Jupyter console Documentation

Release 6.0.0

The Jupyter Development Team

Oct 03, 2018

Contents

The Jupyter console is a terminal frontend for kernels using the Jupyter protocol. The console can be installed with:

```
pip install jupyter-console
```

If you want to use conda instead to perform your installation:

```
conda install -c conda-forge jupyter-console
```

And started with:

```
jupyter console
```

To see configuration options:

```
jupyter console -h
```

To start the console with a particular kernel, ask for it by name:

```
jupyter console --kernel=julia-0.4
```

A list of available kernels can be seen with:

```
jupyter kernelspec list
```

You can connect to a live kernel (e.g. one running in a notebook) with its ID:

```
jupyter console --existing KERNEL_ID
```

or even connect to the most recently started kernel by default:

```
jupyter console --existing
```

Contents:

CHAPTER 1

Configuration options

These options can be set in `~/.jupyter/jupyter_console_config.py`, or at the command line when you start it.

ConnectionFileMixin.connection_file [Unicode] Default: `''`

JSON file in which to store connection info [default: kernel-<pid>.json]

This file will contain the IP, ports, and authentication key needed to connect clients to this kernel. By default, this file will be created in the security dir of the current profile, but can be specified by absolute path.

ConnectionFileMixin.control_port [Int] Default: `0`

set the control (ROUTER) port [default: random]

ConnectionFileMixin.hb_port [Int] Default: `0`

set the heartbeat port [default: random]

ConnectionFileMixin.iopub_port [Int] Default: `0`

set the iopub (PUB) port [default: random]

ConnectionFileMixin.ip [Unicode] Default: `''`

Set the kernel's IP address [default localhost]. If the IP address is something other than localhost, then Consoles on other machines will be able to connect to the Kernel, so be careful!

ConnectionFileMixin.shell_port [Int] Default: `0`

set the shell (ROUTER) port [default: random]

ConnectionFileMixin.stdin_port [Int] Default: `0`

set the stdin (ROUTER) port [default: random]

ConnectionFileMixin.transport ['tcp'|'ipc'] Default: `'tcp'`

No description

JupyterConsoleApp.confirm_exit [CBool] Default: `True`

Set to display confirmation dialog on exit. You can always use 'exit' or 'quit', to force a direct exit without any confirmation.

JupyterConsoleApp.existing [CUnicode] Default: `''`

Connect to an already running kernel

JupyterConsoleApp.kernel_name [Unicode] Default: `'python'`

The name of the default kernel to start.

JupyterConsoleApp.sshkey [Unicode] Default: `''`

Path to the ssh key to use for logging in to the ssh server.

JupyterConsoleApp.sshserver [Unicode] Default: `''`

The SSH server to use to connect to the kernel.

Application.log_datefmt [Unicode] Default: `'%Y-%m-%d %H:%M:%S'`

The date format used by logging formatters for %(asctime)s

Application.log_format [Unicode] Default: `'[%(name)s]%(highlevel)s %(message)s'`

The Logging format template

Application.log_level [0|10|20|30|40|50|'DEBUG'|'INFO'|'WARN'|'ERROR'|'CRITICAL'] Default: `30`

Set the log level by value or name.

JupyterApp.answer_yes [Bool] Default: `False`

Answer yes to any prompts.

JupyterApp.config_file [Unicode] Default: `''`

Full path of a config file.

JupyterApp.config_file_name [Unicode] Default: `''`

Specify a config file to load.

JupyterApp.generate_config [Bool] Default: `False`

Generate default config file.

ZMQTerminalInteractiveShell.banner [Unicode] Default: `'Jupyter console {version}\\n\\n{kernel_banner}'`

Text to display before the first prompt. Will be formatted with variables {version} and {kernel_banner}.

ZMQTerminalInteractiveShell.callable_image_handler [Any] Default: `None`

Callable object called via 'callable' image handler with one argument, *data*, which is *msg["content"]["data"]* where *msg* is the message from iopub channel. For example, you can find base64 encoded PNG data as *data['image/png']*. If your function can't handle the data supplied, it should return *False* to indicate this.

ZMQTerminalInteractiveShell.editing_mode [Unicode] Default: `'emacs'`

Shortcut style to use at the prompt. 'vi' or 'emacs'.

ZMQTerminalInteractiveShell.highlight_matching_brackets [Bool] Default: `True`

Highlight matching brackets.

