines.

`prop_name` — A standard SCALD property name

OPT	Defines whether a property is optional on an instance of a part.
'def'	The default value for the property if it is not present on an instance of the part. The default value must be enclosed in single quotes. If the default value is not included, Packager-XL uses the first entry in the table that matches the other key properties.

For example, the following definition specifies that the VALUE property is optional on the part:

```
: VALUE(OPT='1K') = PART_NUMBER;
```

If the VALUE property is not present on the part, the Packager assumes a default value of 1K and does not generate any warning messages.

If more than one property is specified, all properties must match the values as specified in the table before the part entry is selected.

The following figure shows a physical part table with two properties specified. For each part instance, both VALUE and TOLERANCE must match the specified values before the entry is selected. In this case, changing the TOLERANCE property on a 1K resistor selects a different part (with a corresponding change in cost).

```
File_Type = MULTI_PHYS_TABLE;
PART '1/4W RES'
: VALUE, TOLERANCE = PART_NUMBER COST;
1K, 5% = CB1025 $0.05
1k, 1% = CB1021 $0.50
1.2k, 5% = CB1225 $0.05
1.2K, 1% = CB1221 $0.50
END_PART
END
```

separator

If your instance property list or part property list contains more than one entry, you must choose a separator character. You indicate your choice of separator character simply by using an eligible character in your definition. Your character choice as a separator eliminates the use of that character in expressing a property value. You may use the same separator character in the instance and part property lists or define a different character for each list.

A separator may be any keyboard character (including a space or multiple spaces) that does not have a conflicting definition. It cannot be a letter, a digit, or any of the following special characters:

( )	Opening and closing parentheses, which delimit attributes
{ }	Opening and closing braces, which delimit comments
[ ]	Opening and closing square brackets, which enclose a range for an R attribute. These characters are ineligible only when the R attribute is used.
=	Equal sign, which is an assignment character
:	Colon, which introduces the table format definition
;	Semicolon, which is a statement terminator
‘ “	Quote marks (single or double), which indicate that spaces should be interpreted literally
_	Underscore, which is interpreted just as a letter or number
~	Tilde, which is a continuation character

The characters you define as separators in this format line are the same characters you must use as separators for each table entry that follows.

The following example uses a comma to separate the property names VALUE and TOLERANCE.

Example

```
:VALUE, TOLERANCE = PART_NUMBER, COST;
```

The second half of the `table_ format_definition` describes a list of the properties for the Packager to associate with the new part type. The `prop_name` and `separator` are the same as those defined for the first half of the definition. The separator in this section does not have to be the same one used previously; any of the legal separator characters are allowed. There is no limit to the number of properties that can be specified.

Part Table Entries

The table_entry section of the part table contains the actual physical part table entries that Packager-XL searches to determine the new part types to create.

Each table entry has the following format:

```
instance_val = part_type_val [(name_spec)] [:added_prop ~
```

```
= 'added_val']
```

`instance_val`	The value or values of each instance of the property whose names are defined on the left side of `table_format_definition` must match before the entry is selected. There must be the same number of `instance_val` entries as there are property names on the left side of `table_ format_definition`.
`part_type_val`	The values to attach to the definition subtype in the `chips_prt` file of Packager-XL if this table entry is selected. There must be the same number of part_type_val entries as there are entries on the right side of `table_ format_definition`.
`name_spec`	The new part subtype name specification for this table entry. This is an optional item in the table entry. `name_spec` can follow these three forms: ■ ! ■ subtype_name_suffix ■ ~complete_user_subtype_name See Appendix 1, "Part Subtype Names" for an explanation of the values `name_spec` can take.
`added_ prop`	A list of properties that are added to this part. This allows you to add properties to specific parts without having to redefine the table format for all parts. Each `added_ prop` must be a standard SCALD property name. Commas must separate multiple properties.
`'added_val'`	The property value to match `added_ prop`. The value must be enclosed in single quotes.

`added_ prop` entries are only used when you need to add new properties for the part type created for this table entry.

