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

  * [User guide](../../../../../user-guide/index.html) __

    * [Getting started with conda](../../../../../user-guide/getting-started.html)

    * [Installing conda](../../../../../user-guide/install/index.html) __

      * [Installing on Windows](../../../../../user-guide/install/windows.html)

      * [Installing on macOS](../../../../../user-guide/install/macos.html)

      * [Installing on Linux](../../../../../user-guide/install/linux.html)

      * [RPM and Debian Repositories for Miniconda](../../../../../user-guide/install/rpm-debian.html)

    * [Tasks](../../../../../user-guide/tasks/index.html) __

      * [Managing conda](../../../../../user-guide/tasks/manage-conda.html)

      * [Managing environments](../../../../../user-guide/tasks/manage-environments.html)

      * [Managing channels](../../../../../user-guide/tasks/manage-channels.html)

      * [Managing packages](../../../../../user-guide/tasks/manage-pkgs.html)

      * [Managing Python](../../../../../user-guide/tasks/manage-python.html)

      * [Managing virtual packages](../../../../../user-guide/tasks/manage-virtual.html)

      * [Creating custom channels](../../../../../user-guide/tasks/create-custom-channels.html)

      * [Creating projects](../../../../../user-guide/tasks/creating-projects.html)

      * [Viewing command-line help](../../../../../user-guide/tasks/view-command-line-help.html)

    * [Configuration](../../../../../user-guide/configuration/index.html) __

      * [Using the .condarc conda configuration file](../../../../../user-guide/configuration/use-condarc.html)

      * [Settings](../../../../../user-guide/configuration/settings.html)

      * [Administering a multi-user conda installation](../../../../../user-guide/configuration/admin-multi-user-install.html)

      * [Mirroring channels](../../../../../user-guide/configuration/mirroring.html)

      * [Disabling SSL verification](../../../../../user-guide/configuration/disable-ssl-verification.html)

* [Using non-standard certificates](../../../../../user-guide/configuration/non-standard-certs.html)

* [Using Custom Locations for Environment and Package Cache](../../../../../user-guide/configuration/custom-env-and-pkg-locations.html)

* [Improving interoperability with pip](../../../../../user-guide/configuration/pip-interoperability.html)

* [Using the free channel](../../../../../user-guide/configuration/free-channel.html)

* [Concepts](../../../../../user-guide/concepts/index.html) __

* [Commands](../../../../../user-guide/concepts/conda-commands.html)

* [Packages](../../../../../user-guide/concepts/packages.html)

* [Package specification](../../../../../user-guide/concepts/pkg-specs.html)

* [Package search and install specifications](../../../../../user-guide/concepts/pkg-search.html)

* [Channels](../../../../../user-guide/concepts/channels.html)

* [Environments](../../../../../user-guide/concepts/environments.html)

* [Installing with conda](../../../../../user-guide/concepts/installing-with-conda.html)

* [Performance](../../../../../user-guide/concepts/conda-performance.html)

* [Conda for data scientists](../../../../../user-guide/concepts/data-science.html)

* [Plugins](../../../../../user-guide/concepts/conda-plugins.html)

* [Troubleshooting](../../../../../user-guide/troubleshooting.html)

* [Cheatsheet](../../../../../user-guide/cheatsheet.html)

* [Configuration](../../../../../configuration.html)

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

* [` conda clean`](../../../../../commands/clean.html)

* [`conda compare`](../../../../../commands/compare.html)

* [`conda config`](../../../../../commands/config.html)

* [`conda create`](../../../../../commands/create.html)

* [`conda doctor`](../../../../../commands/doctor.html)

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

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

* [` conda env config vars`](../../../../../commands/env/config/vars/index.html) __

* [` conda env config vars list`](../../../../../commands/env/config/vars/list.html)

* [`conda env config vars set`](../../../../../commands/env/config/vars/set.html)

* [`conda env config vars unset`](../../../../../commands/env/config/vars/unset.html)

* [`conda env create`](../../../../../commands/env/create.html)

