© 2025 ANSYS, Inc. or affiliated companies
Unauthorized use, distribution, or duplication prohibited.

ansys-aedt-toolkits-antenna

ANSYS, Inc.
Southpointe
2600 Ansys Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

May 22, 2025

ANSYS, Inc. and ANSYS Europe, Ltd. are UL registered ISO 9001:2015 companies.

CONTENTS

Useful links: *Installation* | Source repository | Issues

The AEDT Antenna Toolkit is a Python interface for accelerating antenna design using Ansys Electronics Desktop (AEDT). You can launch this toolkit from AEDT or launch it directly from a Python console.

Getting started Learn more about the AEDT Antenna Toolkit and how to install it.

Getting started Antenna wizard Understand how to use the Antenna toolkit wizard.

Antenna wizard API reference Understand the APIs available for the AEDT Antenna Toolkit.

API reference Examples Explore examples that show how to use the API.

Examples Contribute Learn how to contribute to the AEDT Antenna Toolkit codebase or documentation.

Contribute

CHAPTER

ONE

GETTING STARTED

This section explains how to install the AEDT Antenna Toolkit.

Installation Learn how to install the AEDT Antenna Toolkit.

Installation User guide Learn more about the Antenna wizard and how to use it.

User guide

1.1 Installation

1.1.1 Download links

The following installers are available for different operating systems:

Table 1: Available Installers

Installer Link	Operating System
Download	Windows
Download	Ubuntu 22.04
Download	Ubuntu 24.04

Visit the Releases page and pull down the latest installer.

1.1.2 Installing the `Antenna Toolkit`

Windows

First step is installing the `Antenna Toolkit`. In order to do so, follow the next steps.

1. Download the necessary installer from the latest available release. The file should be named `Antenna-Toolkit-Installer.exe`.
2. Execute the installer.
3. Search for the `Antenna Toolkit` and run it.

The `Antenna Toolkit` window should appear at this stage.

Linux

Ubuntu

Prerequisites:

1. **OS** supported for **Ubuntu(24.04 and 22.04)**.

2. Update `apt-get` repository and install the following packages with **sudo** privileges: **wget, gnome, libffi-dev, libssl-dev, libsqlite3-dev, libxcb-xinerama0 and build-essential** packages with **sudo** privileges

```
sudo apt-get update -y
sudo apt-get install wget gnome libffi-dev libssl-dev libsqlite3-dev libxcb-
↪xinerama0 build-essential -y
```

3. Install **zlib** package

```
wget https://zlib.net/current/zlib.tar.gz
tar xvzf zlib.tar.gz
cd zlib-*
make clean
./configure
make
sudo make install
```

To install the `Antenna Toolkit`, follow below steps.

1. Download the necessary installer from the latest available release. The file should be named `Antenna-Toolkit-Installer-ubuntu_*.zip`.
2. Execute the below command on the terminal

```
unzip Antenna-Toolkit-Installer-ubuntu_*.zip
./installer.sh
```

3. Search for the `Antenna Toolkit` and run it.

The `Antenna Toolkit` window should appear at this stage.

To uninstall the `Antenna Toolkit`, follow below steps.

1. Go to File menu. Click Uninstall option.
2. Click `Uninstall` button.

1.1.3 Python installation

The Antenna Toolkit can be installed like any other open source package. From PyPI, you can either install both the backend and user interface (UI) methods or install only the backend methods. To install both the backend and UI methods, run this command:

```
pip install ansys-aedt-toolkits-antenna[all]
```

If you only need the common API, install only the backend methods with this command:

```
pip install ansys-aedt-toolkits-antenna
```

1.1.4 For developers

You can be up and running with four lines of code:

```
git clone https://github.com/ansys/pyaedt-toolkits-antenna
cd pyaedt-toolkits-radar
pip install -e .
```

Now you can run it with:

```
run_toolkit
```

Details

Installing Pytools installer in developer mode allows you to modify the source and enhance it.

Before contributing to the project, please refer to the PyAnsys Developers guide. You need to follow these steps:

1. Start by cloning this repository:

   ```
   git clone https://github.com/ansys/pyaedt-toolkits-antenna
   ```

2. Create a fresh-clean Python environment and activate it. Refer to the official venv documentation if you require further information:

   ```
   # Create a virtual environment
   python -m venv .venv
   # Activate it in a POSIX system
   source .venv/bin/activate
   # Activate it in Windows CMD environment
   .venv\Scripts\activate.bat
   # Activate it in Windows Powershell
   .venv\Scripts\Activate.ps1
   ```

3. Install the project in editable mode:

   ```
   python -m pip install -e .[tests,doc]
   ```

4. Finally, verify your development installation by running:

   ```
   pytest tests -v
   ```

Style and testing

This project uses pre-commit. Install with:

```
pip install pre-commit
run pre-commit install
```

This now runs `pre-commit` for each commit to ensure you follow project style guidelines. For example:

```
git commit -am 'fix style'
isort....................................................................Passed
black....................................................................Passed
blacken-docs.............................................................Passed
flake8...................................................................Passed
codespell................................................................Passed
pydocstyle...............................................................Passed
check for merge conflicts................................................Passed
debug statements (python)................................................Passed
check yaml...............................................................Passed
trim trailing whitespace.................................................Passed
Validate GitHub Workflows................................................Passed
```

If you need to run it again on all files and not just staged files, run:

```
run pre-commit run --all-files
```

Local build

This application can be deployed as a frozen application using pyinstaller with:

```
pip install -e .[freeze]
run pyinstaller frozen.spec
```

This generates application files at `dist/ansys_python_manager` and you can run it locally by executing `Ansys Python Manager.exe`.

Documentation

For building documentation, you can either run the usual rules provided in the Sphinx Makefile:

```
pip install -e .[doc]
doc/make.bat html
# subsequently open the documentation with (under Linux):
<your_browser_name> doc/html/index.html
```

1.2 User guide

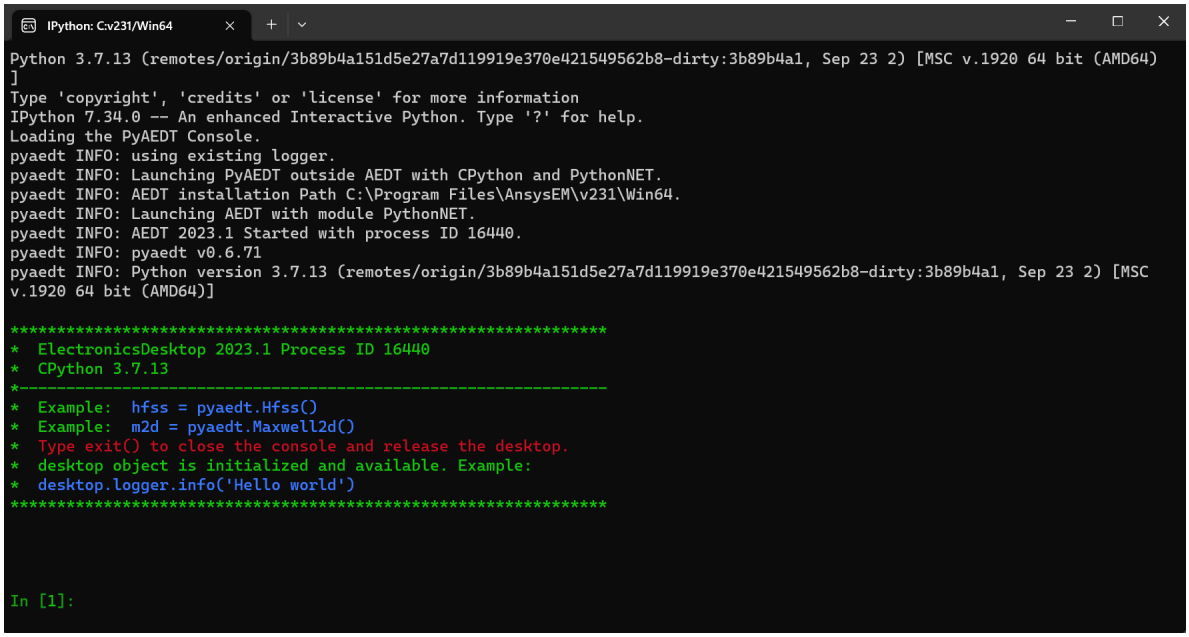
You have multiple options for installing and launching the AEDT Antenna Toolkit:

- You can install the toolkit directly in AEDT using an installation script and then launch it as a wizard. For more information, see *Install toolkit in AEDT and launch the Antenna wizard*.
- You can install the toolkit from a Python console and then launch the Antenna wizard. For more information, see *Install toolkit from Python console and launch the Antenna wizard*.
- You can install the toolkit from a Python console and then use the toolkits APIs. For more information, see *Install toolkit from Python console and use the toolkits APIs*.

1.2.1 Install toolkit in AEDT and launch the Antenna wizard

You can install the AEDT Antenna Toolkit directly in AEDT using the base interpreter from the AEDT installation.

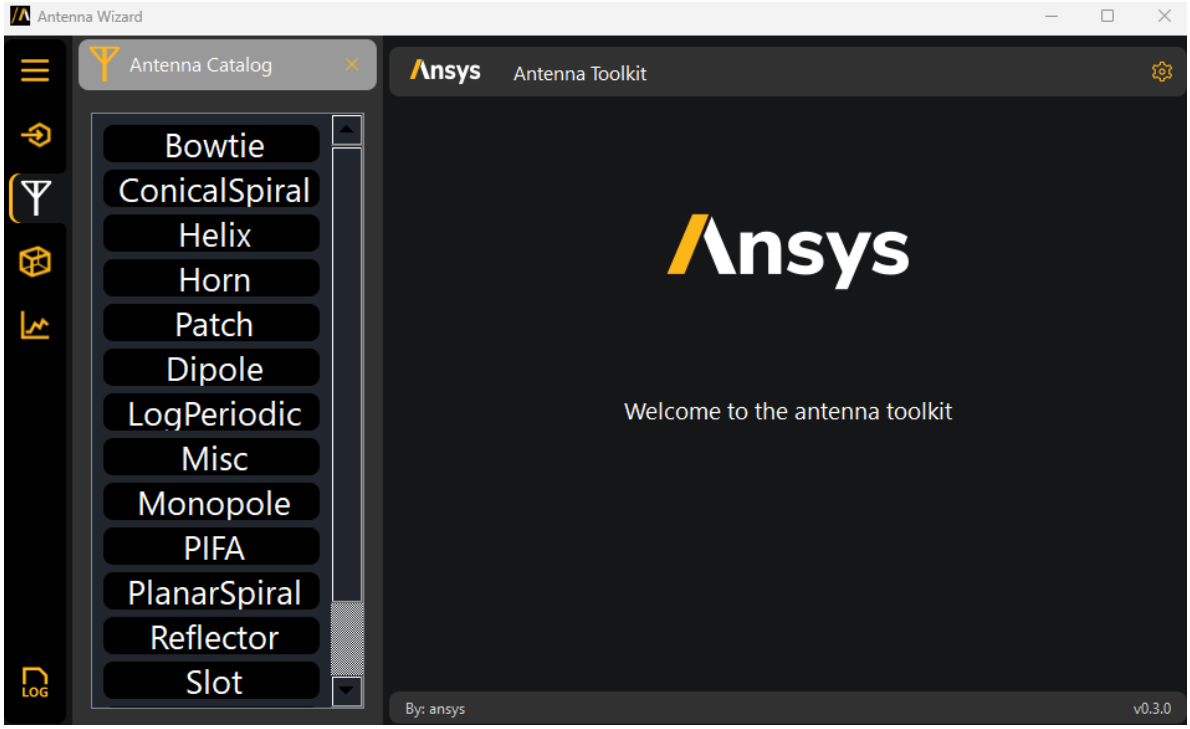
1. From Install from a Python file, follow the steps to install PyAEDT inside AEDT.
2. In AEDT, select **Tools > Toolkit > PyAEDT > Console** to load the PyAEDT console:



3. In the PyAEDT console, run these commands to add the Antenna Toolkit as a wizard (toolkit UI) in AEDT:

```
desktop.add_custom_toolkit("AntennaWizard")
exit()
```

4. In the AEDT toolbar, click the **AntennaWizard** button to open this wizard in AEDT:



The Antenna Toolkit Wizard is connected directly to the AEDT session. For wizard usage information, see *Antenna wizard*.

1.2.2 Install toolkit from Python console and launch the Antenna wizard

You can install the AEDT Antenna Toolkit in a specific Python environment from the AEDT console.

> **Note**
>
> If you have an existing virtual environment, skip step 1.

> **Note**
>
> If you have already installed the toolkit in your virtual environment, skip step 2.

1. Create a fresh-clean Python environment and activate it:

```
# Create a virtual environment
python -m venv .venv

# Activate it in a POSIX system
source .venv/bin/activate

# Activate it in a Windows CMD environment
.venv\Scripts\activate.bat

# Activate it in Windows PowerShell
.venv\Scripts\Activate.ps1
```

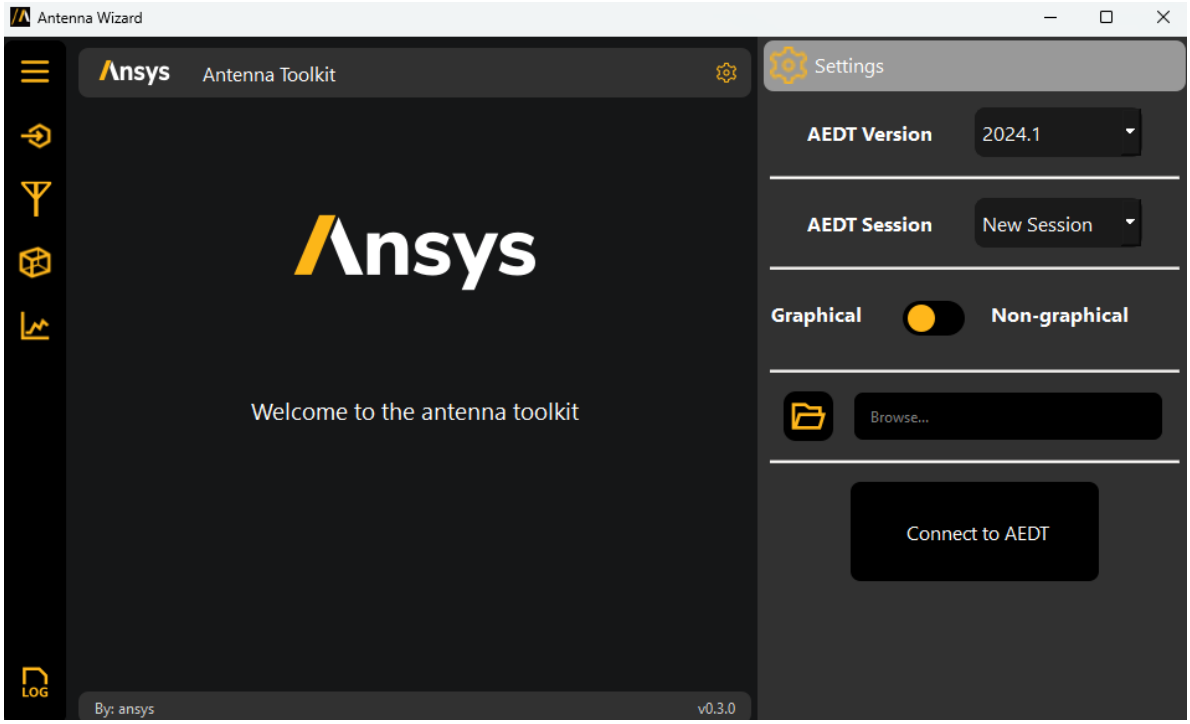
2. Install the toolkit from the GitHub repository:

```
python -m pip install pyaedt-toolkits-antenna[all]
```

3. Launch the Antenna Toolkit Wizard:

```
python .venv\Lib\site-packages\ansys\aedt\toolkits\antenna\run_toolkit.py
```

4. On the **AEDT Settings** tab, create an AEDT session or connect to an existing one:



For wizard usage information, see *Antenna wizard*.

