Sphinx showcase

Release 0.1.0

Certora, Inc

Feb 06, 2024

Contents:

This chapter describes the most useful Sphinx directives and roles.

For additional information see:

- reStructuredText Primer
- Sphinx Directives

1 Standard markup

1.1 Basic inline markup

Fonts

Listing 1: rst

```
* We can use *italic* and **bold**.
* using ``double quotes`` provides a code style.
```

Rendered as:

- We can use *italic* and **bold**.
- using `double quotes` provides a code style.

Headings

For a full list and explanation, see: reStructuredText Primer - Sections.

Listing 2: Headings conventions

```
Section heading
===============

Sub-section
-----------

Sub sub-section
^^^^^^^^^^^^^^^

Even lower level
"""""""""""""""""
```

Note there are no levels assigned to particular heading characters. Sphinx deduces the levels in each `.rst` file.

Horizontal rule

Use four dashes `----` (with empty lines above and below) to get a horizontal rule like the one below.

1.2 Lists

Bullet lists

Listing 3: rst

```
* Bullet item
* Can contain nested lists

  * Like this
  * And this
```

Rendered as:

- Bullet item
- Can contain nested lists
  - Like this
  - And this

Numbered lists

Listing 4: rst

```
#. Numbered list
#. Second item
```

Rendered as:

1. Numbered list
2. Second item

Definition list

Listing 5: rst

```
Some term
   Followed by definition of the term, which must be indented.

   The definition can even consist of multiple paragraphs.

Second term
   Description of the second term.
```

Rendered as:

Some term
: Followed by definition of the term, which must be indented.

  The definition can even consist of multiple paragraphs.

Second term
: Description of the second term.

1.3 Links

External links

Listing 6: rst

```
* Simple external link: `Certora <https://www.certora.com/>`_ (note underscore suffix)
* Using predefined link: `Rick Astley`_ (defined below, again note underscores)

.. _Rick Astley: https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

Rendered as:

- Simple external link: Certora (note underscore suffix)
- Using predefined link: Rick Astley (defined below, again note underscores)

Embedding a Youtube video

Listing 7: rst

```
.. youtube:: VGSsPIsbb6U
   :align: center
```

Rendered as:

https://youtu.be/VGSsPIsbb6U

Internal links

Link anywhere inside the documentation.

Listing 8: rst

```
.. _my-reference-label:

Cross-reference inside documentation
""""""""""""""""""""""""""""""""""""

Set up a label ``.. _my-reference-label`` as shown above.
Note underscore prefix in the label name .
To reference use the ``:ref:`` directive like so:  :ref:`my-reference-label`.
```

Rendered as:

Cross-reference inside documentation

Set up a label `.. _my-reference-label` as shown above. Note underscore prefix in the label name . To reference use the `:ref:` directive like so: *Cross-reference inside documentation*.

Note: This example was taken from Cross-referencing arbitrary locations.

Link to code file on Github

Link to a code file in the *code* folder using the `:clink:` role. The link will be either to Github or to local file, depending on the value of `link_to_github` variable in the `source/conf.py` file. The *code* folder is defined by the `code_path` variable in the `source/conf.py` file. For complete documentation, see codelink_extension.

Listing 9: Syntax

```
:clink:`Optional name <relative-path-from-code-dir>`
```

For example:

Listing 10: rst

```
* Reference to a folder: :clink:`Voting folder <voting>`
* Reference to a file: :clink:`Voting_solution.spec <voting/Voting_solution.spec>`
* Reference without text: :clink:`voting/Voting_solution.spec`
```

Rendered as:

- Reference to a folder: Voting folder
- Reference to a file: Voting_solution.spec
- Reference without text: voting/Voting_solution.spec

2 Code blocks

2.1 Best practice

It is best to include a code-block from a spec or Solidity file that is part of a regtest. This will ensure that you will be alerted if there are any breaking changes. Use the directives described in *From external file*.

Using `includecvl` (see *Including CVL elements* below) has the added benefit that it is protected against changes to the code file itself. Added or removed lines will not affect it.

2.2 In-place code

Code-block

You can insert a *CVL* code block in-place, using the `code-block` directive, as shown below. The same directive can be used for other languages, such as Solidity.

Listing 11: rst

```
.. code-block:: cvl

   methods {
       function balanceOf(address) external returns (uint256) envfree;
   }

   rule testBalance(address user) {
       assert balanceOf(user) > 0;
   }
```

Rendered as:

```
methods {
    function balanceOf(address) external returns (uint256) envfree;
}

rule testBalance(address user) {
    assert balanceOf(user) > 0;
}
```

Additional features, such as line numbers and emphasized lines are demonstrated below. You can find all the options available at: code-block directive.

Listing 12: rst