* [`conda env export`](../../../../../commands/env/export.html)

* [`conda env list`](../../../../../commands/env/list.html)

* [`conda env remove`](../../../../../commands/env/remove.html)

* [`conda env update`](../../../../../commands/env/update.html)

* [`conda info`](../../../../../commands/info.html)

* [`conda init`](../../../../../commands/init.html)

* [`conda install`](../../../../../commands/install.html)

* [`conda list`](../../../../../commands/list.html)

* [`conda notices`](../../../../../commands/notices.html)

* [`conda package`](../../../../../commands/package.html)

* [`conda remove`](../../../../../commands/remove.html)

* [`conda rename`](../../../../../commands/rename.html)

* [`conda run`](../../../../../commands/run.html)

* [`conda search`](../../../../../commands/search.html)

* [`conda update`](../../../../../commands/update.html)

* [Release notes](../../../../../release-notes.html)

* [Glossary](../../../../../glossary.html)

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

* [Architecture](../../../../architecture.html)

* [Contributing to conda](../../../../contributing.html)

* [Development Environment](../../../../development-environment.html)

* [Deep dives](../../../../deep-dives/index.html) __

  * [` conda install`](../../../../deep-dives/install.html)

  * [`conda init` and `conda activate`](../../../../deep-dives/activation.html)

  * [`conda config` and context](../../../../deep-dives/context.html)

  * [Solvers](../../../../deep-dives/solvers.html)

  * [Logging](../../../../deep-dives/logging.html)

* [Writing Tests](../../../../writing-tests/index.html) __

  * [Integration Tests](../../../../writing-tests/integration-tests.html)

* [Deprecations](../../../../deprecations.html)

* [Releasing](../../../../releasing.html)

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

  * [Auth Handlers](../../../../plugins/auth_handlers.html)

  * [Health Checks](../../../../plugins/health_checks.html)

  * [Request Headers](../../../../plugins/request_headers.html)

  * [Post-commands](../../../../plugins/post_commands.html)

  * [Pre-commands](../../../../plugins/pre_commands.html)

  * [Reporter Backends](../../../../plugins/reporter_backends.html)

  * [Settings](../../../../plugins/settings.html)

  * [Solvers](../../../../plugins/solvers.html)

  * [Subcommands](../../../../plugins/subcommands.html)

  * [Virtual Packages](../../../../plugins/virtual_packages.html)

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

  * [Technical specification: solver state](../../../../specs/solver-state.html)

* [API](../../../../api.html) __

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

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

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

      * [` frozendict`](../../_vendor/frozendict/index.html)

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

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

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

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

  * [` collection`](../../auxlib/collection/index.html)

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

  * [`decorators`](../../auxlib/decorators/index.html)

  * [`entity`](../../auxlib/entity/index.html)

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

  * [`ish`](../../auxlib/ish/index.html)

  * [`logz`](../../auxlib/logz/index.html)

  * [`type_coercion`](../../auxlib/type_coercion/index.html)

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

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

  * [`context`](../../base/context/index.html)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  * [`_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)

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

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

      * [`s3`](../../gateways/connection/adapters/s3/index.html)

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

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

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

    * [` create`](../../gateways/disk/create/index.html)

    * [`delete`](../../gateways/disk/delete/index.html)

    * [`link`](../../gateways/disk/link/index.html)

    * [`lock`](../../gateways/disk/lock/index.html)

    * [`permissions`](../../gateways/disk/permissions/index.html)

    * [`read`](../../gateways/disk/read/index.html)

    * [`test`](../../gateways/disk/test/index.html)

    * [`update`](../../gateways/disk/update/index.html)

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

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

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

      * [` core`](../../gateways/repodata/jlap/core/index.html)

      * [`fetch`](../../gateways/repodata/jlap/fetch/index.html)

      * [`interface`](../../gateways/repodata/jlap/interface/index.html)

    * [`lock`](../../gateways/repodata/lock/index.html)

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

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

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

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

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

  * [` channel`](../../models/channel/index.html)

  * [`dist`](../../models/dist/index.html)