1.2.3 Install toolkit from Python console and use the toolkits APIs

You can install the toolkit in a specific Python environment and use the toolkits APIs. The code example included in this topic shows how to use the APIs at the model level and toolkit level.

> **Note**
>
> If you have an existing virtual environment, skip step 1.

> **Note**
>
> If you have already installed the toolkit in your virtual environment, skip step 2.

1. Create a fresh-clean Python environment and activate it:

```
# Create a virtual environment
python -m venv .venv

# Activate it in a POSIX system
source .venv/bin/activate

# Activate it in a Windows CMD environment
.venv\Scripts\activate.bat

# Activate it in Windows PowerShell
.venv\Scripts\Activate.ps1
```

2. Install the toolkit from the GitHub repository:

```
python -m pip install pyaedt-toolkits-antenna
```

3. Open a Python console in your virtual environment:

```
python
```

4. From the command line, use the toolkit to create an antenna.

   This code shows how to launch AEDT, create and synthesize a bowtie antenna, and run a simulation in HFSS:

```
# Import required modules
from ansys.aedt.core import Hfss
from ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie import BowTie

# Open AEDT and create an HFSS design
aedtapp = Hfss()

# Create antenna object
oantenna1 = BowTie(aedtapp)

# Change frequency
oantenna1.frequency = 12.0

# Create antenna in HFSS
oantenna1.model_hfss()

# Create setup in HFSS
oantenna1.setup_hfss()

# Release AEDT
aedtapp.release_desktop()
```

5. To create an antenna from the toolkit level, use the `Toolkit` class.

   This code shows how to use the `Toolkit` class to get available antennas and their properties, open AEDT, update antenna properties, and create a bowtie antenna:

```
# Import required modules
import time
from ansys.aedt.toolkits.antenna.backend.api import ToolkitBackend

# Backend object
toolkit = ToolkitBackend()

# Get available antennas
toolkit.available_antennas

# Get properties
properties = toolkit.get_properties()

# Set properties
properties = toolkit.set_properties({"length_unit": "cm"})
```

(continues on next page)

(continued from previous page)

```
# Launch AEDT in a thread
toolkit.launch_aedt()

# Wait until thread is finished
idle = toolkit.wait_to_be_idle()

# Update antenna properties
response = toolkit.set_properties({"substrate_height": 0.1575, "length_unit": "cm"})

# Create a bowtie antenna
toolkit.get_antenna("BowTie")

# Release AEDT
toolkit.release_aedt()
```

CHAPTER

TWO

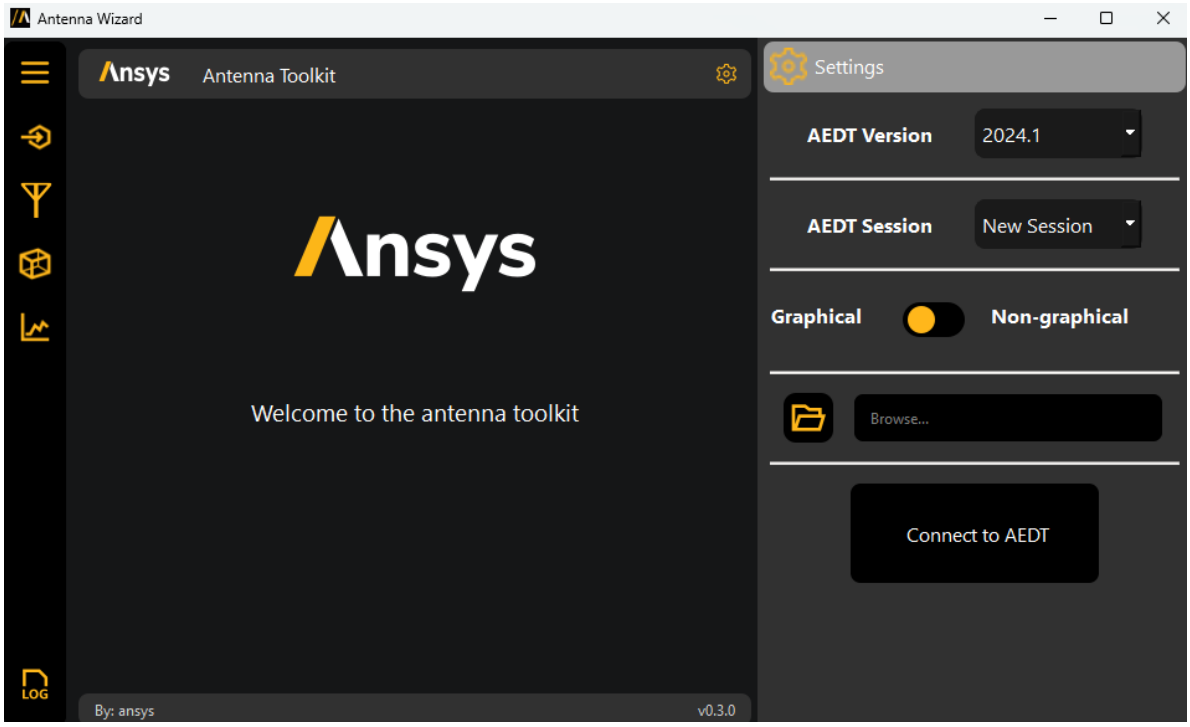
ANTENNA WIZARD

This section describes how to use the Antenna wizard. It assumes that you have already launched the wizard from either the AEDT menu or AEDT console. For toolkit installation and wizard launching information, see these topics:

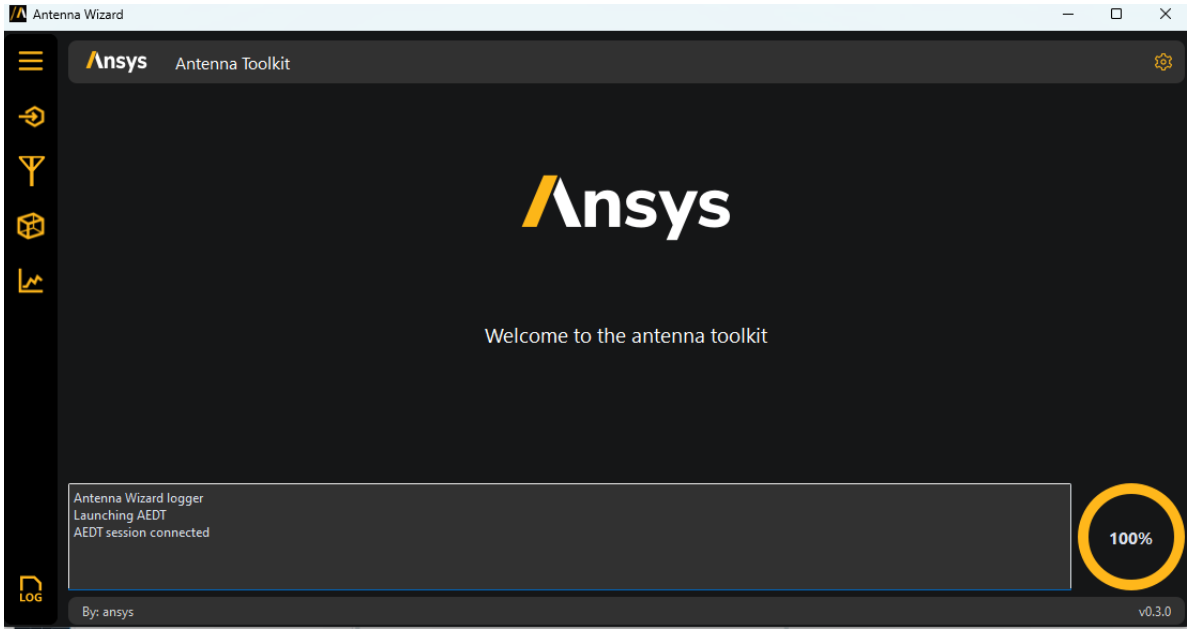
- *Install toolkit in AEDT and launch the Antenna wizard*
- *Install toolkit from Python console and launch the Antenna wizard*

1. On the **Settings** tab, specify settings for either creating an AEDT session or connecting to an existing AEDT session.

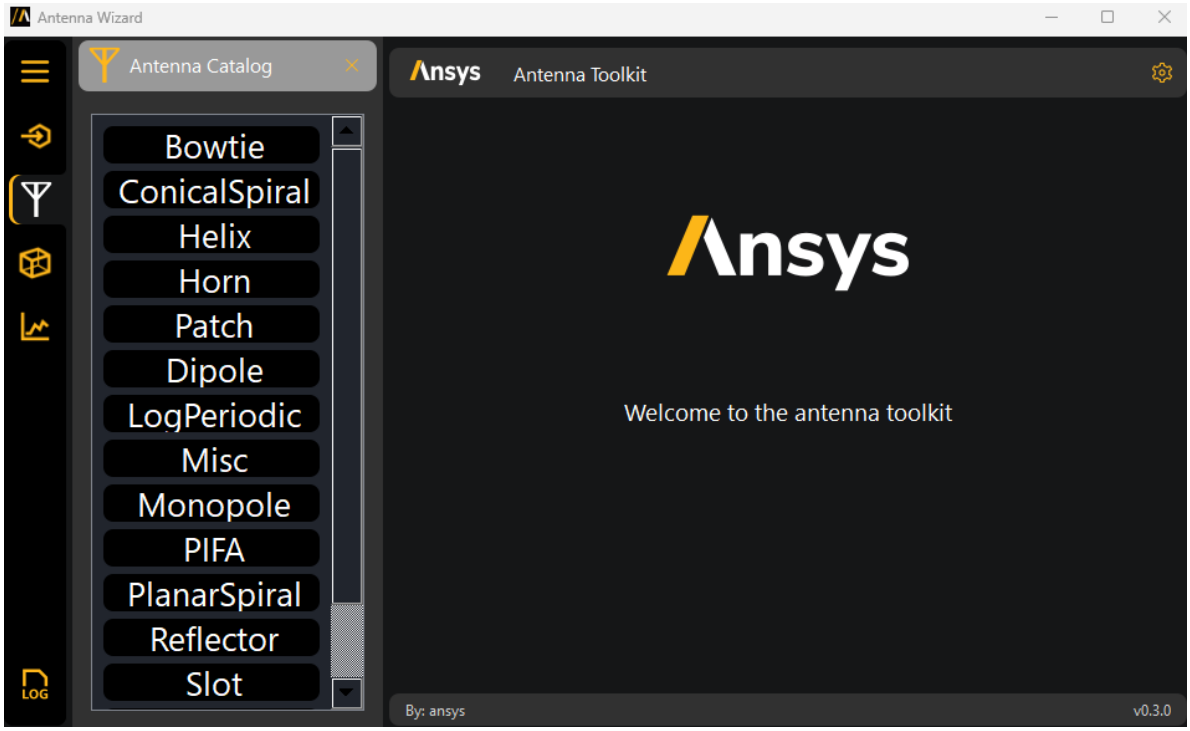
> **Note**
>
> If the Antenna Toolkit Wizard is launched from AEDT, the **Settings** tab does not appear because the toolkit is directly connected to the specific AEDT session.

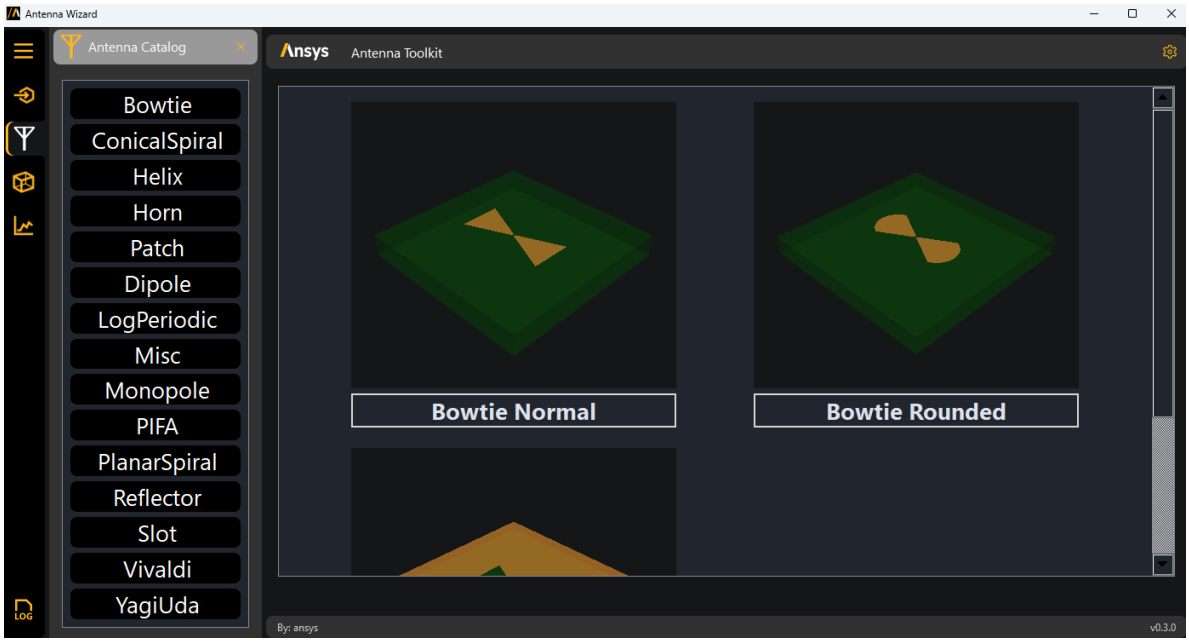


The wizard has a progress circle and a logger box, where you can see the status of every operation.