The figure below shows a physical part table that defines a new property value. Part types created for resistors with a VALUE of 1K, a PART_NUMBER of CB1025, and a COST of $0.05

will also have a TOLERANCE of 5%. Resistors with a VALUE of 1.2K or 1.5K will not have the TOLERANCE property added to the new part type.

```
FILE_TYPE = MULTI_PHYS_TABLE;
PART '1/4W RES'
:VALUE = PART_NUMBER, COST;
1K = CB1025, $0.05: TOLERANCE = '=5%'
1.2K = CB1225, $0.05
1.5K = CB1525, $0.05
END_PART
END.
```

Adding Mechanical Parts to the chips.prt File

```
FILE_TYPE = LIBRARY_PARTS;
primitive 'HEATSINK';
    body
        NC_PINS = '(1)';
        PART_NUMBER = 'HEATSINK';
        CLASS = 'MECHANICAL';
    end_body;
end_primitive;
END.
```

Part Subtype Names

Parts defined in PPTs are assigned new part subtype names. You can control the subtype name with the `name_spec` item in each table entry and with the PACK_TYPE property value on instances of the part. These subtype names do not affect how Packager-XL selects `chips.prt` file entries. Refer to the discussion of PACK_TYPE in Chapter 2 of the Packager-XL Reference Manual.

The figure below illustrates some of the ways to define the part subtype name.

Figure 1-1

```
 VALUE,  TOLERANCE    =   PART_NUMBER, COST   Resulting Name
 1K      2% (1K)      =   1285    $.50        RESISTOR-1K
 2.3K    1% (2.3K)    =   1300    $.50        RESISTOR-2.3K
 1K      5% (1K, 5%)  =   1024    $.24        RESISTOR-1K,5
 5K      1% (!)       =   1000    $.43        RESISTOR-5K,1%
 1K      3% (!)       =   1028    $.24        RESISTOR-1K,3%
 1K      4% (!)       =   1028    $.24        RESISTOR-1K,4%
 10K     5% (~R10K)   =   1029    $.24        R10K
```

You can create subtype names in several ways:

- You can specify the suffix to append to the parent part type name.
- You can tell Packager-XL to use the instance property values as the suffix. This is the default behavior.
- You can specify the entire subtype name.

Automatic Subtype Names

If you leave name_spec out of the table entry, Packager-XL uses the instance property values as the suffix.

User-Defined Suffixes

If you put a string of text between the parentheses in the name_spec, Packager-XL puts a dash in front of the string and appends it to the parent part type name. Lines 2-4 of Figure 1-1 on page 16 use this method.

Instance Property Value Suffixes

If you use an exclamation point (!) as the name_spec, Packager-XL constructs a suffix for the parent part type from the value(s) of the instance property or properties (the property values that tell Packager-XL which table entry to use for each instance). Lines 5 and 6 of Figure 1-1 on page 16 use this method.

Thus, for all three of the suffixes above, the part subtype names follow the form

```
parent_partname[ _PACK_TYPE]-suffix
```

Characters Allowed in a Suffix

The characters that are allowed in a subtype suffix are all letters and digits and the following special characters:

```
,   $   %   #   &   *   +   _   .
```

The PART_TYPE_LENGTH directive controls the subtype name length limit. The length of the part type and the suffix together cannot exceed the value of the PART_TYPE_LENGTH option, which has a maximum value of 255. If the names are longer than this limit, these names are truncated.

If no suffix is specified, Packager-XL uses the instance property values as the suffix.

Complete User Subtype Names

If you use a tilde (~) followed by a string of text, the packager uses that name as the name of the new part. The name is arbitrary and need not contain or even resemble the parent part type name. Line 8 of Figure 1-1 on page 16 uses this method.

