PyAEDT EDB-API cheat sheet

Version: 0.6.78 (stable)

/ Launch EDB-API using PyAEDT

EDB manager manages the AEDB database. An AEDB database is a folder that contains the database representing any part of a PCB. It can be opened and edited using the `Edb` class.

```
# Launch an instance
import pyaedt
edb = pyaedt.Edb(edbversion="2023.1", edbpath=
    aedb_path)
edb.save_edb() # Save the edb file
edb.close_edb() # exits the edb file
```

/ Stackup and Layers

These classes are the containers of the layer and stackup manager of the EDB API.

```
# Add a stackup layer
edb.stakup.add_layer("Name")
# Get the names of the all layers
edb.stackup.stackup_layers.keys()
```

/ Modeler and primitives

These classes are the containers of primitives and all relative methods. Primitives are planes, lines, rectangles, and circles.

```
# Create a polygon by defining points
points = [[0.0, 1e-3], [0.0, 10e-3], [100e-3, 10e-3],
    [100e-3, 1e-3], [0.0, 1e-3]]
edb.modeler.create_polygon_from_points(points,
    layer_name = "Name")
```

/ Components

The `components` class contains API references for net management. The main component object is called directly from the main application using property components.

```
# Get the list of components
edb.components.components.keys()
# Get the net information of a component
edb.components.get_component_net_connection_info("Q13N
    ")
```

/ Nets

The `nets` class contains API references for net management. The main net object is called directly from the main application using the property nets.

```
# List all nets available in AEDB file
edb.nets.netlist
# Delete a net
edb.nets["net_name"].delete()
```

/ Vias and padstacks

These containers are the API references for padstack management. The main padstack object is called directly from the main application using the property padstacks.

```
# Create a via
edb.padstacks.place(position = [5e-3, 5e-3], "MyVia")
# Get pad parameters
edb.padstacks.get_pad_parameters()
```

/ Sources and excitation

These classes are the containers of sources methods of the EDB for both HFSS and Siwave.

```
# Get the dictionary of EDB excitations
edb.excitations
# Create a differential port
edb.hfss.create_differential_wave_port(
    positive_primitive_id = trace_p[0].id,
    positive_points_on_edge = p1_points,
    negative_primitive_id = trace_n[0].id,
    negative_points_on_edge = n1_points, name = "
    wave_port_1")
```

/ Simulation setup

These classes are the containers of `setup` classes in EDB for both HFSS and Siwave.

```
# Set up HFSS simulation
setup = edb.create_hfss_setup(name = "my_setup")
setup.set_solution_single_frequency()
setup.hfss_solver_settings.enhanced_low_freq_accuracy
    = True
setup.hfss_solver_settings.order_basis = "first"
setup.adaptive_settings.add_adaptive_frequency_data("5
    GHz",8,"0.01")
```

```
# Set up SiWave simulation
setup = edb.siwave.add_siwave_dc_analysis(name = "
    myDCIR_4")
setup.use_dc_custom_settings = True
setup.dc_slider_position = 0
setup.add_source_terminal_to_ground("V1", 1)
solve_edb = edb.solve_siwave()
```

/ Simulation configuration

These classes are the containers of simulation configuration constructors for the EDB.

```
# Specify AC settings
sim_setup.ac_settings.start_freq = "100Hz"
sim_setup.ac_settings.stop_freq = "6GHz"
sim_setup.ac_settings.step_freq = "10MHz"

# Run batch solve
sim_setup=edbapp.new_simulation_configuration()
sim_setup.solver_type sim_setup.SOLVER_TYPE.SiwaveSYZ
sim_setup.batch_solve_settings.
    cutout_subdesign_expansion = 0.01
sim_setup.batch_solve_settings.do_cutout_subdesign =
    True
sim_setup.use_default_cutout = False
sim_setup.batch_solve_settings.signal_nets =
    signal_net_list
sim_setup.batch_solve_settings.components =
    component_list
sim_setup.batch_solve_settings.power_nets =
    power_nets_list

# Save configuration file
sim_setup.export_json(os.path.join(project_path, "
    configuration.json"))
edbapp.build_simulation_project(sim_setup)
```

/ SiWave Manager

Siwave is a specialized tool for power integrity, signal integrity, and EMI analysis of IC packages and PCBs. This tool solves power delivery systems and high-speed channels in electronic devices. It can be accessed from PyAEDT in Windows only. All setups can be implemented through the EDB API.

```
from pyaedt.siwave import Siwave
# this call returns the Edb class initialized on 2023
    R1
siwave = Siwave(specified_version="2023.1")
siwave.open_project("pyproject.siw")
siwave.export_element_data("mydata.txt")
siwave.close_project()
```

References from PyAEDT documentation

- EDB API