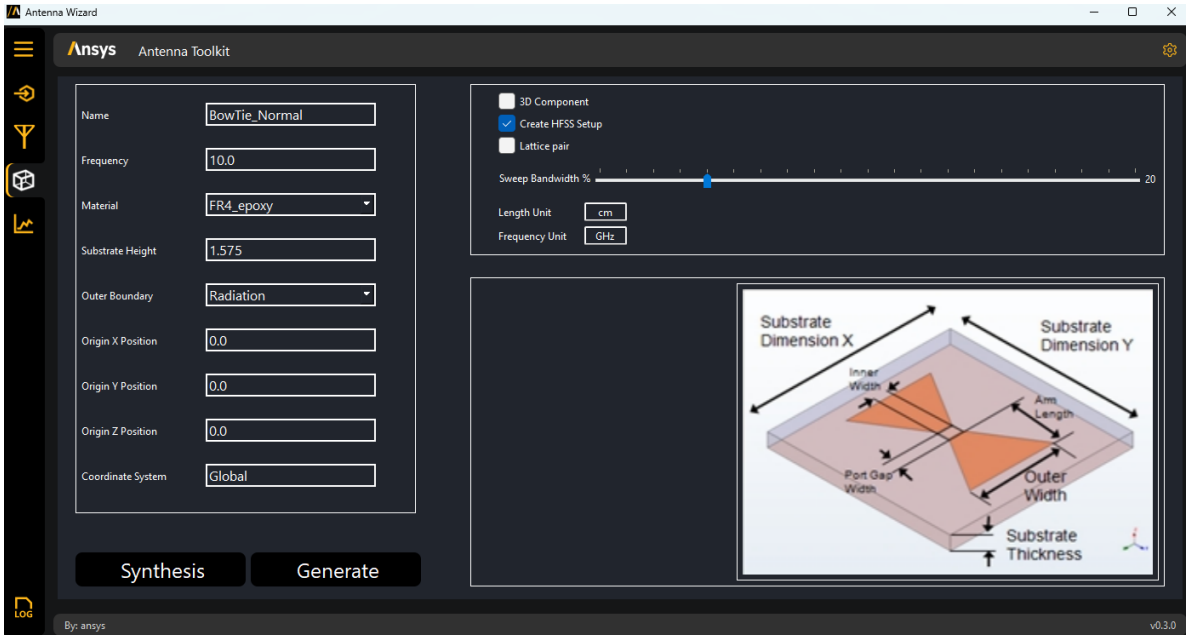


You can choose different antennas from the **Antenna catalog** menu to load the antennas template.





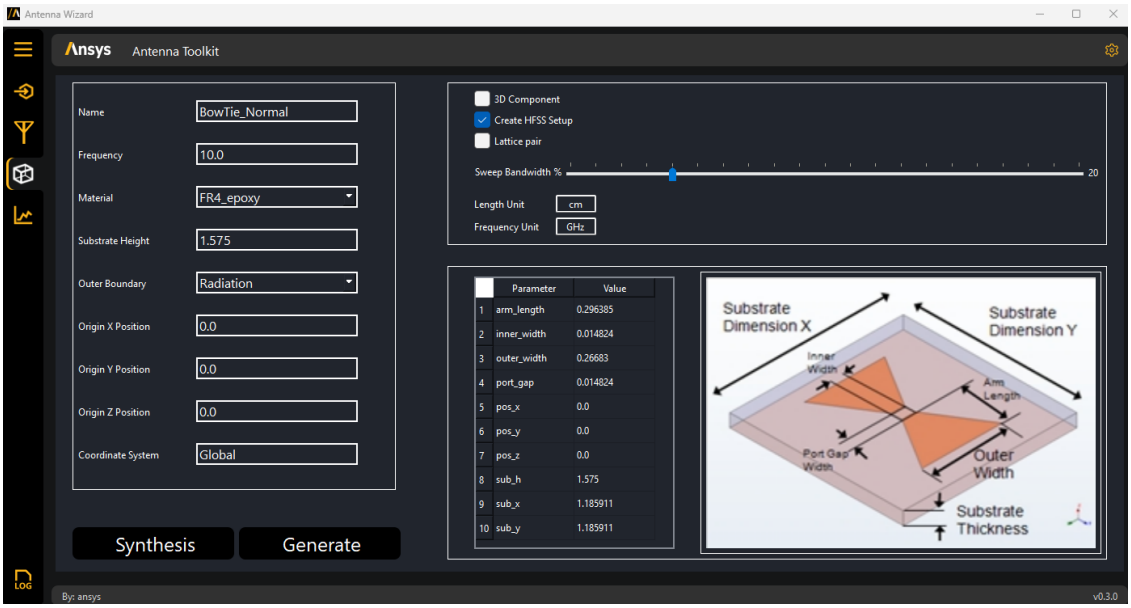
For example, if you select **Antennas > Bowtie > Bowtie Normal**, the central page is updated to the **Synthesis** page and it shows the antenna template:



You have two options: **Synthesis** and **Generate**. The **Generate** button is unavailable if the wizard is not connected to AEDT.

- The **Synthesis** button is for performing the synthesis of the antenna. A connection to AEDT

is not needed.
  You can see the parameters that control the antenna geometry. Additionally, you can do as many syntheses as you want and even change the antenna template.



- The **Generate** button is for creating an HFSS model. It uses the **3D Component**, **Create Hfss Setup**, and **Lattice pair** checkboxes along with the **Sweep Bandwidth %** option It also uses the length and frequency unit to perform the HFSS setup.

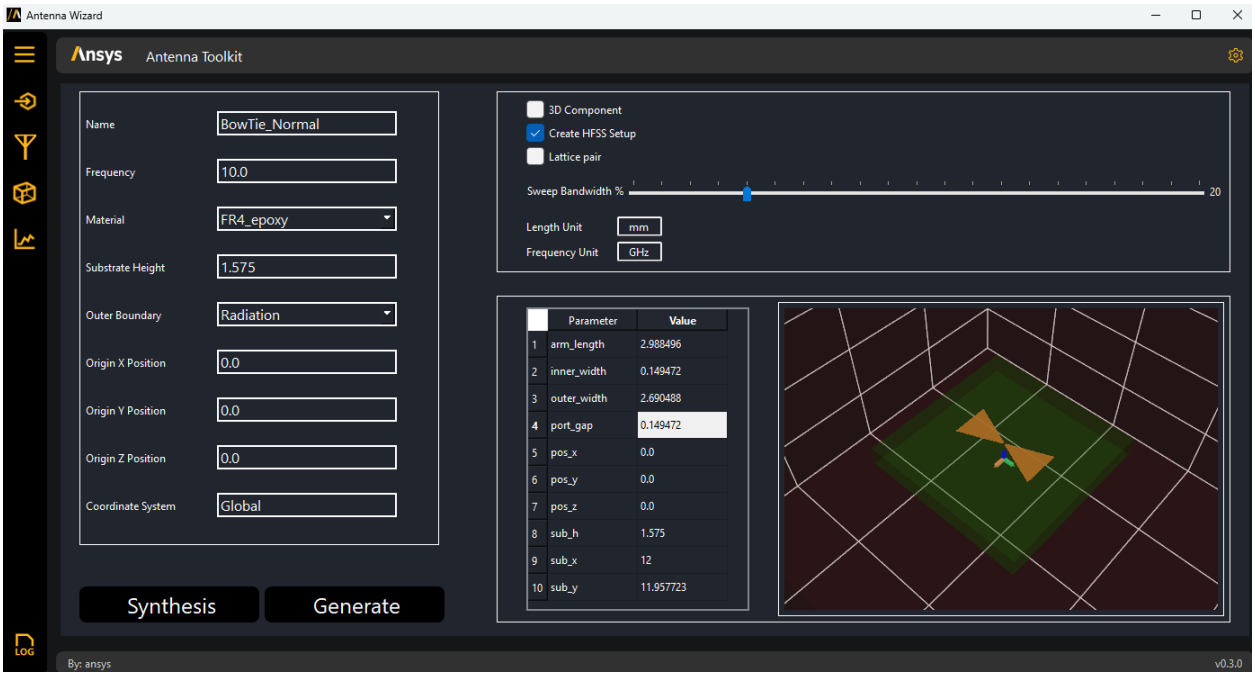
  Descriptions follow for how to use the checkboxes on the **Design** tab:

  – If you select the **3D Component** checkbox, the toolkit creates the antenna and replaces it with a 3D component.

  – If you select the **Generate** checkbox, the toolkit automatically creates the boundaries, excitations, and ports needed to simulate the antenna. Once you create an HFSS model, you cannot create another antenna. Both the **Synthesis** and **Generate** buttons become unavailable. If you want to create another antenna, you must restart the toolkit.

  – If you select the **Lattice pair** checkbox, the toolkit creates a unit cell assigning a lattice pair boundary.

Once you create an antenna, the **Synthesis** tab displays an interactive 3D model rather than the image of the antenna template:



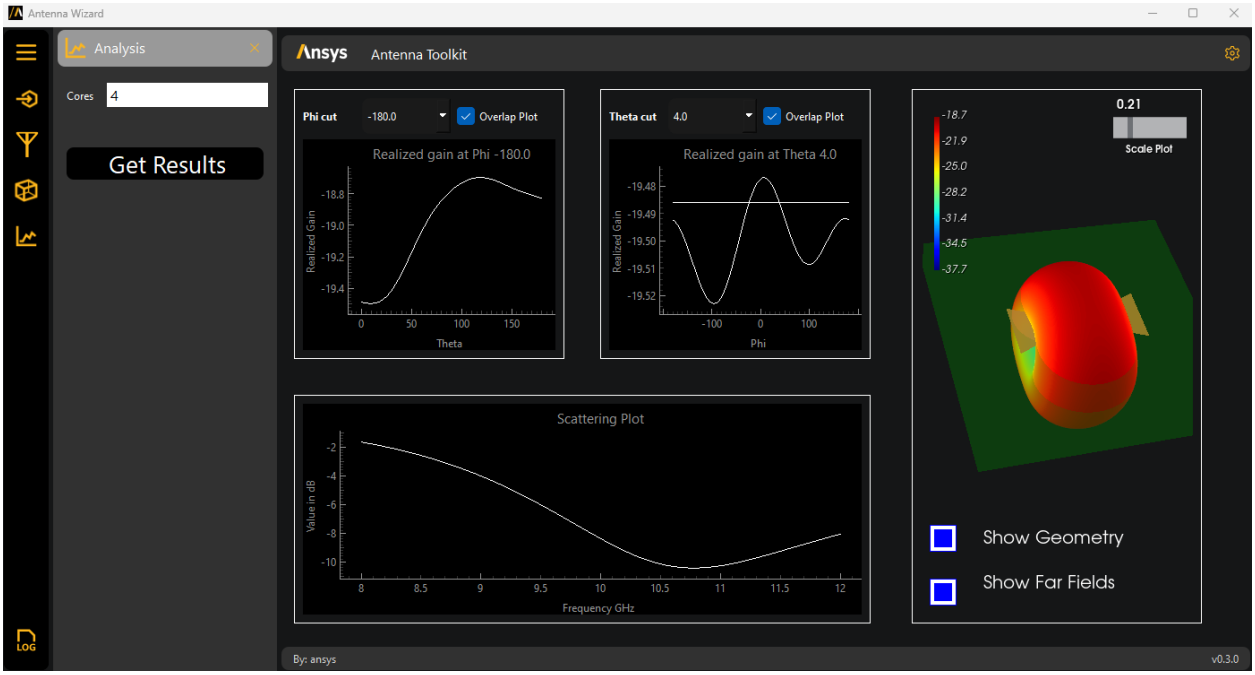
If AEDT is launched in non-graphical mode, you can still see the generated model.

In the wizard, you can modify the parameters interactively, watching both the HFSS model and the interactive 3D plot in the wizard change.

Finally, on the wizards **Analysis** tab, you have the **Get results** button. This second button is unavailable until after you analyze the HFSS design.

When you click **Get results**, the project is analyzed. You can specify the number of cores to use in the simulation.

Once the project is solved, you can click **Get results** on the **Analysis** tab to view results.



CHAPTER

THREE

API REFERENCE

This section provides descriptions of the two APIs available for the AEDT Antenna Toolkit:

- **Toolkit API**: Contains the `Toolkit` class, which provides methods for controlling the toolkit workflow. This API provides methods for synthesizing and creating an antenna. You use the Toolkit API at the toolkit level.
- **Antenna API**: Contains classes for all antenna types available in the toolkit. You use the Antenna API at the model level.

3.1 Toolkit API

The Toolkit API contains the `Toolkit` class, which provides methods for controlling the toolkit workflow. This API provides methods for synthesizing and creating an antenna. You use the Toolkit API at the toolkit level.

The common methods for creating an AEDT session or connecting to an existing AEDT session are provided by the Common PyAEDT toolkit library.

ToolkitBackend()	Provides methods for controlling the toolkit workflow.

3.1.1 ansys.aedt.toolkits.antenna.backend.api.ToolkitBackend

class ansys.aedt.toolkits.antenna.backend.api.**ToolkitBackend**

Provides methods for controlling the toolkit workflow.

This class provides methods for creating an AEDT session, connecting to an existing AEDT session, and synthesizing and creating an antenna in HFSS.

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.api import ToolkitBackend
>>> import time
>>> toolkit = ToolkitBackend()
>>> msg1 = toolkit.launch_aedt()
>>> toolkit.wait_to_be_idle()
>>> toolkit.get_antenna("BowTie")
```

__init__()

Methods

__init__()	
aedt_sessions()	Get information for the active AEDT sessions.
analyze()	Analyze the design.
connect_aedt()	Connect to an existing AEDT session.
connect_design([app_name])	Connect to an application design.
export_aedt_model(*args, **kwargs)	
export_farfield([frequencies, setup, ...])	Export far field data and then encode the file if the encode parameter is enabled.
get_antenna(antenna[, synth_only])	Synthesize and create an antenna in HFSS.
get_design_names()	Get the design names for a specific project.
get_project_name(project_path)	Get the project name from the project path.
get_properties()	Get the toolkit properties.
get_thread_status()	Get the toolkit thread status.
handle_export_failure()	Error handler.
installed_aedt_version()	Get the installed AEDT versions.
is_aedt_connected()	Check if AEDT is connected.
launch_aedt()	Launch AEDT.
launch_thread(process, *args)	Launch the thread.
open_project([project_name])	Open an AEDT project.
release_aedt([close_projects, close_on_exit])	Release AEDT.
save_project([project_path, release_aedt])	Save the project.
scattering_results()	Get antenna scattering results.
serialize_obj_base64(file_path)	Encode a bytes-like object.
set_properties(data)	Assign the passed data to the internal data model.
update_hfss_parameters(key, val)	Update parameters in HFSS.
wait_to_be_idle([timeout])	Wait for the thread to be idle and ready to accept a new task.

You can use the Toolkit API as shown in this example:

```
# Import required modules for the example
import time

# Import backend
from ansys.aedt.toolkits.template.backend.api import ToolkitBackend

# Initialize generic service
toolkit_api = ToolkitBackend()

# Load default properties from a JSON file
properties = toolkit_api.get_properties()

# Set properties
new_properties = {"aedt_version": "2023.1"}
toolkit_api.set_properties(new_properties)
properties = toolkit_api.get_properties()

# Launch AEDT
```

(continues on next page)

(continued from previous page)

```
thread_msg = toolkit_api.launch_thread(toolkit_api.launch_aedt)

# Wait until thread is finished
idle = toolkit_api.wait_to_be_idle()
if not idle:
    print("AEDT not initialized.")
    sys.exit()

# Create geometry
toolkit_api.connect_design("HFSS")

# Create setup when antenna is created
properties.antenna.setup.create_setup = True
properties.antenna.synthesis.outer_boundary = "Radiation"

