Skip to main content

__Back to top

__ `Ctrl`+`K`

[![conda 25.1.2.dev29 documentation -
Home](../../../../_static/conda_logo_full.svg)](../../../../index.html)

  * [Conda](https://docs.conda.io/projects/conda/)
  * [Conda-build](https://docs.conda.io/projects/conda-build)
  * [Miniconda](https://docs.anaconda.com/free/miniconda/)
  * conda.org

  * [__ GitHub](https://github.com/conda/conda "GitHub")
  * [![Element](../../../../_static/element_logo.svg)](http://bit.ly/conda-chat-room "Element")
  * [__ Discourse](https://conda.discourse.group/ "Discourse")

  * [Conda](https://docs.conda.io/projects/conda/)
  * [Conda-build](https://docs.conda.io/projects/conda-build)
  * [Miniconda](https://docs.anaconda.com/free/miniconda/)
  * conda.org

  * [__ GitHub](https://github.com/conda/conda "GitHub")
  * [![Element](../../../../_static/element_logo.svg)](http://bit.ly/conda-chat-room "Element")
  * [__ Discourse](https://conda.discourse.group/ "Discourse")

Navigation

  * [`_os`](../common/_os/index.html) __

    * [` linux`](../common/_os/linux/index.html)

    * [`osx`](../common/_os/osx/index.html)

    * [`unix`](../common/_os/unix/index.html)

    * [`windows`](../common/_os/windows/index.html)

  * [`compat`](../common/compat/index.html)

  * [`configuration`](../common/configuration/index.html)

  * [`constants`](../common/constants/index.html)

  * [`disk`](../common/disk/index.html)

  * [`io`](../common/io/index.html)

  * [`iterators`](../common/iterators/index.html)

  * [`logic`](../common/logic/index.html)

  * [`path`](../common/path/index.html) __

    * [` _cygpath`](../common/path/_cygpath/index.html)

    * [`directories`](../common/path/directories/index.html)

    * [`python`](../common/path/python/index.html)

    * [`windows`](../common/path/windows/index.html)

  * [`pkg_formats`](../common/pkg_formats/index.html) __

    * [` python`](../common/pkg_formats/python/index.html)

  * [`serialize`](../common/serialize/index.html)

  * [`signals`](../common/signals/index.html)

  * [`toposort`](../common/toposort/index.html)

  * [`url`](../common/url/index.html)

* [`core`](../core/index.html) __

  * [` envs_manager`](../core/envs_manager/index.html)

  * [`index`](../core/index/index.html)

  * [`initialize`](../core/initialize/index.html)

  * [`link`](../core/link/index.html)

  * [`package_cache_data`](../core/package_cache_data/index.html)

  * [`path_actions`](../core/path_actions/index.html)

  * [`portability`](../core/portability/index.html)

  * [`prefix_data`](../core/prefix_data/index.html)

  * [`solve`](../core/solve/index.html)

  * [`subdir_data`](../core/subdir_data/index.html)

* [`deprecations`](../deprecations/index.html)

* [`env`](../env/index.html) __

  * [` env`](../env/env/index.html)

  * [`installers`](../env/installers/index.html) __

    * [` base`](../env/installers/base/index.html)

    * [`conda`](../env/installers/conda/index.html)

    * [`pip`](../env/installers/pip/index.html)

  * [`pip_util`](../env/pip_util/index.html)

  * [`specs`](../env/specs/index.html) __

    * [` binstar`](../env/specs/binstar/index.html)

    * [`requirements`](../env/specs/requirements/index.html)

    * [`yaml_file`](../env/specs/yaml_file/index.html)

* [`exception_handler`](../exception_handler/index.html)

* [`exceptions`](../exceptions/index.html)

* [`exports`](../exports/index.html)

* [`gateways`](../gateways/index.html) __

  * [` anaconda_client`](../gateways/anaconda_client/index.html)

  * [`connection`](../gateways/connection/index.html) __

    * [` adapters`](../gateways/connection/adapters/index.html) __

      * [` ftp`](../gateways/connection/adapters/ftp/index.html)

      * [`types`](types/index.html)

      * [`virtual_packages`](virtual_packages/index.html) __

        * [` archspec`](virtual_packages/archspec/index.html)

        * [`conda`](virtual_packages/conda/index.html)