Names from the PACK_TYPE Property Value

Suppose your organization screens LM741 op amps by hand to select low-noise parts and that you use LM741 parts in several different packages. You might attach a property LONOISE=TRUE in the schematic to the instances of the part that must be the special low-noise versions. Your part table file will be as follows:

```
FILE_TYPE = MULTI_PHYS_TABLE;
PART 'LM741'
: LOWNOISE
TRUE (LONOISE) = $1.50
FALSE (NOISY) = $0.75
END_PART
END.
```

The packager would produce part subtype names such as:

- LM741_DIP-LONOISE
- LM741_DIP-NOISY
- LM741_SMD-LONOISE

Here is another example. Consider the part table displayed below:

```
FILE_TYPE = MULTI_PHYS_TABLE;
PART '74LS00'
:A,B,C = PART_NUMBER, COST;
1, 2, 3, (~MODIFIED 74LS00) = 001, 30
4, 5, 6, (~74LS00NEW) = 002, 40
END_PART
END.
```

If there is an instance of a 74LS00 part with values of 1, 2, and 3 for the properties A, B, and C, respectively, Packager-XL creates the subtype name MODIFIED_74LS00 and gives the part the properties PART_NUMBER=001 and COST=30, with the pinouts of the 74LS00 entry in the `chips.prt` file.

If there is an instance of the 74LS00 part with values of 4, 5, and 6 for A, B, and C, and the properties PACK_TYPE=DIP and PART_NAME=74LS00, Packager-XL creates a subtype with the name 74LS00NEW with properties PART_NUMBER=002 and COST=40. This subtype receives the pinouts of the 74LS00_DIP entry in the `chips.prt` file or the 74LS00 entry if there is no 74LS00_DIP in the `chips.prt` file.

It is possible to produce name conflicts when using this syntax in part tables. Suppose that the previous example was changed as follows:

Subtype Name Conflict

```
PART '74LS00'
:A,B,C = PART_NUMBER, COST;
1, 2, 3, (~MODIFIED 74LS00) = 001, 30
4, 5, 6, (~74LS00NEW) = 002, 40
7, 8, 9, (~74LS00NEW) = 003, 50
```

If your design contains a 74LS00 part with the values 4, 5, and 6 for properties A, B, and C, and another 74LS00 part with values 7, 8, and 9 for A, B, and C, Packager-XL issues an error message stating that the last two subtype names are in conflict.

When you use the PACK_TYPE property as one of the instance properties in a table entry, and also use the tilde (~) to specify a complete user subtype name, your subtype name is applied only to the package subtypes of that value of PACK_TYPE.

In the example below, the first entry is only applied to parts of package subtype 74LS00_SO (that is, parts with the property PACK_TYPE=S0). The second entry is only applied to parts of the subtype 74LS00_DIP.

Creating Subtype Names When PACK_TYPE is in the Table Definition

```
PART '74LS00'
:A,B,C = PART_NUMBER, COST;
1, 2, 3, S0 (~MODIFIED 74LS00) = 001, 30
4, 5, 6, DIP (~74LS00NEW) = 002, 40
```

If PACK_TYPE is declared as optional in the part table and has a default value, the entry corresponding to the default value is also applied to the base (parent) part type when it carries no PACK_TYPE property:

Creating Subtype Names When PACK_TYPE is in the Table Definition

```
PART '74LS00'
:A,B,C PACK_TYPE (OPT = 'DIP') = PART_NUMBER, COST;
1, 2, 3, S0 (~MODIFIED 74LS00) = 001, 30
4, 5, 6, DIP (~74LS00NEW) = 002, 40
```

In the above figure, the first entry is applied to instances of the 74LS00 part with PACK_TYPE=SO. The second entry is applied to instances with no PACK_TYPE property, and to instances with PACK_TYPE=DIP.

If the `chips.prt` file entries for 74LS00 and the version with PACK_TYPE=DIP are different, the packager will report an error if the design contains 74LS00s both with and without the PACK_TYPE=DIP property. The error occurs because both parts map to the same subtype name, 74LS00-NEW but have different `chips.prt` file entries (different pinouts).