# Generate antenna
antenna_parameter = toolkit_api.get_antenna("RectangularPatchProbe")

# Release AEDT
toolkit_api.release_aedt()
```

3.2 Antenna API

The Antenna API contains classes for all antenna types available in the toolkit:

3.2.1 Bowtie

This page list the classes available for bowtie antennas:

BowTieNormal(*args, **kwargs)	Manages a bowtie antenna.
BowTieRounded(*args, **kwargs)	Manages a bowtie rounded antenna.
BowTieSlot(*args, **kwargs)	Manages a bowtie slot antenna.

ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie.BowTieNormal

class ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie.**BowTieNormal**(**args*, ***kwargs*)

Manages a bowtie antenna.

This class is accessible through the `Hfss` object [**?**].

Parameters

frequency
[`float`, `optional`] Center frequency. The default is `10.0`.

frequency_unit
[`str`, `optional`] Frequency units. The default is `"GHz"`.

material
[`str`, `optional`] Substrate material. If a material is not defined, a new material, `parametrized`, is defined. The default is `"FR4_epoxy"`.

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

substrate_height
[float, optional] Substrate height. The default is 1.575.

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.BowTie
Bowtie antenna object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie import␣
↪BowTieNormal
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = BowTieNormal(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> oantenna2 = BowTieNormal(app, origin=[200, 50, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model the bowtie antenna in PyDiscovery.
model_hfss()	Draw a bowtie antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up the model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Substrate material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.
substrate_height	Substrate height.

ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie.BowTieRounded

class ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie.**BowTieRounded**(**args*, ***kwargs*)

Manages a bowtie rounded antenna.

This class is accessible through the Hfss object [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Substrate material. If a material is not defined, a new material, parametrized, is defined. The default is "FR4_epoxy".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

substrate_height
[float, optional] Substrate height. The default is 1.575.

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.BowTieRounded
Patch antenna object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie import␣
↪BowTieRounded
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = BowTieRounded(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> oantenna2 = BowTieRounded(app, origin=[200, 50, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw a bowtie rounded antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up the model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Substrate material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.
substrate_height	Substrate height.

ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie.BowTieSlot

class ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie.**BowTieSlot**(**args*, ***kwargs*)

Manages a bowtie slot antenna.

This class is accessible through the `Hfss` object [**?**].

Parameters

frequency
[`float`, `optional`] Center frequency. The default is `10.0`.

frequency_unit
[`str`, `optional`] Frequency units. The default is `"GHz"`.

material
[`str`, `optional`] Substrate material. If a material is not defined, a new material, `parametrized`, is defined. The default is `"FR4_epoxy"`.

outer_boundary
[`str`, `optional`] Boundary type to use. The default is `None`. Options are `"FEBI"`, `"PML"`, `"Radiation"`, and `None`.

length_unit
[`str`, `optional`] Length units. The default is `"mm"`.

substrate_height
[`float`, `optional`] Substrate height. The default is `0.1575`.

parametrized
[bool, `optional`] Whether to create a parametrized antenna. The default is `True`.

Returns

aedt.toolkits.antenna.BowTieSlot
Bowtie antenna object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie import BowTieSlot
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = BowTieSlot(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> oantenna2 = BowTieSlot(app, origin=[200, 50, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw a bowtie slot antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up the model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Substrate material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.
substrate_height	Substrate height.

You must use these methods from PyAEDT as shown in this example:

```
from ansys.aedt.core import Hfss

from ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie import BowTieNormal

aedtapp = Hfss()

# Create antenna
oantenna1 = BowTieNormal(app)
oantenna1.frequency = 12.0
oantenna1.model_hfss()
oantenna1.setup_hfss()
...
aedtapp.release_desktop()
```

3.2.2 Common

This pages lists common methods available in the Antenna API:

TransmissionLine([frequency, frequency_unit])	Provides base methods common to transmission line calculations.
StandardWaveguide([frequency, frequency_unit])	Provides base methods common to standard waveguides.

ansys.aedt.toolkits.antenna.backend.antenna_models.common.TransmissionLine

class ansys.aedt.toolkits.antenna.backend.antenna_models.common.**TransmissionLine**(*frequency=10*, *frequency_unit='GHz'*)

Provides base methods common to transmission line calculations.

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

Returns

aedt.toolkits.antenna.common.TransmissionLine
Transmission line calculator object.

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.common import␣
↪TransmissionLine
>>> tl_calc = TransmissionLine(frequency=2)
>>> tl_calc.stripline_calculator(substrate_height=10, permittivity=2.2,␣
↪impedance=60)
```

__init__(*frequency=10*, *frequency_unit='GHz'*)

Methods

__init__([frequency, frequency_unit])	
microstrip_calculator(substrate_height, ...)	Use the micro strip line calculator to calculate line width and length.
stripline_calculator(substrate_height, ...)	Use the strip line calculator to calculate line width.
suspended_strip_calculator(wavelength, w1, ...)	Use the suspended strip line calculator to calculate effective permittivity.

ansys.aedt.toolkits.antenna.backend.antenna_models.common.StandardWaveguide

class ansys.aedt.toolkits.antenna.backend.antenna_models.common.**StandardWaveguide**(*frequency=10*, *frequency_unit='GHz'*)

Provides base methods common to standard waveguides.

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

Returns

aedt.toolkits.antenna.common.StandardWaveguide
Standard waveguide object.

Examples

```
>>> from ansys.aedt.toolkits.antenna.common import StandardWaveguide
>>> wg_calc = StandardWaveguide()
>>> wg_dim = wg_calc.get_waveguide_dimensions("WR-75")
```

__init__(*frequency=10*, *frequency_unit='GHz'*)

Methods

__init__([frequency, frequency_unit])	
find_waveguide(freq[, units])	Find the closest standard waveguide for the operational frequency.
get_waveguide_dimensions(name[, units])	Get waveguide dimensions.

Attributes

waveguide_list	Standard waveguide list.
wg	

You must use these methods from PyAEDT as shown in this example:

```
from ansys.aedt.toolkits.antenna.backend.antenna_models.common import TransmissionLine

# Transmission line calculation
tl_calc = TransmissionLine(frequency=2)
tl_calc.stripline_calculator(substrate_height=10, permittivity=2.2, impedance=60)
```

3.2.3 Conical spiral

This page lists the classes available for conical spiral antennas:

Archimedean(*args, **kwargs)	Manages conical archimedeal spiral antenna.

ansys.aedt.toolkits.antenna.backend.antenna_models.conical_spiral.Archimedean

class ansys.aedt.toolkits.antenna.backend.antenna_models.conical_spiral.**Archimedean**(**args*, ***kwargs*)

Manages conical archimedeal spiral antenna.

This class is accessible through the app hfss object [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Horn material. If a material is not defined, a new material, parametrized, is defined. The default is "pec".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.Archimedean
Conical archimedean spiral object.

Notes

Examples

```
>>> from ansys.aedt.core import Hfss
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.conical_spiral import␣
↪Archimedean
>>> hfss = Hfss()
>>> antenna = Archimedean(hfss, start_frequency=20.0,
...                               stop_frequency=50.0, frequency_unit="GHz",
...                               outer_boundary='Radiation', length_unit="mm",
...                               antenna_name="Archimedean", origin=[1, 100, 50])
>>> antenna.model_hfss()
>>> antenna.setup_hfss()
>>> hfss.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw a conical archimidean spiral antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Central frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Horn material.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.
start_frequency	Start frequency.
stop_frequency	Stop frequency.

You must use these methods from PyAEDT as shown in this example:

```
from ansys.aedt.core import Hfss

from ansys.aedt.toolkits.antenna.backend.antenna_models.conical_spiral import (
    Archimedean,
)

aedtapp = Hfss()

# Create antenna
oantenna1 = RectangularPatchProbe(app)
oantenna1.frequency = 12.0
oantenna1.model_hfss()
oantenna1.setup_hfss()
...
aedtapp.release_desktop()
```

3.2.4 Helix

This pages lists the classes available for helix antennas:

AxialMode(*args, **kwargs)	Manages an axial mode helix antenna.

ansys.aedt.toolkits.antenna.backend.antenna_models.helix.AxialMode

class ansys.aedt.toolkits.antenna.backend.antenna_models.helix.**AxialMode**(**args*, ***kwargs*)

Manages an axial mode helix antenna.

This class is accessible through the `Hfss` object [**?**].

Parameters

frequency
[`float`, `optional`] Center frequency. The default is `10.0`.

frequency_unit
[`str`, `optional`] Frequency units. The default is `"GHz"`.

material
[`str`, `optional`] Helix material. If the material is not defined, a new material, `parametrized`, is defined. The default is `"pec"`.

outer_boundary
[`str`, `optional`] Boundary type to use. The default is `None`. Options are `"FEBI"`, `"PML"`, `"Radiation"`, and `None`.

length_unit
[`str`, `optional`] Length units. The default is `"mm"`.

parametrized
[bool, `optional`] Whether to create a parametrized antenna. The default is `True`.

Returns

aedt.toolkits.antenna.AxialMode
Antenna object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.helix import AxialMode
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = AxialMode(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> oantenna2 = AxialMode(app, origin=[200, 50, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw an axial mode antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
direction	Helix direction.
feeder_length	Helix feeder length.
frequency	Center frequency.
frequency_unit	Frequency units.
gain	Helix expected gain.
length_unit	Length unit.
material	Helix material.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.

You must use these methods from PyAEDT as shown in this example:

```
from ansys.aedt.core import Hfss

from ansys.aedt.toolkits.antenna.backend.antenna_models.helix import AxialMode

aedtapp = Hfss()

# Create antenna
oantenna1 = AxialMode(app)
oantenna1.model_hfss()
oantenna1.setup_hfss()
...
aedtapp.release_desktop()
```

3.2.5 Horn

This page lists the classes available for horns:

Conical(*args, **kwargs)	Manages a conical horn antenna.
Corrugated(*args, **kwargs)	Manages a corrugated horn antenna.
Elliptical(*args, **kwargs)	Manages an elliptical horn antenna.
EPlane(*args, **kwargs)	Manages an E plane horn antenna.
HPlane(*args, **kwargs)	Manages an H plane horn antenna.
Pyramidal(*args, **kwargs)	Manages a pyramidal horn antenna.
PyramidalRidged(*args, **kwargs)	Manages a pyramidal ridged horn antenna.
QuadRidged(*args, **kwargs)	Manages a quad-ridged horn antenna.

ansys.aedt.toolkits.antenna.backend.antenna_models.horn.Conical

class ansys.aedt.toolkits.antenna.backend.antenna_models.horn.**Conical**(**args*, ***kwargs*)

Manages a conical horn antenna.

This class is accessible through the app hfss object [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Horn material. If a material is not defined, a new material, parametrized, is defined. The default is "pec".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.ConicalHorn
Conical horn object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.horn import Conical
>>> import ansys.aedt.core
>>> oantenna1 = Conical(None)
>>> oantenna1.frequency = 12.0
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = Conical(app)
>>> oantenna1.model_hfss()
```

(continues on next page)

(continued from previous page)

```
>>> oantenna1.setup_hfss()
>>> oantenna2 = Conical(app, origin=[0.2, 0.5, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw a conical horn antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Horn material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.

ansys.aedt.toolkits.antenna.backend.antenna_models.horn.Corrugated

class ansys.aedt.toolkits.antenna.backend.antenna_models.horn.**Corrugated**(**args*, ***kwargs*)

Manages a corrugated horn antenna.

This class is accessible through the app hfss object [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Horn material. If a material is not defined, a new material, parametrized, is defined. The default is "pec".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.CorrugatedHorn
Corrugated horn object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.horn import Corrugated
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = Corrugated(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw a conical horn antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Horn material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.

ansys.aedt.toolkits.antenna.backend.antenna_models.horn.Elliptical

class ansys.aedt.toolkits.antenna.backend.antenna_models.horn.**Elliptical**(**args*, ***kwargs*)

Manages an elliptical horn antenna.

This class is accessible through the app hfss object [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Horn material. If a material is not defined, a new material, parametrized, is defined. The default is "pec".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.EllipticalHorn
Elliptical horn object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.horn import␣
↪PyramidalRidged
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = Elliptical(app)
>>> oantenna1.frequency = 12.0
```

(continues on next page)

(continued from previous page)

```
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> oantenna2 = Elliptical(app, origin=[0.2, 0.5, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw elliptical horn antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Horn material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.

ansys.aedt.toolkits.antenna.backend.antenna_models.horn.EPlane

class ansys.aedt.toolkits.antenna.backend.antenna_models.horn.**EPlane**(**args*, ***kwargs*)

Manages an E plane horn antenna.

This class is accessible through the app hfss object [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Horn material. If a material is not defined, a new material, parametrized, is defined. The default is "pec".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.EPlaneHorn
E plane horn object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.horn import EPlane
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = EPlane(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> oantenna2 = EPlane(app, origin=[0.2, 0.5, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw E plane horn antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Horn material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.

ansys.aedt.toolkits.antenna.backend.antenna_models.horn.HPlane

class ansys.aedt.toolkits.antenna.backend.antenna_models.horn.**HPlane**(**args*, ***kwargs*)

Manages an H plane horn antenna.

This class is accessible through the app hfss object [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Horn material. If a material is not defined, a new material, parametrized, is defined. The default is "pec".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.HPlaneHorn
H plane horn object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.horn import HPlane
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = HPlane(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
```

(continues on next page)

(continued from previous page)

```
>>> oantenna1.setup_hfss()
>>> oantenna2 = HPlane(app, origin=[0.2, 0.5, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw H plane horn antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Horn material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.

ansys.aedt.toolkits.antenna.backend.antenna_models.horn.Pyramidal

class ansys.aedt.toolkits.antenna.backend.antenna_models.horn.**Pyramidal**(**args*, ***kwargs*)

Manages a pyramidal horn antenna.

This class is accessible through the app hfss object [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Horn material. If a material is not defined, a new material, parametrized, is defined. The default is "pec".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.Pyramidal
Pyramidal horn object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.horn import Pyramidal
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = Pyramidal(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> oantenna2 = Pyramidal(app, origin=[0.2, 0.5, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw pyramidal horn antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Horn material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.

ansys.aedt.toolkits.antenna.backend.antenna_models.horn.PyramidalRidged

class ansys.aedt.toolkits.antenna.backend.antenna_models.horn.**PyramidalRidged**(**args*, ***kwargs*)

Manages a pyramidal ridged horn antenna.

This class is accessible through the app hfss object [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Horn material. If a material is not defined, a new material, parametrized, is defined. The default is "pec".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.PyramidalRidged
Pyramidal ridged horn object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.horn import␣
↪PyramidalRidged
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = PyramidalRidged(app)
```

(continues on next page)