        * [`cuda`](virtual_packages/cuda/index.html)

        * [`freebsd`](virtual_packages/freebsd/index.html)

        * [`linux`](virtual_packages/linux/index.html)

        * [`osx`](virtual_packages/osx/index.html)

        * [`windows`](virtual_packages/windows/index.html)

    * [`reporters`](../reporters/index.html)

    * [`resolve`](../resolve/index.html)

    * [`testing`](../testing/index.html) __

      * [` cases`](../testing/cases/index.html)

      * [`fixtures`](../testing/fixtures/index.html)

      * [`gateways`](../testing/gateways/index.html) __

        * [` fixtures`](../testing/gateways/fixtures/index.html)

      * [`helpers`](../testing/helpers/index.html)

      * [`integration`](../testing/integration/index.html)

      * [`notices`](../testing/notices/index.html) __

        * [` fixtures`](../testing/notices/fixtures/index.html)

        * [`helpers`](../testing/notices/helpers/index.html)

      * [`solver_helpers`](../testing/solver_helpers/index.html)

    * [`trust`](../trust/index.html) __

      * [` constants`](../trust/constants/index.html)

      * [`signature_verification`](../trust/signature_verification/index.html)

    * [`utils`](../utils/index.html)

  * [`conda_env`](../../conda_env/index.html) __

* [` cli`](../../conda_env/cli/index.html)

* [`installers`](../../conda_env/installers/index.html)

* [__](../../../../index.html)

* [Developer guide](../../../index.html)

* __

* [` conda`](../index.html)

*

# `plugins`#

In this module, you will find everything relevant to conda's plugin system. It

contains all of the code that plugin authors will use to write plugins, as

well as conda's internal implementations of plugins.

Modules relevant for plugin authors

* [`conda.plugins.hookspec`](hookspec/index.html#module-conda.plugins.hookspec "conda.plugins.hookspec"): all available hook specifications are listed here, including examples of how to use them

* [`conda.plugins.types`](types/index.html#module-conda.plugins.types "conda.plugins.types"): important types to use when defining plugin hooks

Modules relevant for internal development

* [`conda.plugins.manager`](manager/index.html#module-conda.plugins.manager

"conda.plugins.manager"): includes our custom subclass of pluggy's [PluginManager](https://pluggy.readthedocs.io/en/stable/api_reference.html#pluggy.PluginManager) class

Modules with internal plugin implementations

* [`conda.plugins.solvers`](solvers/index.html#module-conda.plugins.solvers "conda.plugins.solvers"): implementation of the "classic" solver

* [`conda.plugins.subcommands.doctor`](subcommands/doctor/index.html#module-conda.plugins.subcommands.doctor "conda.plugins.subcommands.doctor"): `conda doctor` subcommand

* [`conda.plugins.virtual_packages`](virtual_packages/index.html#module-conda.plugins.virtual_packages "conda.plugins.virtual_packages"): registers virtual packages in conda

Classes#

`CondaAuthHandler`	Return type to use when the defining the conda auth handlers hook.
`CondaHealthCheck`	Return type to use when defining conda health checks plugin hook.
`CondaPostCommand`	Return type to use when defining a conda post-command plugin hook.
`CondaPostSolve`	Return type to use when defining a conda post-solve plugin hook.
`CondaPreCommand`	Return type to use when defining a conda pre-command plugin hook.
`CondaPreSolve`	Return type to use when defining a conda pre-solve plugin hook.
`CondaReporterBackend`	Return type to use when defining a conda reporter backend plugin hook.

`CondaRequestHeader` | Define vendor specific headers to include HTTP requests

`CondaSetting` | Return type to use when defining a conda setting plugin hook.

`CondaSolver` | Return type to use when defining a conda solver plugin hook.

`CondaSubcommand` | Return type to use when defining a conda subcommand plugin hook.

`CondaVirtualPackage` | Return type to use when defining a conda virtual package plugin hook.

Attributes#

`hookimpl` | Decorator used to mark plugin hook implementations

---|---

hookimpl#

Decorator used to mark plugin hook implementations

_class _CondaAuthHandler#

Bases: `NamedTuple`

Return type to use when the defining the conda auth handlers hook.