```
.. code-block:: cvl
   :linenos:
   :lineno-start: 7
   :emphasize-lines: 10,17
   :caption: CVL2 code example

   methods
   {
       function DataWarehouse.getRegisteredSlot(
           bytes32 blockHash,
           address account,
           bytes32 slot
       ) external returns (uint256) => _getRegisteredSlot(blockHash, account, slot);
   }

   ghost mapping(address => uint256) _exchangeRateSlotValue;

   function _getRegisteredSlot(
       bytes32 blockHash,
       address account,
       bytes32 slot
   ) returns uint256 {
       return _exchangeRateSlotValue[account];
   }
```

Rendered as:

Listing 13: CVL2 code example

```
7  methods
8  {
9      function DataWarehouse.getRegisteredSlot(
```

(continues on next page)

(continued from previous page)

```
        bytes32 blockHash,
        address account,
        bytes32 slot
    ) external returns (uint256) => _getRegisteredSlot(blockHash, account, slot);
}

ghost mapping(address => uint256) _exchangeRateSlotValue;

function _getRegisteredSlot(
    bytes32 blockHash,
    address account,
    bytes32 slot
) returns uint256 {
    return _exchangeRateSlotValue[account];
}
```

Inline CVL and solidity

You can add inline *CVL* code using the `:cvl:` role, and inline Solidity using the `:solidity:` role. For example, the following paragraph:

```
Type casting between integers in *CVL* has two different forms, :cvl:`assert_uint256`
and :cvl:`require_uint256`. In the :solidity:`constructor(uin256 x)` ...
```

Rendered as:

Type casting between integers in *CVL* has two different forms, **`assert_uint256`** and **`require_uint256`**. In the **`constructor`**`(uin256 x)` ...

2.3 From external file

Including CVL elements

Use the `cvlinclude` directive to include CVL elements *by name*. This is the preferred way to include rules, invariants, ghosts and the methods block. Complete documentation is available at includecvl_extension.

Example

```
.. cvlinclude:: /../../code/voting/Voting_solution.spec
   :cvlobject: numVoted onlyLegalVotedChanges sumResultsEqualsTotalVotes
   :caption: Voting rules
```

Rendered as:

- If the path to the spec file is absolute, it is considered as relative to the `/source/` directory.
- The `:cvlobject:` option accepts names of CVL elements (rule, invariant and ghosts). To include the `methods` block, add `methods` to these names. The elements will be shown in the order they are given.

Note: Hooks are not supported (since they are not supported by the `CVLDoc` package). Use `literalinclude` below.

Including any code

Use the `literalinclude` directive to include code from an external file. As above, providing an absolute path is taken as relative to the `/source/` directory. For all possible options of `literalinclude`, see the literalinclude directive.

Important: An alternative to using line numbers when including code are the `:start-after:`, `:start-at:`, `:end-before:`, and `:end-at:` options. These accept string, which they match to find the desired lines.

For example:

```
.. literalinclude:: /../../code/voting/Voting.sol
   :language: solidity
   :lines: 4-
   :emphasize-lines: 4-6
```

Rendered as:

3 Indexing and glossary

3.1 Indexing

To add terms to the genindex, place an appropriate `.. index` directive before the part you wish to index. See Sphinx - index directive for a comprehensive description of this directive, here are some simple examples.

Simple indexing

The following will create three index entries.

```
.. index:: municipality, town, city
```

Adding single values

```
.. index::
   single: propositional logic
   single: logic; propositional
```

This will create two index entries, the first as "propositional logic" and the second will be a sub-index under "logic".

Adding reference labels to indexes

Use the `:name:` option for adding a label that can be used with `:ref:`. For example:

```
.. index:: formal
   :name: intro_to_formal

   Introduction to formal verification
   -----------------------------------

   See :ref:`intro_to_formal` ...
```

Inline indexing

You can add index entries inline. Here is an example from Sphinx - index directive:

```
This is a normal reST :index:`paragraph` that contains several
:index:`index entries <pair: index; entry>`.
```

3.2 Glossary

For complete documentation on the `glossary` directive see Sphinx - Glossary.

Creating a glossary

Create a glossary using the `.. glossary::` directive, followed by a *Definition list* of the desired terms. A term can have several names, as shown in the following example.

Listing 14: rst

```
.. glossary::

   CVL
      The Certora Veification Language, used for writing specs for Solidity contracts.