(continued from previous page)

```
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> oantenna2 = PyramidalRidged(app, origin=[0.2, 0.5, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw conical horn antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Horn material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.

ansys.aedt.toolkits.antenna.backend.antenna_models.horn.QuadRidged

class ansys.aedt.toolkits.antenna.backend.antenna_models.horn.**QuadRidged**(**args*, ***kwargs*)

Manages a quad-ridged horn antenna.

This class is accessible through the app hfss object [**?**], [**?**], [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Horn material. If a material is not defined, a new material, parametrized, is defined. The default is "pec".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.PyramidalRidged
Pyramidal ridged horn object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.horn import QuadRidged
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = QuadRidged(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> oantenna2 = QuadRidged(app, origin=[0.2, 0.5, 0])
>>> oantenna2.model_hfss()
>>> oantenna2.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw conical horn antenna.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Horn material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.

You must use these methods from PyAEDT as shown in this example:

```
import ansys.aedt.core.Hfss

from ansys.aedt.toolkits.antenna.backend.antenna_models.horn import Conical

aedtapp = Hfss()

# Create antenna
oantenna1 = Conical(app)
oantenna1.frequency = 12.0
oantenna1.model_hfss()
oantenna1.setup_hfss()
...
aedtapp.release_desktop()
```

3.2.6 Patch

This page lists the classes available for patch antennas:

RectangularPatchEdge(*args, **kwargs)	Manages a rectangular patch edge antenna.
RectangularPatchProbe(*args, **kwargs)	Manages a rectangular patch antenna with a coaxial probe.
RectangularPatchInset(*args, **kwargs)	Manages a rectangular patch antenna inset fed.

ansys.aedt.toolkits.antenna.backend.antenna_models.patch.RectangularPatchEdge

class ansys.aedt.toolkits.antenna.backend.antenna_models.patch.**RectangularPatchEdge**(**args*, ***kwargs*)

Manages a rectangular patch edge antenna.

This class is accessible through the `Hfss` object [**?**].

Parameters

frequency
[`float`, `optional`] Center frequency. The default is `10.0`.

frequency_unit
[`str`, `optional`] Frequency units. The default is `"GHz"`.

material
[str, optional] Substrate material. If the material is not defined, a new material, parametrized, is created. The default is "FR4_epoxy".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

substrate_height
[float, optional] Substrate height. The default is 1.575.

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.RectangularPatchEdge
Patch antenna object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.patch import␣
↪RectangularPatchEdge
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = RectangularPatchEdge(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw a rectangular patch edge antenna inset fed.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up the model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Substrate material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.
substrate_height	Substrate height.

ansys.aedt.toolkits.antenna.backend.antenna_models.patch.RectangularPatchProbe

class ansys.aedt.toolkits.antenna.backend.antenna_models.patch.**RectangularPatchProbe**(**args*, ***kwargs*)

Manages a rectangular patch antenna with a coaxial probe.

This class is accessible through the `Hfss` object [**?**].

Parameters

frequency
[`float`, `optional`] Center frequency. The default is `10.0`.

frequency_unit
[`str`, `optional`] Frequency units. The default is `"GHz"`.

material
[`str`, `optional`] Substrate material. If the material is not defined, a new material, `parametrized`, is created. The default is `"FR4_epoxy"`.

outer_boundary
[`str`, `optional`] Boundary type to use. The default is `None`. Options are `"FEBI"`, `"PML"`, `"Radiation"`, and `None`.

length_unit
[`str`, `optional`] Length units. The default is `"mm"`.

substrate_height
[`float`, `optional`] Substrate height. The default is `1.575`.

parametrized
[bool, `optional`] Whether to create a parametrized antenna. The default is `True`.

Returns

`aedt.toolkits.antenna.RectangularPatchProbe`
Patch antenna object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.patch import␣
↪RectangularPatchProbe
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = RectangularPatchProbe(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw rectangular patch antenna with coaxial probe.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up the model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Substrate material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.
substrate_height	Substrate height.

ansys.aedt.toolkits.antenna.backend.antenna_models.patch.RectangularPatchInset

class ansys.aedt.toolkits.antenna.backend.antenna_models.patch.**RectangularPatchInset**(**args*, ***kwargs*)

Manages a rectangular patch antenna inset fed.

This class is accessible through the Hfss object [**?**].

Parameters

frequency
[float, optional] Center frequency. The default is 10.0.

frequency_unit
[str, optional] Frequency units. The default is "GHz".

material
[str, optional] Substrate material. If the material is not defined, a new material, parametrized, is created. The default is "FR4_epoxy".

outer_boundary
[str, optional] Boundary type to use. The default is None. Options are "FEBI", "PML", "Radiation", and None.

length_unit
[str, optional] Length units. The default is "mm".

substrate_height
[float, optional] Substrate height. The default is 1.575.

parametrized
[bool, optional] Whether to create a parametrized antenna. The default is True.

Returns

aedt.toolkits.antenna.RectangularPatchInset
Patch antenna object.

Notes

Examples

```
>>> from ansys.aedt.toolkits.antenna.backend.antenna_models.patch import␣
↪RectangularPatchInset
>>> import ansys.aedt.core
>>> app = ansys.aedt.core.Hfss()
>>> oantenna1 = RectangularPatchInset(app)
>>> oantenna1.frequency = 12.0
>>> oantenna1.model_hfss()
>>> oantenna1.setup_hfss()
>>> app.release_desktop(False, False)
```

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
create_3dcomponent([component_file, ...])	Create a 3D component of the antenna.
create_lattice_pair([lattice_height, ...])	Create a lattice pair box.
duplicate_along_line(vector[, num_clones])	Duplicate the object along a line.
init_model()	Create a radiation boundary.
model_disco()	Model in PyDiscovery.
model_hfss()	Draw a rectangular patch antenna inset fed.
set_variables_in_hfss([not_used])	Create HFSS design variables.
setup_disco()	Set up the model in PyDiscovery.
setup_hfss()	Set up an antenna in HFSS.
synthesis()	Antenna synthesis.
update_synthesis_parameters(new_params)	Update the synthesis parameter from the antenna list.

Attributes

antenna_type	
coordinate_system	Reference coordinate system.
frequency	Center frequency.
frequency_unit	Frequency units.
length_unit	Length unit.
material	Substrate material.
material_properties	Substrate material properties.
name	Antenna name.
origin	Antenna origin.
outer_boundary	Outer boundary.
substrate_height	Substrate height.

You must use these methods from PyAEDT as shown in this example:

```
from ansys.aedt.core import Hfss

from ansys.aedt.toolkits.antenna.backend.antenna_models.patch import (
    RectangularPatchEdge,
)

aedtapp = Hfss()

# Create antenna
oantenna1 = RectangularPatchProbe(app)
oantenna1.frequency = 12.0
oantenna1.model_hfss()
oantenna1.setup_hfss()
...
aedtapp.release_desktop()
```

You use the Antenna API at the model level from PyAEDT.

You can create one or more antennas. An antenna is object-oriented. You can synthesis an antenna without AEDT.

This code shows how to synthesis an antenna:

```
# Import backend
from ansys.aedt.toolkits.antenna.backend.models.horn import Conical

# Synthesize antenna
oantenna1 = Conical()
oantenna1.frequency = 12.0
```

This code shows how to synthesize and create a model of an antenna in HFSS:

```
# Import HFSS
from ansys.aedt.core import Hfss

# Import backend
from ansys.aedt.toolkits.antenna.backend.models.horn import Conical

# Synthesize antenna
aedtapp = Hfss()

# Create antenna
oantenna1 = Conical(app)
oantenna1.model_hfss()
oantenna1.setup_hfss()
...
aedtapp.release_desktop()
```

CHAPTER

FOUR

EXAMPLES

End-to-end examples show how to use the API of the AEDT Antenna Toolkit.

4.1 Antenna synthesis

These examples show how to use the different antenna classes:

4.1.1 Bowtie antenna synthesis

This example demonstrates how to synthesize a bowtie antenna using the `BowTieRounded` class. It initiates AEDT through PyAEDT, sets up an empty HFSS design, and proceeds to create the antenna.

Perform required imports

Import the antenna toolkit class and PyAEDT.

[1]:
```
import tempfile
```

[2]:
```
import ansys.aedt.core
```

[3]:
```
from ansys.aedt.toolkits.antenna.backend.antenna_models.bowtie import BowTieRounded
```

Set AEDT version

Set AEDT version.

[4]:
```
aedt_version = "2025.1"
```

Set non-graphical mode

Set non-graphical mode.

[5]:
```
non_graphical = True
```

Create temporary directory

[6]:
```
temp_dir = tempfile.TemporaryDirectory(suffix="_ansys")
project_name = ansys.aedt.core.generate_unique_project_name(root_name=temp_dir.name,␣
↪project_name="bowtie_example")
```

Create antenna object only for synthesis

Create antenna object.

[7]:
```
oantenna1 = BowTieRounded(None)
print(
    "Arm length: {}{} at {}{}".format(
        str(oantenna1.synthesis_parameters.arm_length.value),
        oantenna1.length_unit,
        oantenna1.frequency,
        oantenna1.frequency_unit,
    )
)
```

```
Arm length: 3.7mm at 10.0GHz
```

Change synthesis frequency

Modify resonance frequency and modify parameters.

[8]:
```
oantenna1.frequency = 12.0
print(
    "Arm length: {}{} at {}{}".format(
        str(oantenna1.synthesis_parameters.arm_length.value),
        oantenna1.length_unit,
        oantenna1.frequency,
        oantenna1.frequency_unit,
    )
)
```

```
Arm length: 3.03mm at 12.0GHz
```

Create an empty HFSS design

Create an empty HFSS design.

[9]:
```
app = ansys.aedt.core.Hfss(project=project_name, version=aedt_version, non_graphical=non_
↪graphical)
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
PyAEDT INFO: Log on console is enabled.
PyAEDT INFO: Log on file C:\Users\ansys\AppData\Local\Temp\pyaedt_ansys_5b6f3baa-9f88-
↪4283-a028-dd43004ee663.log is enabled.
PyAEDT INFO: Log on AEDT is disabled.
PyAEDT INFO: Debug logger is disabled. PyAEDT methods will not be logged.
PyAEDT INFO: Launching PyAEDT with gRPC plugin.
PyAEDT INFO: New AEDT session is starting on gRPC port 52973.
PyAEDT INFO: Electronics Desktop started on gRPC port: 52973 after 6.136591911315918␣
↪seconds.
PyAEDT INFO: AEDT installation Path C:\Program Files\ANSYS Inc\v251\AnsysEM
PyAEDT INFO: Ansoft.ElectronicsDesktop.2025.1 version started with process ID 10900.
PyAEDT INFO: Project bowtie_example has been created.
```

(continues on next page)

(continued from previous page)

```
PyAEDT INFO: No design is present. Inserting a new design.
PyAEDT INFO: Added design 'HFSS_EHV' of type HFSS.
PyAEDT INFO: Aedt Objects correctly read
```

Create antenna in HFSS

Create antenna object, change frequency synthesis, create antenna, and set up in HFSS.

[10]:
```
oantenna1 = BowTieRounded(app)
```

```
PyAEDT INFO: Modeler class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: Materials class has been initialized! Elapsed time: 0m 0sec
```

[11]:
```
# Create antenna in HFSS.
oantenna1.model_hfss()
```

```
PyAEDT WARNING: Property Command is read-only.
PyAEDT INFO: Parsing design objects. This operation can take time
PyAEDT INFO: Refreshing bodies from Object Info
PyAEDT INFO: Bodies Info Refreshed Elapsed time: 0m 0sec
PyAEDT INFO: 3D Modeler objects parsed. Elapsed time: 0m 0sec
PyAEDT INFO: Union of 2 objects has been executed.
PyAEDT WARNING: Property Command is read-only.
PyAEDT WARNING: Property Command is read-only.
```

Create antenna setup.

[12]:
```
oantenna1.setup_hfss()
```

```
PyAEDT INFO: Boundary Perfect E PerfE_A0UH9Q has been created.
PyAEDT INFO: Boundary Perfect E PerfE_BQE26N has been created.
PyAEDT INFO: Boundary AutoIdentify port_Patch_LZH36B_1 has been created.
```

[12]:
```
True
```

Change default name.

[13]:
```
oantenna1.name = "MyAmazingAntenna"
```

Create antenna in HFSS

Create antenna object, change origin parameter in the antenna definition, create antenna, and set up in HFSS.

[14]:
```
oantenna2 = BowTieRounded(app, origin=[2, 5, 0], name="MyAntenna")
oantenna2.model_hfss()
oantenna2.setup_hfss()
```

```
PyAEDT WARNING: Property Command is read-only.
PyAEDT INFO: Parsing design objects. This operation can take time
PyAEDT INFO: Refreshing bodies from Object Info
PyAEDT INFO: Bodies Info Refreshed Elapsed time: 0m 0sec
PyAEDT INFO: 3D Modeler objects parsed. Elapsed time: 0m 0sec
PyAEDT INFO: Union of 2 objects has been executed.
PyAEDT WARNING: Property Command is read-only.
```

(continues on next page)

(continued from previous page)

```
PyAEDT WARNING: Property Command is read-only.
PyAEDT INFO: Boundary Perfect E PerfE_96RNIX has been created.
PyAEDT INFO: Boundary Perfect E PerfE_5PA3CD has been created.
PyAEDT INFO: Boundary AutoIdentify port_MyAntenna_1 has been created.
```

[14]:
```
True
```

Plot HFSS model

Plot geometry with PyVista.

[15]:
```
app.plot()
```

```
C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-antenna\.venv\Lib\site-
↪packages\ansys\aedt\core\visualization\plot\pyvista.py:56: UserWarning: Graphics␣
↪dependencies are required. Please install the ``graphics`` target to use this method.␣
↪You can install it by running `pip install pyaedt[graphics]` or `pip install␣
↪pyaedt[all]`.
  warnings.warn(ERROR_GRAPHICS_REQUIRED)
C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-antenna\.venv\Lib\site-
↪packages\ansys\aedt\core\visualization\plot\matplotlib.py:54: UserWarning: Graphics␣
↪dependencies are required. Please install the ``graphics`` target to use this method.␣
↪You can install it by running `pip install pyaedt[graphics]` or `pip install␣
↪pyaedt[all]`.
  warnings.warn(ERROR_GRAPHICS_REQUIRED)
PyAEDT INFO: Parsing C:/Users/ansys/AppData/Local/Temp/tmpftm85_ao_ansys/pyaedt_prj_6IR/
↪bowtie_example.aedt.
PyAEDT INFO: File C:/Users/ansys/AppData/Local/Temp/tmpftm85_ao_ansys/pyaedt_prj_6IR/
↪bowtie_example.aedt correctly loaded. Elapsed time: 0m 0sec
PyAEDT INFO: aedt file load time 0.0
PyAEDT INFO: PostProcessor class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: PostProcessor class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: Post class has been initialized! Elapsed time: 0m 0sec
PyAEDT ERROR: ************************************************************
PyAEDT ERROR:   File "<frozen runpy>", line 198, in _run_module_as_main
PyAEDT ERROR:   File "<frozen runpy>", line 88, in _run_code
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel_launcher.py", line 18, in <module>
PyAEDT ERROR:     app.launch_new_instance()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\traitlets\config\application.py", line 1075, in launch_
↪instance
PyAEDT ERROR:     app.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelapp.py", line 739, in start
PyAEDT ERROR:     self.io_loop.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\tornado\platform\asyncio.py", line 211, in start
PyAEDT ERROR:     self.asyncio_loop.run_forever()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 645, in run_forever
PyAEDT ERROR:     self._run_once()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
```

(continues on next page)

(continued from previous page)