Parameters:

* **name** \-- Name (e.g., `basic-auth`). This name should be unique and only one may be registered at a time.

* **handler** \-- Type that will be used as the authentication handler during network requests.

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python
v3.13\)")_#

handler _: [type](https://docs.python.org/3/library/functions.html#type "\(in
Python
v3.13\)")[[ChannelAuthBase](types/index.html#conda.plugins.types.ChannelAuthBase
"conda.plugins.types.ChannelAuthBase")]_#

_class _CondaHealthCheck#

Bases: `NamedTuple`

Return type to use when defining conda health checks plugin hook.

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")_#

action _: Callable[[[str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)"), [bool](https://docs.python.org/3/library/functions.html#bool "\(in Python v3.13\)")], [None](https://docs.python.org/3/library/constants.html#None "\(in Python v3.13\)")]_#

_class _CondaPostCommand#

Bases: `NamedTuple`

Return type to use when defining a conda post-command plugin hook.

For details on how this is used, see [`conda_post_commands()`](../../../plugins/post_commands.html#conda.plugins.hookspec.CondaSpecs.conda_post_commands "conda.plugins.hookspec.CondaSpecs.conda_post_commands").

Parameters:

  * **name** \-- Post-command name (e.g., `custom_plugin_post_commands`).

  * **action** \-- Callable which contains the code to be run.

  * **run_for** \-- Represents the command(s) this will be run on (e.g. `install` or `create`).

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")_#

action _: Callable[[[str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")], [None](https://docs.python.org/3/library/constants.html#None "\(in Python v3.13\)")]_#

run_for _: [set](https://docs.python.org/3/library/stdtypes.html#set "\(in Python v3.13\)")[[str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")]_#

_class _CondaPostSolve#

Return type to use when defining a conda post-solve plugin hook.

For details on how this is used, see

[`conda_post_solves()`](hookspec/index.html#conda.plugins.hookspec.CondaSpecs.conda_post_so

lves

"conda.plugins.hookspec.CondaSpecs.conda_post_solves").

Parameters:

  * **name** \-- Post-solve name (e.g., `custom_plugin_post_solve`).

  * **action** \-- Callable which contains the code to be run.

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python

v3.13\)")_#

action _: Callable[[[str](https://docs.python.org/3/library/stdtypes.html#str

"\(in Python v3.13\)"),

[tuple](https://docs.python.org/3/library/stdtypes.html#tuple "\(in Python

v3.13\)")[[conda.models.records.PackageRecord](../models/records/index.html#conda.models.recor

ds.PackageRecord

"conda.models.records.PackageRecord"), Ellipsis],

[tuple](https://docs.python.org/3/library/stdtypes.html#tuple "\(in Python

v3.13\)")[[conda.models.records.PackageRecord](../models/records/index.html#conda.models.recor

ds.PackageRecord

"conda.models.records.PackageRecord"), Ellipsis]],

[None](https://docs.python.org/3/library/constants.html#None "\(in Python

v3.13\)")]_#

_class _CondaPreCommand#

Bases: `NamedTuple`

Return type to use when defining a conda pre-command plugin hook.

For details on how this is used, see

[`conda_pre_commands()`](../../../plugins/pre_commands.html#conda.plugins.hookspec.CondaSpec

s.conda_pre_commands

"conda.plugins.hookspec.CondaSpecs.conda_pre_commands").

Parameters:

  * **name** \-- Pre-command name (e.g., `custom_plugin_pre_commands`).

  * **action** \-- Callable which contains the code to be run.

  * **run_for** \-- Represents the command(s) this will be run on (e.g. `install` or `create`).

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")_#

action _: Callable[[[str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")], [None](https://docs.python.org/3/library/constants.html#None "\(in Python v3.13\)")]_#

run_for _: [set](https://docs.python.org/3/library/stdtypes.html#set "\(in Python v3.13\)")[[str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")]_#

_class _CondaPreSolve#

Return type to use when defining a conda pre-solve plugin hook.

For details on how this is used, see

[`conda_pre_solves()`](hookspec/index.html#conda.plugins.hookspec.CondaSpecs.conda_pre_solv

es

"conda.plugins.hookspec.CondaSpecs.conda_pre_solves").

Parameters:

  * **name** \-- Pre-solve name (e.g., `custom_plugin_pre_solve`).