  * [`enums`](../../models/enums/index.html)

  * [`leased_path_entry`](../../models/leased_path_entry/index.html)

  * [`match_spec`](../../models/match_spec/index.html)

  * [`package_info`](../../models/package_info/index.html)

  * [`prefix_graph`](../../models/prefix_graph/index.html)

  * [`records`](../../models/records/index.html)

  * [`version`](../../models/version/index.html)

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

  * [` cache`](../../notices/cache/index.html)

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

  * [`fetch`](../../notices/fetch/index.html)

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

  * [`views`](../../notices/views/index.html)

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

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

  * ` hookspec`

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

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

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

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

    * [` console`](../reporter_backends/console/index.html)

    * [`json`](../reporter_backends/json/index.html)

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

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

    * [` doctor`](../subcommands/doctor/index.html) __

      * [` health_checks`](../subcommands/doctor/health_checks/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) __

# `hookspec`#

Pluggy hook specifications ("hookspecs") to register conda plugins.

Each hookspec defined in `CondaSpecs` contains an example of how to use it.

## Classes#

`CondaSpecs`	The conda plugin hookspecs, to be used by developers.

## Attributes#

`spec_name`	Name used for organizing conda hook specifications
`_hookspec`	The conda plugin hook specifications, to be used by developers
`hookimpl`	Decorator used to mark plugin hook implementations

spec_name _ = 'conda'_#

Name used for organizing conda hook specifications

_hookspec#

The conda plugin hook specifications, to be used by developers

hookimpl#

Decorator used to mark plugin hook implementations

_class _CondaSpecs#

The conda plugin hookspecs, to be used by developers.

conda_solvers() ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaSolver](../../../../plugins/solvers.html#conda.plugins.types.Cond

aSolver

"conda.plugins.types.CondaSolver")]#

Register solvers in conda.

Example:

```
import logging


from conda import plugins
from conda.core import solve


log = logging.getLogger(__name__)



class VerboseSolver(solve.Solver):
    def solve_final_state(self, *args, **kwargs):
        log.info("My verbose solver!")
        return super().solve_final_state(*args, **kwargs)
@plugins.hookimpl
def conda_solvers():
    yield plugins.CondaSolver(
        name="verbose-classic",
        backend=VerboseSolver,
    )
```

Returns:

An iterable of solver entries.

conda_subcommands() ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaSubcommand](../../../../plugins/subcommands.html#conda.plugi

ns.types.CondaSubcommand

"conda.plugins.types.CondaSubcommand")]#

Register external subcommands in conda.

Example:

```
from conda import plugins


def example_command(args):
    print("This is an example command!")


@plugins.hookimpl
def conda_subcommands():
    yield plugins.CondaSubcommand(
        name="example",
        summary="example command",
        action=example_command,
    )
```

Returns:

An iterable of subcommand entries.

conda_virtual_packages() ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaVirtualPackage](../../../../plugins/virtual_packages.html#conda.p

lugins.types.CondaVirtualPackage

"conda.plugins.types.CondaVirtualPackage")]#

Register virtual packages in Conda.

Example:

```
    from conda import plugins



    @plugins.hookimpl
    def conda_virtual_packages():
        yield plugins.CondaVirtualPackage(
            name="my_custom_os",
            version="1.2.3",
            build="x86_64",
        )
```

Returns:

An iterable of virtual package entries.

conda_pre_commands() ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaPreCommand](../../../../plugins/pre_commands.html#conda.plu

gins.types.CondaPreCommand

"conda.plugins.types.CondaPreCommand")]#

Register pre-command functions in conda.

Example:

```
from conda import plugins


def example_pre_command(command):
    print("pre-command action")
@plugins.hookimpl
def conda_pre_commands():
    yield plugins.CondaPreCommand(
        name="example-pre-command",
        action=example_pre_command,
        run_for={"install", "create"},
    )
```

conda_post_commands() ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaPostCommand](../../../../plugins/post_commands.html#conda.pl

ugins.types.CondaPostCommand

"conda.plugins.types.CondaPostCommand")]#

Register post-command functions in conda.