```
↪events.py", line 1999, in _run_once
PyAEDT ERROR:     handle._run()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\
↪events.py", line 88, in _run
PyAEDT ERROR:     self._context.run(self._callback, *self._args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 545, in dispatch_queue
PyAEDT ERROR:     await self.process_one()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 534, in process_one
PyAEDT ERROR:     await dispatch(*args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 437, in dispatch_shell
PyAEDT ERROR:     await result
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 362, in execute_request
PyAEDT ERROR:     await super().execute_request(stream, ident, parent)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 778, in execute_request
PyAEDT ERROR:     reply_content = await reply_content
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 449, in do_execute
PyAEDT ERROR:     res = shell.run_cell(
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\zmqshell.py", line 549, in run_cell
PyAEDT ERROR:     return super().run_cell(*args, **kwargs)
PyAEDT ERROR:   File "C:\Users\ansys\AppData\Local\Temp\ipykernel_10232\123557834.py",␣
↪line 1, in <module>
PyAEDT ERROR:     app.plot()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\internal\checks.py", line 104, in␣
↪wrapper
PyAEDT ERROR:     return method(self, *args, **kwargs)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\application\analysis_3d.py", line 286,␣
↪in plot
PyAEDT ERROR:     return self.post.plot_model_obj(
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\post\post_common_3d.py",␣
↪line 2009, in plot_model_obj
PyAEDT ERROR:     model.plot()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\plot\pyvista.py", line␣
↪1370, in plot
PyAEDT ERROR:     self.populate_pyvista_object()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\plot\pyvista.py", line␣
↪1248, in populate_pyvista_object
PyAEDT ERROR:     self.pv = pv.Plotter(notebook=self.is_notebook, off_screen=self.off_
↪screen, window_size=self.windows_size)
PyAEDT ERROR:               ^^
PyAEDT ERROR: Name 'pv' is not defined on populate_pyvista_object
```

(continues on next page)

(continued from previous page)

```
PyAEDT ERROR: Last Electronics Desktop Message - [error]  script macro error: command is␣
↪read only. (07:51:52 am  may 22, 2025)
PyAEDT ERROR: ***************************************************************
PyAEDT ERROR: ***************************************************************
PyAEDT ERROR:   File "<frozen runpy>", line 198, in _run_module_as_main
PyAEDT ERROR:   File "<frozen runpy>", line 88, in _run_code
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel_launcher.py", line 18, in <module>
PyAEDT ERROR:     app.launch_new_instance()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\traitlets\config\application.py", line 1075, in launch_
↪instance
PyAEDT ERROR:     app.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelapp.py", line 739, in start
PyAEDT ERROR:     self.io_loop.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\tornado\platform\asyncio.py", line 211, in start
PyAEDT ERROR:     self.asyncio_loop.run_forever()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 645, in run_forever
PyAEDT ERROR:     self._run_once()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 1999, in _run_once
PyAEDT ERROR:     handle._run()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\
↪events.py", line 88, in _run
PyAEDT ERROR:     self._context.run(self._callback, *self._args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 545, in dispatch_queue
PyAEDT ERROR:     await self.process_one()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 534, in process_one
PyAEDT ERROR:     await dispatch(*args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 437, in dispatch_shell
PyAEDT ERROR:     await result
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 362, in execute_request
PyAEDT ERROR:     await super().execute_request(stream, ident, parent)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 778, in execute_request
PyAEDT ERROR:     reply_content = await reply_content
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 449, in do_execute
PyAEDT ERROR:     res = shell.run_cell(
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\zmqshell.py", line 549, in run_cell
PyAEDT ERROR:     return super().run_cell(*args, **kwargs)
PyAEDT ERROR:   File "C:\Users\ansys\AppData\Local\Temp\ipykernel_10232\123557834.py",␣
↪line 1, in <module>
PyAEDT ERROR:     app.plot()
```

(continues on next page)

(continued from previous page)

```
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\internal\checks.py", line 104, in␣
↪wrapper
PyAEDT ERROR:     return method(self, *args, **kwargs)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\application\analysis_3d.py", line 286,␣
↪in plot
PyAEDT ERROR:     return self.post.plot_model_obj(
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\post\post_common_3d.py",␣
↪line 2009, in plot_model_obj
PyAEDT ERROR:     model.plot()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\plot\pyvista.py", line␣
↪1385, in plot
PyAEDT ERROR:     self.pv.add_key_event("s", s_callback)
PyAEDT ERROR:     ^^^^^^^^^^^^^^^^^^^^
PyAEDT ERROR: 'nonetype' object has no attribute 'add_key_event' on plot
PyAEDT ERROR: Last Electronics Desktop Message - [error]  script macro error: command is␣
↪read only. (07:51:52 am  may 22, 2025)
PyAEDT ERROR: ***************************************************************
```

[15]:
```
<ansys.aedt.core.visualization.plot.pyvista.ModelPlotter at 0x2f2e1f643a0>
```

Release AEDT

Release AEDT.

[16]:
```
app.release_desktop(True, True)
```

```
PyAEDT INFO: Desktop has been released and closed.
```

[16]:
```
True
```

Clean temporary directory

[17]:
```
temp_dir.cleanup()
```

4.2 Antenna toolkit

These examples show how to use the `ToolkitBackend` class:

4.2.1 Antenna toolkit example

This example demonstrates how to use the `ToolkitBackend` class. It initiates AEDT through PyAEDT, sets up an empty HFSS design, and proceeds to create the antenna.

Perform required imports

[1]:
```
import sys
import tempfile
```

[2]:
```
from ansys.aedt.core import generate_unique_project_name
from ansys.aedt.core.visualization.advanced.farfield_visualization import FfdSolutionData
```

```
C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-antenna\.venv\Lib\site-
↪packages\ansys\aedt\core\visualization\plot\matplotlib.py:54: UserWarning: Graphics␣
↪dependencies are required. Please install the ``graphics`` target to use this method.␣
↪You can install it by running `pip install pyaedt[graphics]` or `pip install␣
↪pyaedt[all]`.
  warnings.warn(ERROR_GRAPHICS_REQUIRED)
C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-antenna\.venv\Lib\site-
↪packages\ansys\aedt\core\visualization\plot\pyvista.py:56: UserWarning: Graphics␣
↪dependencies are required. Please install the ``graphics`` target to use this method.␣
↪You can install it by running `pip install pyaedt[graphics]` or `pip install␣
↪pyaedt[all]`.
  warnings.warn(ERROR_GRAPHICS_REQUIRED)
```

[3]:
```
from ansys.aedt.toolkits.antenna.backend.api import ToolkitBackend
from ansys.aedt.toolkits.antenna.backend.models import properties
```

Set AEDT version

Set AEDT version.

[4]:
```
aedt_version = "2025.1"
```

Set non-graphical mode

Set non-graphical mode.

[5]:
```
non_graphical = True
```

Create temporary directory

[6]:
```
temp_dir = tempfile.TemporaryDirectory(suffix="_ansys")
project_name = generate_unique_project_name(root_name=temp_dir.name, project_name=
↪"antenna_toolkit")
```

Set default properties

[7]:
```
properties.aedt_version = aedt_version
properties.non_graphical = non_graphical
properties.active_project = project_name
```

Initialize toolkit

Initialize the toolkit.

[8]:
```
toolkit_api = ToolkitBackend()
```

Get available_antennas

[9]:
```
print(toolkit_api.available_antennas)
```

```
['BowTieNormal', 'BowTieRounded', 'BowTieSlot', 'Archimedean', 'Log', 'Sinuous',
↪'AxialMode', 'Conical', 'Corrugated', 'EPlane', 'Elliptical', 'HPlane', 'Pyramidal',
↪'PyramidalRidged', 'QuadRidged', 'RectangularPatchEdge', 'RectangularPatchInset',
↪'RectangularPatchProbe']
```

Get default properties

[10]:
```
backend_properties = toolkit_api.get_properties()
frequency = backend_properties["antenna"]["synthesis"]["frequency"]
frequency_units = backend_properties["antenna"]["synthesis"]["frequency_unit"]
length_unit = backend_properties["antenna"]["synthesis"]["length_unit"]
```

Modify default length units

[11]:
```
properties.antenna.synthesis.length_unit = "cm"
```

Create antenna object only for synthesis

Create antenna object.

[12]:
```
antenna_parameters_1 = toolkit_api.get_antenna("RectangularPatchProbe", synth_only=True)
```

```
INFO - AEDT is released.
```

[13]:
```
print(
    "Patch X length: {}{} at {}{}".format(
        str(antenna_parameters_1["patch_x"]),
        length_unit,
        frequency,
        frequency_units,
    )
)
```

```
Patch X length: 0.912871meter at 10.0GHz
```

Change synthesis frequency

Modify resonance frequency and modify parameters with the `set_properties()` method.

[14]:
```
new_frequency1 = 12.0
new_properties = {"frequency": new_frequency1}
toolkit_api.set_properties(new_properties)
```

```
INFO - Updating internal properties.
DEBUG - Updating 'frequency' with value 12.0
DEBUG - Properties were updated successfully.
```

[14]:
```
(True, 'Properties were updated successfully.')
```

[15]:
```
antenna_parameters_2 = toolkit_api.get_antenna("RectangularPatchProbe", synth_only=True)
```

```
INFO - AEDT is released.
```

[16]:
```
print(
    "Patch X length: {}{} at {}{}".format(
        str(antenna_parameters_2["patch_x"]),
        length_unit,
        new_frequency1,
        frequency_units,
    )
)
```

```
Patch X length: 0.760726meter at 12.0GHz
```

Change synthesis frequency

Modify resonance frequency with properties directly.

[17]:
```
new_frequency2 = 15.0
properties.antenna.synthesis.frequency = new_frequency2
```

[18]:
```
antenna_parameters_3 = toolkit_api.get_antenna("RectangularPatchProbe", synth_only=True)
```

```
INFO - AEDT is released.
```

[19]:
```
print(
    "Patch X length: {}{} at {}{}".format(
        str(antenna_parameters_3["patch_x"]),
        length_unit,
        new_frequency2,
        frequency_units,
    )
)
```

```
Patch X length: 0.608581meter at 15.0GHz
```

Initialize AEDT

Launch a new AEDT session in a thread.

[20]:
```
thread_msg = toolkit_api.launch_thread(toolkit_api.launch_aedt)
```

```
DEBUG - Starting thread: Toolkit_Thread
DEBUG - Toolkit is not connected to AEDT.
DEBUG - Launching AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
```

Wait for the toolkit thread to be idle

Wait for the toolkit thread to be idle and ready to accept a new task.

[21]:
```
idle = toolkit_api.wait_to_be_idle()
if not idle:
    print("AEDT not initialized.")
    sys.exit()
```

```
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is busy and processing a task.
DEBUG - Toolkit is idle and ready to accept a new task.
```

Connect to HFSS design

Create an HFSS design.

[22]:
```
toolkit_api.connect_design("HFSS")
```

```
DEBUG - Toolkit is not connected to AEDT.
DEBUG - Connecting AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
PyAEDT INFO: Log on console is enabled.
PyAEDT INFO: Log on file C:\Users\ansys\AppData\Local\Temp\pyaedt_ansys_ca281d9d-94de-
↪445a-8866-6ce9242a93c4.log is enabled.
PyAEDT INFO: Log on AEDT is disabled.
PyAEDT INFO: Debug logger is disabled. PyAEDT methods will not be logged.
PyAEDT INFO: Launching PyAEDT with gRPC plugin.
PyAEDT INFO: Connecting to AEDT session on gRPC port 52533.
PyAEDT INFO: AEDT installation Path C:\Program Files\ANSYS Inc\v251\AnsysEM
```

```
DEBUG - Toolkit is connected to AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Returning found Desktop session with PID 9544!
PyAEDT INFO: Project antenna_toolkit has been created.
PyAEDT INFO: Added design 'HFSS_GO01VV' of type HFSS.
```

(continues on next page)

(continued from previous page)

```
PyAEDT INFO: Aedt Objects correctly read
PyAEDT INFO: Project antenna_toolkit Saved correctly
PyAEDT INFO: Active Design set to HFSS_GO01VV
```

```
DEBUG - Project name: antenna_toolkit
INFO - Updating internal properties.
DEBUG - Updating 'project_list' with value ['C:/Users/ansys/AppData/Local/Temp/
↪tmpbur7r48h_ansys/pyaedt_prj_NP9/antenna_toolkit.aedt']
DEBUG - Updating 'active_design' with value HFSS_GO01VV
DEBUG - Updating 'active_project' with value C:/Users/ansys/AppData/Local/Temp/
↪tmpbur7r48h_ansys/pyaedt_prj_NP9/antenna_toolkit.aedt
DEBUG - Updating 'design_list' with value {'antenna_toolkit': ['HFSS_GO01VV']}
DEBUG - Properties were updated successfully.
INFO - Toolkit is connected to AEDT design.
```

[22]:
```
True
```

Create setup when antenna is created

Set `create_setup` property.

[23]:
```
properties.antenna.setup.create_setup = True
properties.antenna.synthesis.outer_boundary = "Radiation"
```

Create antenna in HFSS

Create antenna and set up in HFSS.

[24]:
```
antenna_parameter = toolkit_api.get_antenna("RectangularPatchProbe")
```

```
PyAEDT INFO: Modeler class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: Materials class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: Open Region correctly created.
PyAEDT INFO: Project antenna_toolkit Saved correctly
PyAEDT WARNING: Property Command is read-only.
PyAEDT WARNING: Property Command is read-only.
PyAEDT WARNING: Property Command is read-only.
PyAEDT WARNING: Argument `cs_plane` is deprecated for method `create_circle`; use␣
↪`orientation` instead.
PyAEDT WARNING: Property Command is read-only.
PyAEDT WARNING: Property Command is read-only.
PyAEDT WARNING: Property Command is read-only.
PyAEDT WARNING: Property Command is read-only.
PyAEDT WARNING: Argument `cs_plane` is deprecated for method `create_circle`; use␣
↪`orientation` instead.
PyAEDT WARNING: Property Command is read-only.
PyAEDT INFO: Parsing design objects. This operation can take time
PyAEDT INFO: Refreshing bodies from Object Info
PyAEDT INFO: Bodies Info Refreshed Elapsed time: 0m 0sec
PyAEDT INFO: 3D Modeler objects parsed. Elapsed time: 0m 0sec
PyAEDT INFO: Boundary Perfect E PerfE_YL75JT has been created.
PyAEDT INFO: Boundary Perfect E PerfE_F2ILTQ has been created.
PyAEDT INFO: Boundary Perfect E PerfE_1VITIN has been created.
```

(continues on next page)

(continued from previous page)