   Prover
   Certora Prover
      The tool used for verifying specs written in :term:`CVL`.
```

Rendered as:

CVL
: The Certora Veification Language, used for writing specs for Solidity contracts.

Prover
Certora Prover
: The tool used for verifying specs written in *CVL*.

Referencing a glossary term

Use the `:term:` role to refer to a glossary term, for example:

Listing 15: rst

```
* Simple reference such as :term:`CVL`
* Showing alternative text like :term:`The Prover <Prover>`
```

Rendered as:

- Simple reference such as *CVL*
- Showing alternative text like *The Prover*

4 Comments and TODOs

4.1 RestructuredText comments

```
.. This is a comment in RestructuredText, the entire paragraph will be ignored
   by sphinx. Just note the indentation.
```

4.2 Development build

We can have content that is visible only in *dev-build* mode. To enable dev-build mode, add `-t is_dev_build` to the `sphinx-build` command (see build_html and generating_pdf). For example:

```
sphinx-build -b html docs/source/ docs/build/html -t is_dev_build
```

Note: In dev-build the html title (on the side bar) will have "- Development" added to it. This behavior can be modified in the `/source/conf.py` file.

Contents for dev-build only

To produce contents that will appear only in dev-build, use the `.. only` directive, like this:

Listing 16: rst

```
.. only:: is_dev_build

   The following will only be included in dev builds.
```

Rendered as:

The following will only be included in dev builds.

TODOs

TODO comments will only appear in dev-build. To add a TODO comment:

Listing 17: rst

```
.. todo:: This is an example of a TODO comment, it can also have several paragraphs.
```

Rendered as:

Todo: This is an example of a TODO comment, it can also have several paragraphs.

To create a list containing all the TODO comments:

```
.. todolist::
```

5 Admonitions

Admonitions are used for warnings, info and so on. Here is a collection of admonitions examples.

```
.. note::

   For providing notes and information to the user.

   The admonition can contain several paragraphs and also other elements, like:

   * Lists
   * Math

.. attention::

   Pay attention,

.. important::

   For marking very important things.

.. tip::

   Tips for the reader.

.. hint::

   Provide hints.

.. warning::

   Warn about dangerous things.

.. seealso::

   For providing more references.

.. admonition:: General admonition - any title you want

   The freedom to admonish.
```

Rendered as:

Note: For providing notes and information to the user.

The admonition can contain several paragraphs and also other elements, like:

- Lists
- Math

Attention: Pay attention,

Important: For marking very important things.

Tip: Tips for the reader.

Hint: Provide hints.

Warning: Warn about dangerous things.

See also:

For providing more references.

General admonition - any title you want

The freedom to admonish.

6 Panels

The panels use the sphinx-design extension. Follow the link for more details.

6.1 Single card

```
.. card:: Card Title

   Content of the card. See
   `sphinx-design <https://sphinx-design.readthedocs.io/en/rtd-theme/index.html>`_
   for more details.
```

Rendered as:

Card Title Content of the card. See sphinx-design for more details.

6.2 Grid with two cards

```
.. grid:: 2

    .. grid-item-card::  Title 1

        Left card

    .. grid-item-card::  Title 2

        Right card
```

Rendered as:

Title 1 Left card

Title 2 Right card

Placing code side by side

Note the limited width of the columns!

Spec

Listing 18: Invariant

```
invariant totalIsBiggest(address user)
    balanceOf(user) <= totalBalance();
```

Solidity

Listing 19: Solidity

```
function balanceOf(
  address user
) external view returns (bool) {
  return _balances[user];
}
```

6.3 Drop-down

Drop-down content is useful for providing hidden hints. Here is a simple drop-down:

```
.. dropdown:: Dropdown title
   :animate: fade-in-slide-down

    Dropdown content, for example an important hint.

    See `sphinx-design - dropdowns
    <https://sphinx-design.readthedocs.io/en/rtd-theme/dropdowns.html>`_ for more␣
↪options.
```

Rendered as:

Dropdown title

Dropdown content, for example an important hint.

See sphinx-design - dropdowns for more options.

7 Using Latex

7.1 In-line math

For inline math use the `:math:` role. For example:

```
Let :math:`\mathcal{C}` be the category of groups and :math:`f: G \to H` be a
morphism in :math:`\mathcal{C}`.
```

Rendered as:

Let $\mathcal{C}$ be the category of groups and $f : G \to H$ be a morphism in $\mathcal{C}$.

7.2 Centered math

Use the `math` directive, as shown below. See Directives - math for additional options and examples.

```
.. math::

   (a + b)^2  &=  (a + b)(a + b) \\
              &=  a^2 + 2ab + b^2
```

Rendered as:

$$\begin{aligned}(a + b)^2 &= (a + b)(a + b) \\ &= a^2 + 2ab + b^2\end{aligned}$$

7.3 Advanced use

Here is an example of showing a conditional function.

Listing 20: Conditional function in Latex

```
.. math::
   :nowrap:

   \begin{equation}
   f(x) =
   \begin{cases}
        0  & \text{if } x \leq 0 \\
        x^2 & \text{otherwise}
   \end{cases}
   \end{equation}
```

Rendered as:

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x^2 & \text{otherwise} \end{cases} \tag{1}$$

Note: When using the `.. math::` directive, Sphinx will wrap the latex code inside the Latex `split` environment before rendering it. Using the `:nowrap:` option disables this behavior.

For example, the code from *Centered math* is rendered as the following Latex code:

```
\begin{split}
    (a + b)^2  &=  (a + b)(a + b) \\
               &=  a^2 + 2ab + b^2
\end{split}
```

---

8 Miscellaneous

8.1 Tables

There are several ways to add tables in reStructuredText, there are described in

- reStructuredText Primer - Tables
- CSV Tables
- List Tables

Here is an example of a *list table*.