ZMQTerminalInteractiveShell.highlighting_style [Unicode] Default: `''`

The name of a Pygments style to use for syntax highlighting

ZMQTerminalInteractiveShell.highlighting_style_overrides [Dict] Default: `{}`

Override highlighting format for specific tokens

ZMQTerminalInteractiveShell.history_load_length [Int] Default: `1000`

How many history items to load into memory

ZMQTerminalInteractiveShell.image_handler ['PIL'|'stream'|'tempfile'|'callable'] Default: `'PIL'`

Handler for image type output. This is useful, for example, when connecting to the kernel in which pylab inline backend is activated. There are four handlers defined. 'PIL': Use Python Imaging Library to popup image; 'stream': Use an external program to show the image. Image will be fed into the STDIN of the program. You will need to configure *stream_image_handler*; 'tempfile': Use an external program to show the image. Image will be saved in a temporally file and the program is called with the temporally file. You will need to configure *tempfile_image_handler*; 'callable': You can set any Python callable which is called with the image data. You will need to configure *callable_image_handler*.

ZMQTerminalInteractiveShell.include_other_output [Bool] Default: `False`

Whether to include output from clients other than this one sharing the same kernel.

Outputs are not displayed until enter is pressed.

ZMQTerminalInteractiveShell.kernel_is_complete_timeout [Float] Default: `1`

Timeout (in seconds) for giving up on a kernel's is_complete response.

If the kernel does not respond at any point within this time, the kernel will no longer be asked if code is complete, and the console will default to the built-in is_complete test.

ZMQTerminalInteractiveShell.kernel_timeout [Float] Default: `60`

Timeout for giving up on a kernel (in seconds).

On first connect and restart, the console tests whether the kernel is running and responsive by sending kernel_info_requests. This sets the timeout in seconds for how long the kernel can take before being presumed dead.

ZMQTerminalInteractiveShell.mime_preference [List] Default: `['image/png', 'image/jpeg', 'image/svg+xml']`

Preferred object representation MIME type in order. First matched MIME type will be used.

ZMQTerminalInteractiveShell.other_output_prefix [Unicode] Default: `'[remote] '`

Prefix to add to outputs coming from clients other than this one.

Only relevant if include_other_output is True.

ZMQTerminalInteractiveShell.simple_prompt [Bool] Default: `False`

Use simple fallback prompt. Features may be limited.

ZMQTerminalInteractiveShell.stream_image_handler [List] Default: `[]`

Command to invoke an image viewer program when you are using 'stream' image handler. This option is a list of string where the first element is the command itself and reminders are the options for the command. Raw image data is given as STDIN to the program.

ZMQTerminalInteractiveShell.tempfile_image_handler [List] Default: `[]`

Command to invoke an image viewer program when you are using 'tempfile' image handler. This option is a list of string where the first element is the command itself and reminders are the options for the command. You can use {file} and {format} in the string to represent the location of the generated image file and image format.

ZMQTerminalInteractiveShell.true_color [Bool] Default: `False`

Use 24bit colors instead of 256 colors in prompt highlighting. If your terminal supports true color, the following command should print 'TRUECOLOR' in orange: printf "x1b[38;2;255;100;0mTRUECOLORx1b[0mn"

ZMQTerminalInteractiveShell.use_kernel_is_complete [Bool] Default: `True`

Whether to use the kernel's is_complete message handling. If False, then the frontend will use its own is_complete handler.

KernelManager.autorestart [Bool] Default: `True`

Should we autorestart the kernel if it dies.

KernelManager.kernel_cmd [List] Default: `[]`

DEPRECATED: Use kernel_name instead.

The Popen Command to launch the kernel. Override this if you have a custom kernel. If kernel_cmd is specified in a configuration file, Jupyter does not pass any arguments to the kernel, because it cannot make any assumptions about the arguments that the kernel understands. In particular, this means that the kernel does not receive the option –debug if it given on the Jupyter command line.

KernelManager.shutdown_wait_time [Float] Default: `5.0`

Time to wait for a kernel to terminate before killing it, in seconds.

KernelRestarter.debug [Bool] Default: `False`

Whether to include every poll event in debugging output.

Has to be set explicitly, because there will be *a lot* of output.

KernelRestarter.random_ports_until_alive [Bool] Default: `True`

Whether to choose new random ports when restarting before the kernel is alive.

KernelRestarter.restart_limit [Int] Default: `5`

The number of consecutive autorestarts before the kernel is presumed dead.

KernelRestarter.time_to_dead [Float] Default: `3.0`

Kernel heartbeat interval in seconds.

Session.buffer_threshold [Int] Default: `1024`

Threshold (in bytes) beyond which an object's buffer should be extracted to avoid pickling.

Session.check_pid [Bool] Default: `True`

Whether to check PID to protect against calls after fork.

This check can be disabled if fork-safety is handled elsewhere.

Session.copy_threshold [Int] Default: `65536`

Threshold (in bytes) beyond which a buffer should be sent without copying.

