cadence®

Virtuoso EDIF 200 Reader and Writer SKILL Reference

Product Version IC23.1
June 2023

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

Contents

1

Customizing EDIF 200 In

You can customize EDIF 200 In output files by creating a SKILL file. A SKILL file contains SKILL procedures that specify changes to object properties in the source design.

This section lists the SKILL procedures you can add to a SKILL file and provides an example of each procedure.

You use the *User-Defined SKILL File* field on the EDIF 200 In form to specify the name of the SKILL file. Specify the full path if it is not in your run directory.

When you start EDIF 200 In, the software automatically looks for the SKILL file in the run directory or in the directory you specify in the *User-Defined SKILL File* field. When the SKILL file is found, the software runs the procedures and modifies the property information that is written to the EDIF 200 In output files.

A SKILL procedure returns a value of `t` when it runs successfully and a value of `nil` when it fails. When a procedure fails, EDIF 200 In processes the property, if possible, as the property is described in the source EDIF file.

See the Cadence SKILL Language User Guide and the Cadence SKILL Language Reference for information about writing SKILL procedures.

Licensing Requirements

For information on licensing in the Virtuoso Studio Design Environment, see *Virtuoso Software Licensing and Configuration Guide*.

edifinDisplay

```
edifinDisplay(
     t_edifFormName
     )
     => t / nil
```

Description

Displays the EDIF 200 In form.

Arguments

t_edifFormName	Specifies the Edif200 Import form name.

Value Returned

`t`	Indicates that the command succeeded.
`nil`	Indicates that the command failed.

Example

```
edifinDisplay(transEdifInForm)
```

edifLayerNumMap

```
edifLayerNumMap(
     t_figureGroup
     )
     => list( t_layerName t_purposeName ) / nil
```

Description

Maps the name of `figureGroup` specified in the input EDIF file to the layer-purpose pairs in the technology file.

The technology file is used when libraries are created by EDIF 200 In. The layers defined by `edifLayerNumMap` overwrite the default layers. The procedure returns a list that contains two strings that specify the layer and purpose names.

Arguments

t_figureGroup	A `figureGroup` name in the EDIF input file.

Value Returned

t_layerName	Returns *layerName*, which maps to the specified `figureGroup`.
t_purposeName	Returns *purposeName*, which maps to the specified `figureGroup`.
`nil`	Returns `nil` if mapping is not possible.

Skeletal Example

```
procedure( edifLayerNumMap( figureGroup "t" )
     prog((flag)
          flag = case( figureGroup
               ("arg1" return( list( "arg2" "arg3" )))
          ) ;end case
          if(!flag then
              return( list( "arg2" "arg3" ))
          );end if
     );prog
);procedure
```

`"arg1"`	Specifies the `figureGroup` name in the input EDIF file.

`"arg2"`	Specifies the DFII layer to which the `figureGroup` is mapped.
`"arg3"`	Specifies the DFII layer-purpose to which the `figureGroup` is mapped.

Example

```
procedure( edifLayerNumMap(figureGroup "t")
     prog((flag)
          flag = case( figureGroup
               ("DEVICE_BODY_FGP" return( list( "device" "drawing" )))
               ("NET_FGP" return( list( "wire" "drawing" )))
               ("PIN_FGP" return( list( "pin" "drawing" )))
               ("PIN_BODY_FGP" return( list( "pin" "drawing" )))
          );end case
          if(!flag then
               return( list( "device" "drawing" ))
          );end if
     );prog
);procedure
```

In this example, any `figureGroup` names that are not among those listed in the `case` statement are mapped to the layer `"device"` with the `"drawing"` purpose. The default layer name for `port` is `"pin"`. The default layer name for `net` is `"wire"`. The default layer name for `net label` is `"wire"` and the `layer-purpose` is `"label"`.

edifinMakeRenameString

```
edifinMakeRenameString(
     t_inputString
     )
     => l_outputString / nil
```

Description

Creates a special string when the string in the EDIF 200 In input file is an illegal name.

The function takes the illegal name string from the EDIF file as an argument and returns a list value that contains the new rename string. You can use `nil` as the `returnString` value. In this case, a `nil` value is returned, which indicates that no new string was provided and the original characters are deleted.

Arguments

t_inputString	An illegal name in the EDIF input file.

Value Returned

l_outputString	Indicates the list that contains the new rename string.
`nil`	Indicates no new string was provided.

Skeletal Example

```
procedure( edifinMakeRenameString( inputString "t" )
     prog( (returnString)
          case( inputString
               ( "arg1" returnString = "arg2" ))
          return( list( returnString ))
     );prog
);procedure
```

`"arg1"`	Specifies the string in the EDIF file to map.
`"arg2"`	Specifies the translated string in the DFII file.

To map names to a DFII database, you can use any alphanumeric character and any of the following special characters:

_ (underscore)	+ (plus)	- (hyphen)	: (colon)
<> (angle brackets)	{} (braces)	[] (square brackets)	() (parentheses)

If you use an invalid character, EDIF 200 In ignores the name and uses the current EDIF name.

Note: You can use this mapping convention to map the `instanceName`, `portName`, and `netName` constructs. However, when you rename instance names, the `charMapForInstName` function overrides this `edifinMakeRenameString` function.

Example

```
procedure( edifinMakeRenameString( inputString )
     prog( (returnString)
               case( inputString
                    ( "C4" returnString = "C<4>" )
               ); case changes name "C4" into "C<4>"
               return( list( returnString))
     );prog
);procedure
```

The `inputString` is the name of the string in the EDIF file to map. The `returnString` is the name of the DFII database name that is created.

ediFinishStatus

```
ediFinishStatus(
     x_returnCode
     )
```

Description

Provides an exit status number when EDIF 200 In is done.

Arguments

x_returnCode

- 0—means there were no errors and no warnings.
- -1—means there were errors and warnings.
- 1—means there were warnings but no errors.

Example

```
procedure( ediFinishStatus( returnCode "x")
returnCode = 0 if there are no errors and no warnings
returnCode = -1 if there are errors and warnings
returnCode = 1 is there are warnings and no errors
)
```