  * **action** \-- Callable which contains the code to be run.

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python

v3.13\)")_#

action _:

Callable[[[frozenset](https://docs.python.org/3/library/stdtypes.html#frozenset

"\(in Python

v3.13\)")[[conda.models.match_spec.MatchSpec](../models/match_spec/index.html#conda.models.

match_spec.MatchSpec

"conda.models.match_spec.MatchSpec")],

[frozenset](https://docs.python.org/3/library/stdtypes.html#frozenset "\(in

Python

v3.13\)")[[conda.models.match_spec.MatchSpec](../models/match_spec/index.html#conda.models.

match_spec.MatchSpec

"conda.models.match_spec.MatchSpec")]],

[None](https://docs.python.org/3/library/constants.html#None "\(in Python

v3.13\)")]_#

_class _CondaReporterBackend#

Return type to use when defining a conda reporter backend plugin hook.

For details on how this is used, see:

[`conda_reporter_backends()`](hookspec/index.html#conda.plugins.hookspec.CondaSpecs.conda_reporter_backends

"conda.plugins.hookspec.CondaSpecs.conda_reporter_backends").

Parameters:

* **name** \-- name of the reporter backend (e.g., `email_reporter`) This is how the reporter backend with be references in configuration files.

* **description** \-- short description of what the reporter handler does

* **renderer** \-- implementation of `ReporterRendererBase` that will be used as the reporter renderer

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")_#

description _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")_#

renderer _: [type](https://docs.python.org/3/library/functions.html#type "\(in Python v3.13\)")[[ReporterRendererBase](types/index.html#conda.plugins.types.ReporterRendererBase "conda.plugins.types.ReporterRendererBase")]_#

_class _CondaRequestHeader#

Define vendor specific headers to include HTTP requests

For details on how this is used, see [`conda_request_headers()`](../../../plugins/request_headers.html#conda.plugins.hookspec.CondaSpecs.conda_request_headers "conda.plugins.hookspec.CondaSpecs.conda_request_headers") and [`conda_session_headers()`](../../../plugins/request_headers.html#conda.plugins.hookspec.CondaSpecs.conda_session_headers "conda.plugins.hookspec.CondaSpecs.conda_session_headers").

Parameters:

* **name** \-- name of the header used in the HTTP request

* **value** \-- value of the header used in the HTTP request

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")_#

value _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in

Python v3.13\)")_#

_class _CondaSetting#

Return type to use when defining a conda setting plugin hook.

For details on how this is used, see

[`conda_settings()`](../../../plugins/settings.html#conda.plugins.hookspec.CondaSpecs.conda_settings

"conda.plugins.hookspec.CondaSpecs.conda_settings").

Parameters:

  * **name** \-- name of the setting (e.g., `config_param`)

  * **description** \-- description of the setting that should be targeted towards users of the plugin

  * **parameter** \-- Parameter instance containing the setting definition

  * **aliases** \-- alternative names of the setting

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")_#

description _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")_#

parameter _: [conda.common.configuration.Parameter](../common/configuration/index.html#conda.common.configuration.Parameter "conda.common.configuration.Parameter")_#

aliases _: [tuple](https://docs.python.org/3/library/stdtypes.html#tuple "\(in Python v3.13\)")[[str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)"), Ellipsis]_#

_class _CondaSolver#

Bases: `NamedTuple`

Return type to use when defining a conda solver plugin hook.

For details on how this is used, see

[`conda_solvers()`](../../../plugins/solvers.html#conda.plugins.hookspec.CondaSpecs.conda_solvers

"conda.plugins.hookspec.CondaSpecs.conda_solvers").

Parameters:

  * **name** \-- Solver name (e.g., `custom-solver`).

  * **backend** \-- Type that will be instantiated as the solver backend.

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python

v3.13\)")_#

backend _: [type](https://docs.python.org/3/library/functions.html#type "\(in

Python

v3.13\)")[[conda.core.solve.Solver](../core/solve/index.html#conda.core.solve.Solver

"conda.core.solve.Solver")]_#

_class _CondaSubcommand#

Return type to use when defining a conda subcommand plugin hook.

For details on how this is used, see

[`conda_subcommands()`](../../../plugins/subcommands.html#conda.plugins.hookspec.CondaSpecs.

conda_subcommands

"conda.plugins.hookspec.CondaSpecs.conda_subcommands").