```
PyAEDT INFO: Boundary AutoIdentify port_Patch_1Y2UOL_1 has been created.
PyAEDT INFO: Parsing C:/Users/ansys/AppData/Local/Temp/tmpbur7r48h_ansys/pyaedt_prj_NP9/
↪antenna_toolkit.aedt.
PyAEDT INFO: File C:/Users/ansys/AppData/Local/Temp/tmpbur7r48h_ansys/pyaedt_prj_NP9/
↪antenna_toolkit.aedt correctly loaded. Elapsed time: 0m 0sec
PyAEDT INFO: aedt file load time 0.015642166137695312
PyAEDT INFO: Project antenna_toolkit Saved correctly
PyAEDT INFO: Desktop has been released.
```

```
INFO - AEDT is released.
```

Try. to create antenna

The AEDT Antenna Toolkit API does not allow the creation of more than one antenna. However, you can use the antenna models API to create more than one antenna.

[25]:
```
new_antenna = toolkit_api.get_antenna("BowTie")
```

```
DEBUG - Antenna is already created.
```

[26]:
```
print(new_antenna)
```

```
False
```

Set properties

Move antenna X position

[27]:
```
toolkit_api.update_hfss_parameters("pos_x", "20")
```

```
DEBUG - Toolkit is not connected to AEDT.
DEBUG - Connecting AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
PyAEDT INFO: Log on console is enabled.
PyAEDT INFO: Log on file C:\Users\ansys\AppData\Local\Temp\pyaedt_ansys_ca281d9d-94de-
↪445a-8866-6ce9242a93c4.log is enabled.
PyAEDT INFO: Log on AEDT is disabled.
PyAEDT INFO: Debug logger is disabled. PyAEDT methods will not be logged.
PyAEDT INFO: Launching PyAEDT with gRPC plugin.
PyAEDT INFO: Connecting to AEDT session on gRPC port 52533.
PyAEDT INFO: AEDT installation Path C:\Program Files\ANSYS Inc\v251\AnsysEM
```

```
DEBUG - Toolkit is connected to AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Returning found Desktop session with PID 9544!
PyAEDT INFO: Project antenna_toolkit set to active.
PyAEDT INFO: Active Design set to HFSS_GO01VV
PyAEDT INFO: Aedt Objects correctly read
```

```
INFO - Toolkit is connected to AEDT design.
PyAEDT INFO: Desktop has been released.
INFO - AEDT is released.
[27]: True
```

Fit all

```
[28]: toolkit_api.connect_design()
DEBUG - Toolkit is not connected to AEDT.
DEBUG - Connecting AEDT.
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
PyAEDT INFO: Log on console is enabled.
PyAEDT INFO: Log on file C:\Users\ansys\AppData\Local\Temp\pyaedt_ansys_ca281d9d-94de-
↪445a-8866-6ce9242a93c4.log is enabled.
PyAEDT INFO: Log on AEDT is disabled.
PyAEDT INFO: Debug logger is disabled. PyAEDT methods will not be logged.
PyAEDT INFO: Launching PyAEDT with gRPC plugin.
PyAEDT INFO: Connecting to AEDT session on gRPC port 52533.
PyAEDT INFO: AEDT installation Path C:\Program Files\ANSYS Inc\v251\AnsysEM
DEBUG - Toolkit is connected to AEDT.
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Returning found Desktop session with PID 9544!
PyAEDT INFO: Project antenna_toolkit set to active.
PyAEDT INFO: Active Design set to HFSS_GO01VV
PyAEDT INFO: Aedt Objects correctly read
INFO - Toolkit is connected to AEDT design.
[28]: True
```

```
[29]: toolkit_api.aedtapp.modeler.fit_all()
PyAEDT INFO: Modeler class has been initialized! Elapsed time: 0m 0sec
```

```
[30]: toolkit_api.release_aedt(False, False)
PyAEDT INFO: Desktop has been released.
INFO - AEDT is released.
[30]: True
```

Set properties

Move antenna X position to origin

[31]:
```
toolkit_api.update_hfss_parameters("pos_x", "0")
```

```
DEBUG - Toolkit is not connected to AEDT.
DEBUG - Connecting AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
PyAEDT INFO: Log on console is enabled.
PyAEDT INFO: Log on file C:\Users\ansys\AppData\Local\Temp\pyaedt_ansys_ca281d9d-94de-
↪445a-8866-6ce9242a93c4.log is enabled.
PyAEDT INFO: Log on AEDT is disabled.
PyAEDT INFO: Debug logger is disabled. PyAEDT methods will not be logged.
PyAEDT INFO: Launching PyAEDT with gRPC plugin.
PyAEDT INFO: Connecting to AEDT session on gRPC port 52533.
PyAEDT INFO: AEDT installation Path C:\Program Files\ANSYS Inc\v251\AnsysEM
```

```
DEBUG - Toolkit is connected to AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Returning found Desktop session with PID 9544!
PyAEDT INFO: Project antenna_toolkit set to active.
PyAEDT INFO: Active Design set to HFSS_GO01VV
PyAEDT INFO: Aedt Objects correctly read
```

```
INFO - Toolkit is connected to AEDT design.
```

```
PyAEDT INFO: Desktop has been released.
```

```
INFO - AEDT is released.
```

[31]:
```
True
```

Analyze design in batch mode

[32]:
```
toolkit_api.analyze()
```

```
DEBUG - Toolkit is not connected to AEDT.
DEBUG - Connecting AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
PyAEDT INFO: Log on console is enabled.
PyAEDT INFO: Log on file C:\Users\ansys\AppData\Local\Temp\pyaedt_ansys_ca281d9d-94de-
↪445a-8866-6ce9242a93c4.log is enabled.
PyAEDT INFO: Log on AEDT is disabled.
PyAEDT INFO: Debug logger is disabled. PyAEDT methods will not be logged.
PyAEDT INFO: Launching PyAEDT with gRPC plugin.
PyAEDT INFO: Connecting to AEDT session on gRPC port 52533.
PyAEDT INFO: AEDT installation Path C:\Program Files\ANSYS Inc\v251\AnsysEM
```

```
DEBUG - Toolkit is connected to AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Returning found Desktop session with PID 9544!
PyAEDT INFO: Project antenna_toolkit set to active.
PyAEDT INFO: Active Design set to HFSS_GO01VV
PyAEDT INFO: Aedt Objects correctly read
```

```
INFO - Toolkit is connected to AEDT design.
```

```
PyAEDT INFO: Project antenna_toolkit Saved correctly
PyAEDT INFO: Key Desktop/ActiveDSOConfigurations/HFSS correctly changed.
PyAEDT INFO: Solving all design setups.
PyAEDT INFO: Key Desktop/ActiveDSOConfigurations/HFSS correctly changed.
PyAEDT INFO: Design setup None solved correctly in 0.0h 1.0m 11.0s
PyAEDT INFO: Desktop has been released.
```

```
INFO - AEDT is released.
```

[32]:
```
True
```

Get scattering results

[33]:
```
scattering_data = toolkit_api.scattering_results()
```

```
DEBUG - Toolkit is not connected to AEDT.
DEBUG - Connecting AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
PyAEDT INFO: Log on console is enabled.
PyAEDT INFO: Log on file C:\Users\ansys\AppData\Local\Temp\pyaedt_ansys_ca281d9d-94de-
↪445a-8866-6ce9242a93c4.log is enabled.
PyAEDT INFO: Log on AEDT is disabled.
PyAEDT INFO: Debug logger is disabled. PyAEDT methods will not be logged.
PyAEDT INFO: Launching PyAEDT with gRPC plugin.
PyAEDT INFO: Connecting to AEDT session on gRPC port 52533.
PyAEDT INFO: AEDT installation Path C:\Program Files\ANSYS Inc\v251\AnsysEM
```

```
DEBUG - Toolkit is connected to AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Returning found Desktop session with PID 9544!
PyAEDT INFO: Project antenna_toolkit set to active.
PyAEDT INFO: Active Design set to HFSS_GO01VV
PyAEDT INFO: Aedt Objects correctly read
```

```
INFO - Toolkit is connected to AEDT design.
```

```
PyAEDT INFO: Parsing C:/Users/ansys/AppData/Local/Temp/tmpbur7r48h_ansys/pyaedt_prj_NP9/
↪antenna_toolkit.aedt.
PyAEDT INFO: File C:/Users/ansys/AppData/Local/Temp/tmpbur7r48h_ansys/pyaedt_prj_NP9/
↪antenna_toolkit.aedt correctly loaded. Elapsed time: 0m 0sec
```

(continues on next page)

(continued from previous page)

```
PyAEDT INFO: aedt file load time 0.031255483627319336
PyAEDT INFO: PostProcessor class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: PostProcessor class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: Post class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: Solution Data Correctly Loaded.
PyAEDT INFO: Desktop has been released.
```

```
INFO - AEDT is released.
```

Get farfield results

[34]:
```
frequency_str = [str(properties.antenna.synthesis.frequency) + properties.antenna.
↪synthesis.frequency_unit]
farfield_metadata, farfield_frequency = toolkit_api.export_farfield(
    frequencies=frequency_str, sphere="3D", encode=False
)
```

```
DEBUG - Toolkit is not connected to AEDT.
DEBUG - Connecting AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
PyAEDT INFO: Log on console is enabled.
PyAEDT INFO: Log on file C:\Users\ansys\AppData\Local\Temp\pyaedt_ansys_ca281d9d-94de-
↪445a-8866-6ce9242a93c4.log is enabled.
PyAEDT INFO: Log on AEDT is disabled.
PyAEDT INFO: Debug logger is disabled. PyAEDT methods will not be logged.
PyAEDT INFO: Launching PyAEDT with gRPC plugin.
PyAEDT INFO: Connecting to AEDT session on gRPC port 52533.
PyAEDT INFO: AEDT installation Path C:\Program Files\ANSYS Inc\v251\AnsysEM
```

```
DEBUG - Toolkit is connected to AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Returning found Desktop session with PID 9544!
PyAEDT INFO: Project antenna_toolkit set to active.
PyAEDT INFO: Active Design set to HFSS_GO01VV
PyAEDT INFO: Aedt Objects correctly read
```

```
INFO - Toolkit is connected to AEDT design.
```

```
PyAEDT INFO: Project antenna_toolkit Saved correctly
PyAEDT INFO: Far field sphere 3D is assigned
PyAEDT INFO: Exporting antenna metadata...
PyAEDT INFO: Parsing C:/Users/ansys/AppData/Local/Temp/tmpbur7r48h_ansys/pyaedt_prj_NP9/
↪antenna_toolkit.aedt.
PyAEDT INFO: File C:/Users/ansys/AppData/Local/Temp/tmpbur7r48h_ansys/pyaedt_prj_NP9/
↪antenna_toolkit.aedt correctly loaded. Elapsed time: 0m 0sec
PyAEDT INFO: aedt file load time 0.03127431869506836
PyAEDT INFO: PostProcessor class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: PostProcessor class has been initialized! Elapsed time: 0m 0sec
```

(continues on next page)

(continued from previous page)

```
PyAEDT INFO: Post class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: Antenna metadata exported.
PyAEDT INFO: Exporting geometry...
PyAEDT INFO: Modeler class has been initialized! Elapsed time: 0m 0sec
PyAEDT ERROR: ************************************************************
PyAEDT ERROR:   File "<frozen runpy>", line 198, in _run_module_as_main
PyAEDT ERROR:   File "<frozen runpy>", line 88, in _run_code
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel_launcher.py", line 18, in <module>
PyAEDT ERROR:     app.launch_new_instance()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\traitlets\config\application.py", line 1075, in launch_
↪instance
PyAEDT ERROR:     app.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelapp.py", line 739, in start
PyAEDT ERROR:     self.io_loop.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\tornado\platform\asyncio.py", line 211, in start
PyAEDT ERROR:     self.asyncio_loop.run_forever()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 645, in run_forever
PyAEDT ERROR:     self._run_once()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 1999, in _run_once
PyAEDT ERROR:     handle._run()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\
↪events.py", line 88, in _run
PyAEDT ERROR:     self._context.run(self._callback, *self._args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 545, in dispatch_queue
PyAEDT ERROR:     await self.process_one()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 534, in process_one
PyAEDT ERROR:     await dispatch(*args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 437, in dispatch_shell
PyAEDT ERROR:     await result
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 362, in execute_request
PyAEDT ERROR:     await super().execute_request(stream, ident, parent)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 778, in execute_request
PyAEDT ERROR:     reply_content = await reply_content
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 449, in do_execute
PyAEDT ERROR:     res = shell.run_cell(
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\zmqshell.py", line 549, in run_cell
PyAEDT ERROR:     return super().run_cell(*args, **kwargs)
PyAEDT ERROR:   File "C:\Users\ansys\AppData\Local\Temp\ipykernel_8412\1978182973.py",␣
↪line 2, in <module>
```

(continues on next page)

(continued from previous page)

```
PyAEDT ERROR:     farfield_metadata, farfield_frequency = toolkit_api.export_farfield(
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\toolkits\antenna\backend\api.py", line 336,␣
↪in export_farfield
PyAEDT ERROR:     farfield_exporter = self.aedtapp.get_antenna_data(
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\hfss.py", line 6358, in get_antenna_
↪data
PyAEDT ERROR:     metadata_file = ffd.export_farfield()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\post\farfield_exporter.py
↪", line 223, in export_farfield
PyAEDT ERROR:     obj_list = self.__create_geometries(geometry_path)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\post\farfield_exporter.py
↪", line 338, in __create_geometries
PyAEDT ERROR:     model_pv = self.__app.post.get_model_plotter_geometries(plot_air_
↪objects=False)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\post\post_common_3d.py",␣
↪line 1927, in get_model_plotter_geometries
PyAEDT ERROR:     model.generate_geometry_mesh()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\plot\pyvista.py", line␣
↪1619, in generate_geometry_mesh
PyAEDT ERROR:     self.pv = pv.Plotter(notebook=self.is_notebook, window_size=self.
↪windows_size)
PyAEDT ERROR:               ^^
PyAEDT ERROR: Name 'pv' is not defined on generate_geometry_mesh
PyAEDT ERROR: Last Electronics Desktop Message - [error]  script macro error: command is␣
↪read only. (07:49:34 am  may 22, 2025)
PyAEDT ERROR: ***************************************************************
PyAEDT INFO: Exporting embedded element patterns... Done: 0.9561047554016113 seconds
PyAEDT INFO: Desktop has been released.
```

```
INFO - AEDT is released.
```

Get antenna model

[35]:
```
files = toolkit_api.export_aedt_model(encode=False)
```

```
DEBUG - Toolkit is not connected to AEDT.
DEBUG - Connecting AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
PyAEDT INFO: Log on console is enabled.
PyAEDT INFO: Log on file C:\Users\ansys\AppData\Local\Temp\pyaedt_ansys_ca281d9d-94de-
↪445a-8866-6ce9242a93c4.log is enabled.
PyAEDT INFO: Log on AEDT is disabled.
PyAEDT INFO: Debug logger is disabled. PyAEDT methods will not be logged.
```

(continues on next page)

(continued from previous page)

```
PyAEDT INFO: Launching PyAEDT with gRPC plugin.
PyAEDT INFO: Connecting to AEDT session on gRPC port 52533.
PyAEDT INFO: AEDT installation Path C:\Program Files\ANSYS Inc\v251\AnsysEM
```