Example:

```
    from conda import plugins
    def example_post_command(command):
        print("post-command action")



    @plugins.hookimpl
    def conda_post_commands():
        yield plugins.CondaPostCommand(
            name="example-post-command",
            action=example_post_command,
            run_for={"install", "create"},
        )
```

conda_auth_handlers() ->
[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl
e
"\(in Python
v3.13\)")[[conda.plugins.types.CondaAuthHandler](../../../../plugins/auth_handlers.html#conda.plugin
s.types.CondaAuthHandler
"conda.plugins.types.CondaAuthHandler")]#

Register a conda auth handler derived from the requests API.

This plugin hook allows attaching requests auth handler subclasses, e.g. when authenticating requests against individual channels hosted at HTTP/HTTPS services.

Example:

```
import os
from conda import plugins
from requests.auth import AuthBase


class EnvironmentHeaderAuth(AuthBase):
    def __init__(self, *args, **kwargs):
        self.username = os.environ["EXAMPLE_CONDA_AUTH_USERNAME"]
        self.password = os.environ["EXAMPLE_CONDA_AUTH_PASSWORD"]

    def __call__(self, request):
        request.headers["X-Username"] = self.username
        request.headers["X-Password"] = self.password
        return request


@plugins.hookimpl
def conda_auth_handlers():
    yield plugins.CondaAuthHandler(
        name="environment-header-auth",
        handler=EnvironmentHeaderAuth,
    )
```

conda_health_checks() ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaHealthCheck](../../../../plugins/health_checks.html#conda.plugin

s.types.CondaHealthCheck

"conda.plugins.types.CondaHealthCheck")]#

Register health checks for conda doctor.

This plugin hook allows you to add more "health checks" to conda doctor that

you can write to diagnose problems in your conda environment. Check out the

health checks already shipped with conda for inspiration.

Example:

```
    from conda import plugins
    def example_health_check(prefix: str, verbose: bool):
        print("This is an example health check!")


    @plugins.hookimpl
    def conda_health_checks():
        yield plugins.CondaHealthCheck(
            name="example-health-check",
            action=example_health_check,
        )
```

conda_pre_solves() ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaPreSolve](../types/index.html#conda.plugins.types.CondaPreS

olve

"conda.plugins.types.CondaPreSolve")]#

Register pre-solve functions in conda that are used in the general solver API,

before the solver processes the package specs in search of a solution.

Example:

```
from conda import plugins
from conda.models.match_spec import MatchSpec


def example_pre_solve(
    specs_to_add: frozenset[MatchSpec],
    specs_to_remove: frozenset[MatchSpec],
):
    print(f"Adding {len(specs_to_add)} packages")
    print(f"Removing {len(specs_to_remove)} packages")


@plugins.hookimpl
def conda_pre_solves():
    yield plugins.CondaPreSolve(
        name="example-pre-solve",
        action=example_pre_solve,
    )
```

conda_post_solves() ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaPostSolve](../types/index.html#conda.plugins.types.CondaPost

Solve

"conda.plugins.types.CondaPostSolve")]#

Register post-solve functions in conda that are used in the general solver

API, after the solver has provided the package records to add or remove from

the conda environment.

Example:

```
    from conda import plugins
    from conda.models.records import PackageRecord


    def example_post_solve(
        repodata_fn: str,
        unlink_precs: tuple[PackageRecord, ...],
        link_precs: tuple[PackageRecord, ...],
    ):
        print(f"Uninstalling {len(unlink_precs)} packages")
        print(f"Installing {len(link_precs)} packages")
    @plugins.hookimpl
    def conda_post_solves():
        yield plugins.CondaPostSolve(
            name="example-post-solve",
            action=example_post_solve,
        )
```

conda_settings() ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaSetting](../../../../plugins/settings.html#conda.plugins.types.Con

daSetting

"conda.plugins.types.CondaSetting")]#

Register new setting

The example below defines a simple string type parameter

Example:

```
    from conda import plugins
    from conda.common.configuration import PrimitiveParameter, SequenceParameter



    @plugins.hookimpl
    def conda_settings():
        yield plugins.CondaSetting(
            name="example_option",
            description="This is an example option",
            parameter=PrimitiveParameter("default_value", element_type=str),
            aliases=("example_option_alias",),
        )
```

conda_reporter_backends() ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaReporterBackend](../types/index.html#conda.plugins.types.Con

daReporterBackend

"conda.plugins.types.CondaReporterBackend")]#

Register new reporter backend

The example below defines a reporter backend that uses the `pprint` module in

Python.