Session.debug [Bool] Default: `False`

Debug output in the Session

Session.digest_history_size [Int] Default: `65536`

The maximum number of digests to remember.

The digest history will be culled when it exceeds this value.

Session.item_threshold [Int] Default: `64`

The maximum number of items for a container to be introspected for custom serialization. Containers larger than this are pickled outright.

Session.key [CBytes] Default: `b''`

execution key, for signing messages.

Session.keyfile [Unicode] Default: `''`

path to file containing execution key.

Session.metadata [Dict] Default: `{}`

Metadata dictionary, which serves as the default top-level metadata dict for each message.

Session.packer [DottedObjectName] Default: `'json'`

The name of the packer for serializing messages. Should be one of 'json', 'pickle', or an import name for a custom callable serializer.

Session.session [CUnicode] Default: `''`

The UUID identifying this session.

Session.signature_scheme [Unicode] Default: `'hmac-sha256'`

The digest scheme used to construct the message signatures. Must have the form 'hmac-HASH'.

Session.unpacker [DottedObjectName] Default: `'json'`

The name of the unpacker for unserializing messages. Only used with custom functions for *packer*.

Session.username [Unicode] Default: `'username'`

Username for the Session. Default is your system username.

CHAPTER 2

Making a release as a maintainer

This document guides a maintainer through creating a release of the Jupyter console.

2.1 Clean the repository

Remove all non-tracked files with:

```
git clean -xfdi
```

This will ask you for confirmation before removing all untracked files. Make sure the `dist/` folder is clean and does not contain any stale builds from previous attempts.

2.2 Create the release

1. Set Environment variables

   Set environment variables to document current release version, and git tag:

   ```
   VERSION=4.1.0
   ```

2. Update version number in `jupyter_console/_version.py`. Make sure that a valid PEP 440 version is being used.

3. Commit and tag the release with the current version number:

   ```
   git commit -am "release $VERSION"
   git tag $VERSION
   ```

4. You are now ready to build the `sdist` and `wheel`:

   ```
   python setup.py sdist --formats=gztar
   python setup.py bdist_wheel
   ```

5. You can now test the `wheel` and the `sdist` locally before uploading to PyPI. Make sure to use twine to upload the archives over SSL.

```
twine upload dist/*
```

6. If all went well, change the `jupyter_console/_version.py` to the next release.
7. Push directly on master, not forgetting to push `--tags` too.

CHAPTER 3

Changes in Jupyter console

A summary of changes in Jupyter console releases.

3.1 5.3

- Highlight matching parentheses. PR #147
- The config option `JupyterConsoleApp.confirm_exit` replaces `ZMQTerminalInteractiveShell.confirm_exit`, to avoid redundancy. PR #141.

3.2 5.2

- When using a kernel that the console did not start, exiting with Ctrl-D now leaves it running. PR #127
- Added Ctrl-\ shortcut to quit the console. PR #130
- Input prompt numbers are now updated when another frontend has executed code in the same kernel. PR #119
- Fix setting next input with newer versions of prompt_toolkit. PR #123
- Ensure history entries are unicode, not bytes, on Python 2. PR #122

3.3 5.1

- New `ZMQTerminalInteractiveShell.true_color` config option to use 24-bit colour.
- New `ZMQTerminalInteractiveShell.confirm_exit` config option to turn off asking 'are you sure' on exit.
- New `--simple-prompt` flag to explicitly use the fallback mode without prompt_toolkit.
- Fixed executing an empty input.

- Fixed formatting for code and outputs from other frontends executing code.
- Avoid using functions which will be removed in IPython 6.

3.4 5.0

3.4.1 5.0.0

Interactive Shell architecture

- Disinherit shell class from IPython Interactive Shell (PR #68). This separates jupyter_console's `ZMQTerminalInteractiveShell` from IPython's `TerminalInteractiveShell` and `InteractiveShell` classes.
- Update SIGINT handler to not use the old interactive API shell. PR #80

Image Handling improvement

- use PIL as default image handler PR #79
- better indication of whether image data was handled PR #77

Prompts improvement

- use prompt_toolkit 1.0 PR #74
- don't use prompt_manager PR #75
- remove `colors_force` flag that have no effects: PR #88

3.5 4.1

3.5.1 4.1.1

- fix for readline history
- don't confuse sys.path with virtualenvs

3.5.2 4.1.0

- readline/completion fixes
- use is_complete messages to determine if input is complete (important for non-Python kernels)
- fix: 4.0 was looking for jupyter_console_config in IPython config directories, not Jupyter

3.6 4.0

3.6.1 4.0.3

- fix `jupyter console --generate-config`

3.6.2 4.0.2

- setuptools fixes for Windows

3.6.3 4.0.0

First release as a standalone package.