Sample Physical Part Table

The table below shows an example of a physical part table for 1/4-watt resistors. Comments are enclosed in braces and precede the element they describe. The line numbers on the left are not actually part of the file. They are used to describe the format of the table.

Line 1	Starts the physical part table file and tells Packager-XL that the file is a multiple physical part table file. It can contain more than one part type.
Lines 2 through 4	Separates the first line from the lines that follow. Packager-XL ignores blank lines and comment lines, but these lines make the file more easy to read. Comments are enclosed by curly braces ({ }). Comments can cross line boundaries, but they cannot be nested.
Line 5	Starts the physical part table entries for the 1/4W RES part type. The part type name must be enclosed in single quotation marks. Either single or double quotation marks are required if the part name includes spaces.
Lines 9 and 10	Indicate that all 1/4-watt resistors have the body properties CLASS and JEDEC_TYPE added to the part type, with the values DISCRETE and CR1/4W, respectively.
Line 14	Describes the format for each line in the table for the 1/4-watt resistor. In this example, the property that can be used to modify the resistor is VALUE. The properties added to the new part types are PART_NUMBER and COST. The comma that separates the PART_NUMBER and COST properties in this line defines the separator character between values within the table. As such, you cannot use a comma to express a property value within this part type table.
Lines 18 to 28	The actual physical part table entries that Packager-XL uses to determine the new part types to be created. For example, line 18 specifies that all 1/4-watt resistors with a `VALUE` property of 1K are assigned to a new part type. This new part type has the same definition as a 1/4-watt resistor without a `VALUE` property plus the additional properties `PART_NUMBER` and `COST` with the values of CB1025 and $0.05, respectively. If the 1/4-watt resistor had a VALUE of 4.7K, the added PART_NUMBER and COST would be CB4725 and $0.05.
Line 32	The end of the part table for the part type 1/4W RES. There can be additional part type definitions for other part types within the same physical part table.
Line 36	The end of the file.

Sample Physical Part Table

```
1   FILE_TYPE = MULTI_PHYS_TABLE;
2
3   ( 1/4-watt resistor table )
4
5   PART '1/4 W RES'
6
7   ( part_type_prop_list (ALLEGRO specific props )
8
9   CLASS = DISCRETE
10  JEDEC_TYPE = CR1/4W
11
12  ( table_format_definition )
13
14  : VALUE = PART_NUMBER, COST;
15
16  (table_entry section - resistors )
17
18  1K = CB1025, $0.05
19  1.2K = CB1225, $0.05
20  1.5K = CB1525, $0.05
21  2.2K = CB2225, $0.05
22  2.7K = CB2725, $0.05
23  3.3K = CB3325, $0.05
24  3.9K = CB3925, $0.05
25  4.7K = CB4725, $0.05
26  5.6K = CB5625, $0.05
27  6.8K = CB6825, $0.05
28  8.2K = CB8225, $0.05
29
30 ( end of the 1/4 W RES entries)
31
32 END_PART
33
34 ( end of the physical part table file )
35
36  END.
```

2

Working with Part Table Editor

Part table files enable you to create new parts from a basic part type and attach new body properties to a part. Cadence provides a part table file editor called Part Table Editor, which enables you to create part table files. You can use the Part Table Editor to:

- Create a part table
- Deleting a part table
- Verify the part table
- Add new parts to the part table

The Part Table Editor lets you copy header rows and part rows. If you copy a part row, the Part Table Editor also copies the corresponding header row. You can copy items from both the tree area and the grid area.

Launching the Part Table Editor

The Part Table Editor can be launched from within Library Explorer and Part Developer. You can also launch the tool from the hierarchy.

Launching the Part Table Editor from Library Explorer

To launch the Part Table Editor from Library Explorer:

1. Choose *Tools – Part Table Editor*.

   This launches the Part Table Editor.

Launching the Part Table Editor from Part Developer

To launch the Part Table Editor from Part Developer:

1. Choose *Tools – Part Table Editor*.

This launches the Part Table Editor.