```
DEBUG - Toolkit is connected to AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Returning found Desktop session with PID 9544!
PyAEDT INFO: Project antenna_toolkit set to active.
PyAEDT INFO: Active Design set to HFSS_GO01VV
PyAEDT INFO: Aedt Objects correctly read
```

```
INFO - Toolkit is connected to AEDT design.
```

```
PyAEDT INFO: Project antenna_toolkit Saved correctly
PyAEDT INFO: Parsing C:/Users/ansys/AppData/Local/Temp/tmpbur7r48h_ansys/pyaedt_prj_NP9/
↪antenna_toolkit.aedt.
PyAEDT INFO: File C:/Users/ansys/AppData/Local/Temp/tmpbur7r48h_ansys/pyaedt_prj_NP9/
↪antenna_toolkit.aedt correctly loaded. Elapsed time: 0m 0sec
PyAEDT INFO: aedt file load time 0.05015277862548828
PyAEDT INFO: PostProcessor class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: PostProcessor class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: Post class has been initialized! Elapsed time: 0m 0sec
PyAEDT WARNING: Argument `export_as_single_objects` is deprecated for method `export_
↪model_obj`; use `export_as_multiple_objects` instead.
PyAEDT INFO: Modeler class has been initialized! Elapsed time: 0m 0sec
PyAEDT INFO: Desktop has been released.
```

```
INFO - AEDT is released.
```

Release AEDT

Release AEDT.

[36]:
```
toolkit_api.release_aedt(True, True)
```

```
DEBUG - Toolkit is not connected to AEDT.
DEBUG - Connecting AEDT.
```

```
PyAEDT INFO: Python version 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC␣
↪v.1943 64 bit (AMD64)].
PyAEDT INFO: PyAEDT version 0.16.2.
PyAEDT INFO: Initializing new Desktop session.
PyAEDT INFO: Log on console is enabled.
PyAEDT INFO: Log on file C:\Users\ansys\AppData\Local\Temp\pyaedt_ansys_ca281d9d-94de-
↪445a-8866-6ce9242a93c4.log is enabled.
PyAEDT INFO: Log on AEDT is disabled.
PyAEDT INFO: Debug logger is disabled. PyAEDT methods will not be logged.
PyAEDT INFO: Launching PyAEDT with gRPC plugin.
PyAEDT INFO: Connecting to AEDT session on gRPC port 52533.
PyAEDT INFO: AEDT installation Path C:\Program Files\ANSYS Inc\v251\AnsysEM
```

```
DEBUG - Toolkit is connected to AEDT.
```

```
PyAEDT INFO: Desktop has been released and closed.
INFO - AEDT is released.
[36]: True
```

Plot results

Plot exported files

[37]:
```
from ansys.aedt.core.visualization.plot.pyvista import ModelPlotter
```

[38]:
```
model = ModelPlotter()
for file in files:
    model.add_object(file[0], file[1], file[2])
```

[39]:
```
model.plot(show=False)
```

```
PyAEDT ERROR: ***************************************************************
PyAEDT ERROR:   File "<frozen runpy>", line 198, in _run_module_as_main
PyAEDT ERROR:   File "<frozen runpy>", line 88, in _run_code
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel_launcher.py", line 18, in <module>
PyAEDT ERROR:     app.launch_new_instance()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\traitlets\config\application.py", line 1075, in launch_
↪instance
PyAEDT ERROR:     app.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelapp.py", line 739, in start
PyAEDT ERROR:     self.io_loop.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\tornado\platform\asyncio.py", line 211, in start
PyAEDT ERROR:     self.asyncio_loop.run_forever()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 645, in run_forever
PyAEDT ERROR:     self._run_once()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 1999, in _run_once
PyAEDT ERROR:     handle._run()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\
↪events.py", line 88, in _run
PyAEDT ERROR:     self._context.run(self._callback, *self._args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 545, in dispatch_queue
PyAEDT ERROR:     await self.process_one()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 534, in process_one
PyAEDT ERROR:     await dispatch(*args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 437, in dispatch_shell
PyAEDT ERROR:     await result
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 362, in execute_request
```

(continues on next page)

(continued from previous page)

```
PyAEDT ERROR:     await super().execute_request(stream, ident, parent)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 778, in execute_request
PyAEDT ERROR:     reply_content = await reply_content
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 449, in do_execute
PyAEDT ERROR:     res = shell.run_cell(
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\zmqshell.py", line 549, in run_cell
PyAEDT ERROR:     return super().run_cell(*args, **kwargs)
PyAEDT ERROR:   File "C:\Users\ansys\AppData\Local\Temp\ipykernel_8412\1893892073.py",␣
↪line 1, in <module>
PyAEDT ERROR:     model.plot(show=False)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\plot\pyvista.py", line␣
↪1370, in plot
PyAEDT ERROR:     self.populate_pyvista_object()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\plot\pyvista.py", line␣
↪1248, in populate_pyvista_object
PyAEDT ERROR:     self.pv = pv.Plotter(notebook=self.is_notebook, off_screen=self.off_
↪screen, window_size=self.windows_size)
PyAEDT ERROR:               ^^
PyAEDT ERROR: Name 'pv' is not defined on populate_pyvista_object
PyAEDT ERROR: ************************************************************
PyAEDT ERROR: ************************************************************
PyAEDT ERROR:   File "<frozen runpy>", line 198, in _run_module_as_main
PyAEDT ERROR:   File "<frozen runpy>", line 88, in _run_code
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel_launcher.py", line 18, in <module>
PyAEDT ERROR:     app.launch_new_instance()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\traitlets\config\application.py", line 1075, in launch_
↪instance
PyAEDT ERROR:     app.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelapp.py", line 739, in start
PyAEDT ERROR:     self.io_loop.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\tornado\platform\asyncio.py", line 211, in start
PyAEDT ERROR:     self.asyncio_loop.run_forever()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 645, in run_forever
PyAEDT ERROR:     self._run_once()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 1999, in _run_once
PyAEDT ERROR:     handle._run()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\
↪events.py", line 88, in _run
PyAEDT ERROR:     self._context.run(self._callback, *self._args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 545, in dispatch_queue
```

(continues on next page)

(continued from previous page)

```
PyAEDT ERROR:     await self.process_one()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 534, in process_one
PyAEDT ERROR:     await dispatch(*args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 437, in dispatch_shell
PyAEDT ERROR:     await result
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 362, in execute_request
PyAEDT ERROR:     await super().execute_request(stream, ident, parent)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 778, in execute_request
PyAEDT ERROR:     reply_content = await reply_content
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 449, in do_execute
PyAEDT ERROR:     res = shell.run_cell(
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\zmqshell.py", line 549, in run_cell
PyAEDT ERROR:     return super().run_cell(*args, **kwargs)
PyAEDT ERROR:   File "C:\Users\ansys\AppData\Local\Temp\ipykernel_8412\1893892073.py",␣
↪line 1, in <module>
PyAEDT ERROR:     model.plot(show=False)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\visualization\plot\pyvista.py", line␣
↪1385, in plot
PyAEDT ERROR:     self.pv.add_key_event("s", s_callback)
PyAEDT ERROR:     ^^^^^^^^^^^^^^^^^^^
PyAEDT ERROR: 'nonetype' object has no attribute 'add_key_event' on plot
PyAEDT ERROR: ************************************************************
```

[39]:
```
False
```

Load far field

[40]:
```
farfield_data = FfdSolutionData(farfield_metadata)
```

Plot far field

[41]:
```
data = farfield_data.plot_3d(show=False)
```

```
PyAEDT ERROR: ************************************************************
PyAEDT ERROR:   File "<frozen runpy>", line 198, in _run_module_as_main
PyAEDT ERROR:   File "<frozen runpy>", line 88, in _run_code
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel_launcher.py", line 18, in <module>
PyAEDT ERROR:     app.launch_new_instance()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\traitlets\config\application.py", line 1075, in launch_
↪instance
PyAEDT ERROR:     app.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelapp.py", line 739, in start
```

(continues on next page)

(continued from previous page)

```
PyAEDT ERROR:     self.io_loop.start()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\tornado\platform\asyncio.py", line 211, in start
PyAEDT ERROR:     self.asyncio_loop.run_forever()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 645, in run_forever
PyAEDT ERROR:     self._run_once()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\base_
↪events.py", line 1999, in _run_once
PyAEDT ERROR:     handle._run()
PyAEDT ERROR:   File "C:\actions-runner\_work\_tool\Python\3.12.10\x64\Lib\asyncio\
↪events.py", line 88, in _run
PyAEDT ERROR:     self._context.run(self._callback, *self._args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 545, in dispatch_queue
PyAEDT ERROR:     await self.process_one()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 534, in process_one
PyAEDT ERROR:     await dispatch(*args)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 437, in dispatch_shell
PyAEDT ERROR:     await result
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 362, in execute_request
PyAEDT ERROR:     await super().execute_request(stream, ident, parent)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\kernelbase.py", line 778, in execute_request
PyAEDT ERROR:     reply_content = await reply_content
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\ipkernel.py", line 449, in do_execute
PyAEDT ERROR:     res = shell.run_cell(
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ipykernel\zmqshell.py", line 549, in run_cell
PyAEDT ERROR:     return super().run_cell(*args, **kwargs)
PyAEDT ERROR:   File "C:\Users\ansys\AppData\Local\Temp\ipykernel_8412\2321922880.py",␣
↪line 1, in <module>
PyAEDT ERROR:     data = farfield_data.plot_3d(show=False)
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\internal\checks.py", line 157, in aedt_
↪graphics_decorator
PyAEDT ERROR:     check_graphics_available()
PyAEDT ERROR:   File "C:\actions-runner\_work\pyaedt-toolkits-antenna\pyaedt-toolkits-
↪antenna\.venv\Lib\site-packages\ansys\aedt\core\internal\checks.py", line 139, in␣
↪check_graphics_available
PyAEDT ERROR:     raise ImportError(ERROR_GRAPHICS_REQUIRED)
PyAEDT ERROR: Graphics dependencies are required. please install the ``graphics`` target␣
↪to use this method. you can install it by running `pip install pyaedt[graphics]` or␣
↪`pip install pyaedt[all]`. on aedt_graphics_decorator
PyAEDT ERROR: ************************************************************
```

Clean temporary directory

[42]:
```
temp_dir.cleanup()
```

CHAPTER

FIVE

CONTRIBUTE

Overall guidance on contributing to a PyAnsys repository appears in Contributing in the *PyAnsys Developers Guide*. Ensure that you are thoroughly familiar with this guide before attempting to contribute to PyAEDT or its toolkits.

The following contribution information is specific to PyAEDT toolkits.

5.1 Clone the repository

To clone and install the latest version of this toolkit in development mode, run:

```
git clone https://github.com/ansys/pyaedt-toolkits-antenna.git
cd pyaedt-toolkits-antenna
python -m pip install --upgrade pip
pip install -e .
```

5.2 Add new antennas

TBD

5.3 Post issues

Use the AEDT Antenna Toolkit Issues page to report bugs and request new features.

5.4 View documentation

Documentation for the latest stable release is hosted at https://aedt.antenna.toolkit.docs.pyansys.com/.

In the upper right corner of the documentations title bar, there is an option for switching from viewing the documentation for the latest stable release to viewing the documentation for the development version or previously released versions.

5.5 Adhere to code style

PyAEDT toolkit is compliant with PyAnsys code style. It uses the tool pre-commit to select the code style. You can install and activate this tool with:

```
pip install pre-commit
pre-commit run --all-files
```

You can also install this as a pre-commit hook with:

```
pre-commit install
```

This way, its not possible for you to push code that fails the style checks. For example:

```
$ pre-commit install
$ git commit -am "Add my cool feature."
black....................................................................Passed
isort (python)...........................................................Passed
flake8...................................................................Passed
codespell................................................................Passed
fix requirements.txt.....................................................Passed
blacken-docs.............................................................Passed
```

5.5.1 Maximum line length

Best practice is to keep the length at or below 120 characters for code and comments. Lines longer than this might not display properly on some terminals and tools or might be difficult to follow.

CHAPTER

SIX

RELEASE NOTES

This document contains the release notes for the project.

6.1 0.8.0 - May 20, 2025

Dependencies

Bump actions/download-artifact from 4.1.9 to 4.3.0	#330

Maintenance

Update v0.8.dev0	#320
update CHANGELOG for v0.7.1	#325
Fix codecov job	#327
Update CICD to Python 3.12	#328

6.2 0.7.1 - May 08, 2025

Fixed

Upload artifacts	#318
Fix release artifacts	#323

Maintenance

update CHANGELOG for v0.6.0	#317
Fix release process	#321

6.3 0.6.0 - May 06, 2025

Dependencies

Bump codecov/codecov-action from 4 to 5	#315

Maintenance

Add Changelog	#314

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- search

BIBLIOGRAPHY

[1] C. Balanis, Wideband and Travelling-Wave Antennas, *Modern Antenna Handbook*, New York, 2008.

[1] C. Balanis, Wideband and Travelling-Wave Antennas, *Modern Antenna Handbook*, New York, 2008.

[1] C. Balanis, Wideband and Travelling-Wave Antennas, *Modern Antenna Handbook*, New York, 2008.

[1] R. Johnson, Frequency Independent Antennas, Antenna Engineering Handbook, 3rd ed. New York, McGraw-Hill, 1993.

[1] C. Balanis, Wideband and Travelling-Wave Antennas, *Modern Antenna Handbook*, New York, 2008.

[1] C. Balanis, Aperture Antennas: Analysis, Design, and Applications, *Modern Antenna Handbook*, New York, 2008.

[1] C. Balanis, Horn Antennas, Antenna Theory Analysis, 3rd ed. Hoboken: Wiley, 2005, sec. 13.6, pp. 785-791.

[1] C. Balanis, Aperture Antennas: Analysis, Design, and Applications, *Modern Antenna Handbook*, New York, 2008.

[1] C. Balanis, Aperture Antennas: Analysis, Design, and Applications, *Modern Antenna Handbook*, New York, 2008.

[1] C. Balanis, Aperture Antennas: Analysis, Design, and Applications, *Modern Antenna Handbook*, New York, 2008.

[1] C. Balanis, Aperture Antennas: Analysis, Design, and Applications, *Modern Antenna Handbook*, New York, 2008.

[1] C. Balanis, Aperture Antennas: Analysis, Design, and Applications, *Modern Antenna Handbook*, New York, 2008.

[1] K. L. Walton and V. C. Sundberg, Broadband ridged horn design, Microwave J., vol. 4, pp. 96-101, Apr. 1964.

[2] C. Bruns et al., Analysis and Simulations of a 1-18 GHz Broadband Double-Ridged Horn Antenna, IEEE Electromag. Compat., vol. 45, pp. 55-60, Feb 2003.

[3] C. Balanis, Horn Antennas, Antenna Theory Analysis, 3rd ed. Hoboken: Wiley, 2005, ch. 13.

[1] C. Balanis, Microstrip Antennas, *Antenna Theory*, 2nd Ed. New York: Wiley, 1997.

[1] C. Balanis, Microstrip Antennas, *Antenna Theory*, 2nd Ed. New York: Wiley, 1997.

[1] C. Balanis, Microstrip Antennas, *Antenna Theory*, 2nd Ed. New York: Wiley, 1997.