Example:

```
from pprint import pformat

from conda import plugins
from conda.plugins.types import (
    CondaReporterBackend,
    ReporterRendererBase,
    ProgressBarBase,
)


class PprintReporterRenderer(ReporterRendererBase):
    "Implementation of the ReporterRendererBase"

    def detail_view(self, data):
        return pformat(data)

    def envs_list(self, data):
        formatted_data = pformat(data)
        return f"Environments: {formatted_data}"

    def progress_bar(self, description, io_context_manager) -> ProgressBarBase:
        "Returns our custom progress bar implementation"
        return PprintProgressBar(description, io_context_manager)



class PprintProgressBar(ProgressBarBase):
    "Blank implementation of ProgressBarBase which does nothing"

    def update_to(self, fraction) -> None:
        pass

    def refresh(self) -> None:
        pass

    def close(self) -> None:
        pass



@plugins.hookimpl
def conda_reporter_backends():
    yield CondaReporterBackend(
        name="pprint",
        description="Reporter backend based on the pprint module",
        renderer=PprintReporterRenderer,
    )
```

conda_session_headers(_host :

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

v3.13\)")_) ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaRequestHeader](../../../../plugins/request_headers.html#conda.

plugins.types.CondaRequestHeader

"conda.plugins.types.CondaRequestHeader")]#

Register new HTTP request headers

The example below defines how to add HTTP headers for all requests with the

hostname of `example.com`.

Example:

```
    from conda import plugins


    HOSTS = {"example.com", "sub.example.com"}



    @plugins.hookimpl
    def conda_session_headers(host: str):
        if host in HOSTS:
```

```
        yield plugins.CondaRequestHeader(
            name="Example-Header",
            value="example",
        )
```

conda_request_headers(_host :

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

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

"\(in Python v3.13\)")_) ->

[collections.abc.Iterable](https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterabl

e

"\(in Python

v3.13\)")[[conda.plugins.types.CondaRequestHeader](../../../../plugins/request_headers.html#conda.

plugins.types.CondaRequestHeader

"conda.plugins.types.CondaRequestHeader")]#

Register new HTTP request headers

The example below defines how to add HTTP headers for all requests with the

hostname of `example.com` and a `path/to/endpoint.json` path.

Example:

```
from conda import plugins


HOSTS = {"example.com", "sub.example.com"}
ENDPOINT = "/path/to/endpoint.json"



@plugins.hookimpl
def conda_request_headers(host: str, path: str):
    if host in HOSTS and path == ENDPOINT:
        yield plugins.CondaRequestHeader(
            name="Example-Header",
            value="example",
        )
```

__On this page

* Classes
* Attributes
  * `spec_name`
  * `_hookspec`
  * `hookimpl`
  * `CondaSpecs`
    * `CondaSpecs.conda_solvers()`
    * `CondaSpecs.conda_subcommands()`
    * `CondaSpecs.conda_virtual_packages()`

* `CondaSpecs.conda_pre_commands()`

* `CondaSpecs.conda_post_commands()`

* `CondaSpecs.conda_auth_handlers()`

* `CondaSpecs.conda_health_checks()`

* `CondaSpecs.conda_pre_solves()`

* `CondaSpecs.conda_post_solves()`

* `CondaSpecs.conda_settings()`

* `CondaSpecs.conda_reporter_backends()`

* `CondaSpecs.conda_session_headers()`

* `CondaSpecs.conda_request_headers()`

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

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