Creating a New Part Table File

You can create a new library-level part table file by using the library-level Part Table Editor.

1. Choose *File – New – New File*.

   The new part table file appears. You can begin creating parts. See Adding New Parts to a Part Table File on page 24 for details.

Note: The Part Table Editor opens a new file automatically if you invoke it in the standalone mode.

Opening a Part Table File

1. Choose *File – Open*.

   The Open dialog box appears.
2. Select the `.ptf` file you want to open.

Deleting Parts in a Part Table File

1. Select the part to be deleted.
2. Choose *Edit – Delete*.

Adding New Parts to a Part Table File

1. Select the `.ptf` file.
2. Choose *File – New Part*.

   The new part appears in the tree view under the part table file.
3. Enter a new name for the part.

   This name should be unique. The Part Table Editor automatically checks for the existence of other parts in the part table with the same name. After you specify the name, you need to enter the header and part-row information.

Specifying Header Row Information

1. Select Header.

   The header section of the part table appears on the right hand side.
2. In the *Key Properties* group box, enter the name.
3. Specify whether the property is optional or not.

   If you specify the property as optional, the default value is displayed in the value field.
4. Enter the optional value in the *Value* field.

   If the property is not optional, enter the default value in the value field.
5. Specify whether the value has to be added as a physical property.
6. Click *Add* to add a property row in the group box.
7. Click *Delete* to delete the selected row.
8. In the *Injected Properties* group box, enter values for *Name* and *Value*.
9. Click *Add* to add a row in the group box.
10. Click *Delete* to delete a header row.
11. In the *Global Properties* group box, enter values for *Name* and *Value*.
12. Click *Add* to add a row in the group box.
13. Click *Delete* to delete a header row.

Note: All values in the header row are used to populate the part rows by default.

Specifying Part Row Information

You specify information for all part rows if you select *Part Row* in the tree area. If you select the part row name, you can specify information for that specific part row.

You can copy other part rows to create a new part row. When you copy a part row, the Part Table Editor also copies the corresponding header row.

1. Select Part Rows in the tree area.
2. Click *Add* to add a row in the group box.
3. Enter all values for properties in this row.

4. Repeat steps 2 and 3 until all rows are added.
5. Click *Delete* to delete selected part rows.
6. Select a row and enter a value for *Row Name*.
7. Specify whether the name you specify should be a suffix or an absolute name.

   Note: There are three naming conventions for a row:

   Default naming style.
   The Part Table Editor creates a name automatically. The format is <part name>_<pack_type>-<key property value 1>, <key property value 2>, ...

   Absolute name
   The name you specify is taken as the row name.

   Suffix name
   The name you specify is appended as a suffix.
   <part name>_<pack_type>-<suffix-name-provided>

   In the *Added Properties* group box, enter values for *Name* and *Value*.
8. Repeat steps 5 & 6 for all the part rows.
9. Click *Add* to add a property row in the group box.
10. Click *Delete* to delete a selected row.

Note: After creating a part table, you can save it using *File – Save***.**

Copying Existing Parts in a Part Table File

1. Select the part to be copied.
2. Choose *Edit – Copy*.
3. Choose *Edit – Paste*.

   The Part Table Editor pastes the copy of the part and names it `Copy of <name of the original part>`.

   Note: For the paste operation to work successfully, ensure that the part list pane is the active pane, and one of the parts is selected. The Paste operation is not successful, if between step 2 and step 3, you select the `.ptf` file or click on the right pane of the Ptf Editor window.

Verifying Part Tables

The Part Table Editor lets you verify the part table you create or modify. The Part Table Editor automatically verifies the part table file when you save it.

To verify your part table file:

➤ Choose *File – Checks*.

The Part Table Editor runs the checks and reports the errors.

Possible errors could be:

- No key properties defined for a part row
- Duplicate injected properties specified for a part row
- Duplicate global properties found for a part
- Duplicate header properties found
- No injected properties found in a header
- Duplicate part row names found

Index

Symbols

B

C

I

P

V