Parameters:

  * **name** \-- Subcommand name (e.g., `conda my-subcommand-name`).

  * **summary** \-- Subcommand summary, will be shown in `conda --help`.

  * **action** \-- Callable that will be run when the subcommand is invoked.

  * **configure_parser** \-- Callable that will be run when the subcommand parser is initialized.

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python

v3.13\)")_#

summary _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")_#

action _: Callable[[[argparse.Namespace](https://docs.python.org/3/library/argparse.html#argparse.Namespace "\(in Python v3.13\)") | [tuple](https://docs.python.org/3/library/stdtypes.html#tuple "\(in Python v3.13\)")[[str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")]], [int](https://docs.python.org/3/library/functions.html#int "\(in Python v3.13\)") | [None](https://docs.python.org/3/library/constants.html#None "\(in Python v3.13\)")]_#

configure_parser _: Callable[[[argparse.ArgumentParser](https://docs.python.org/3/library/argparse.html#argparse.ArgumentParser "\(in Python v3.13\)")], [None](https://docs.python.org/3/library/constants.html#None "\(in Python v3.13\)")] | [None](https://docs.python.org/3/library/constants.html#None "\(in Python v3.13\)")_#

_class _CondaVirtualPackage#

Bases: `NamedTuple`

Return type to use when defining a conda virtual package plugin hook.

For details on how this is used, see [`conda_virtual_packages()`](../../../plugins/virtual_packages.html#conda.plugins.hookspec.CondaSpecs.conda_virtual_packages "conda.plugins.hookspec.CondaSpecs.conda_virtual_packages").

Parameters:

  * **name** \-- Virtual package name (e.g., `my_custom_os`).

  * **version** \-- Virtual package version (e.g., `1.2.3`).

  * **build** \-- Virtual package build string (e.g., `x86_64`).

name _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)")_#

version _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)") | [None](https://docs.python.org/3/library/constants.html#None "\(in Python v3.13\)")_#

build _: [str](https://docs.python.org/3/library/stdtypes.html#str "\(in Python v3.13\)") | [None](https://docs.python.org/3/library/constants.html#None "\(in Python v3.13\)")_#

to_virtual_package() -> [conda.models.records.PackageRecord](../models/records/index.html#conda.models.records.PackageRecord "conda.models.records.PackageRecord")#

__On this page

  * `CondaPostSolve.name`

  * `CondaPostSolve.action`

* `CondaPreCommand`

  * `CondaPreCommand.name`

  * `CondaPreCommand.action`

  * `CondaPreCommand.run_for`

* `CondaPreSolve`

  * `CondaPreSolve.name`

  * `CondaPreSolve.action`

* `CondaReporterBackend`

  * `CondaReporterBackend.name`

  * `CondaReporterBackend.description`

  * `CondaReporterBackend.renderer`

* `CondaRequestHeader`

  * `CondaRequestHeader.name`

  * `CondaRequestHeader.value`

* `CondaSetting`

  * `CondaSetting.name`

  * `CondaSetting.description`

  * `CondaSetting.parameter`

  * `CondaSetting.aliases`

* `CondaSolver`

  * `CondaSolver.name`

  * `CondaSolver.backend`

* `CondaSubcommand`

  * `CondaSubcommand.name`

  * `CondaSubcommand.summary`

    * `CondaSubcommand.action`

    * `CondaSubcommand.configure_parser`

  * `CondaVirtualPackage`

    * `CondaVirtualPackage.name`

    * `CondaVirtualPackage.version`

    * `CondaVirtualPackage.build`

    * `CondaVirtualPackage.to_virtual_package()`

[__Edit on GitHub](https://github.com/conda/conda/edit/main/docs/source/dev-guide/api/conda/plugins/index.rst)

[__Show Source](../../../../_sources/dev-guide/api/conda/plugins/index.rst.txt)

Created using [Sphinx](https://www.sphinx-doc.org/) 7.4.7.

[Analytics Dashboard __](https://docs-conda-io.goatcounter.com "Analytics Dashboard")

Built with the [PyData Sphinx Theme](https://pydata-sphinx-theme.readthedocs.io/en/stable/index.html) 0.15.4.