```
.. list-table:: Table title
   :header-rows: 1

   * - Column Header
     - 2nd Column Header
     - 3rd Column Header

   * - Row 1 Column 1 item
     - Row 1 Column 2 item
     - An item

   * - An item
     - Row 2 Column 2 item
     - Row 2 Column 3 item
```

Rendered as:

Table 1: Table title

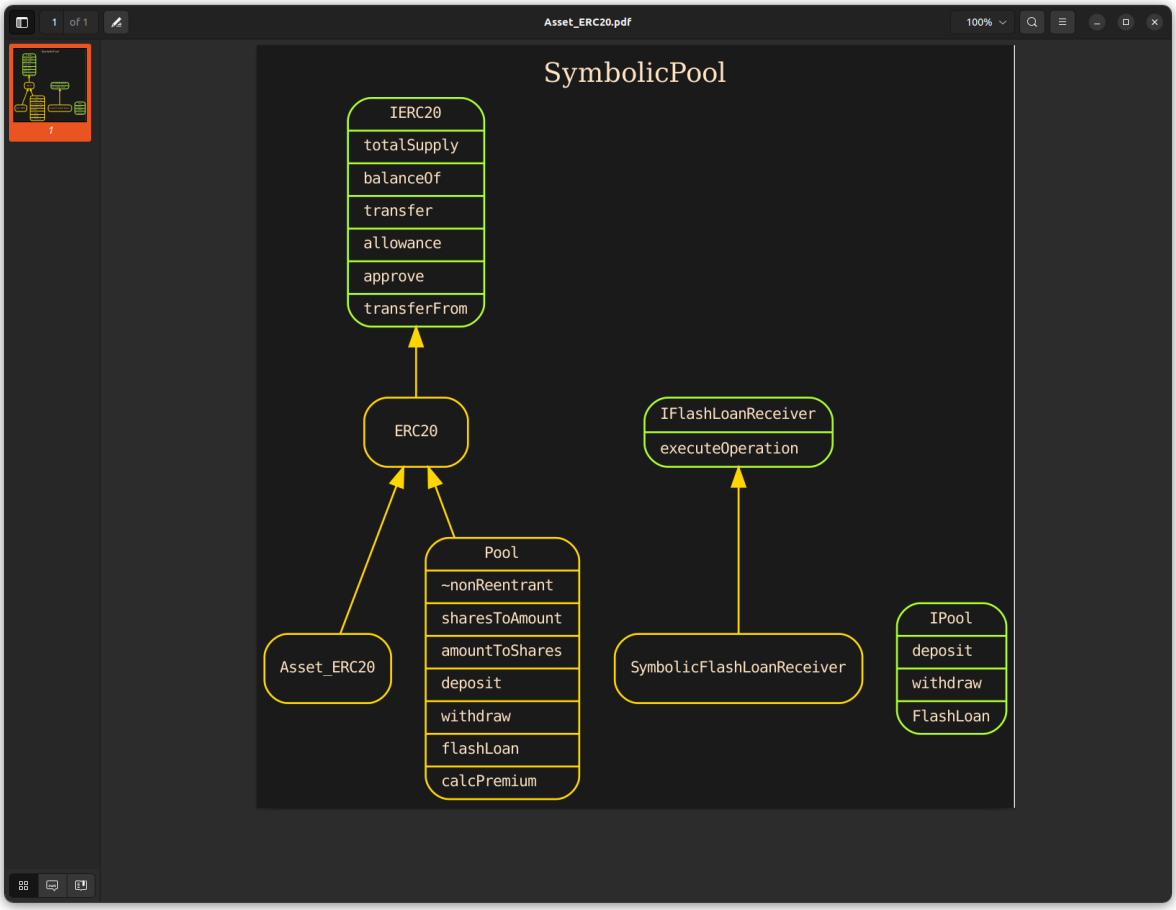
Column Header	2nd Column Header	3rd Column Header
Row 1 Column 1 item	Row 1 Column 2 item	An item
An item	Row 2 Column 2 item	Row 2 Column 3 item

8.2 Adding an image

To insert an image or picture use the `.. image` directive, as shown below. The specified path to the image `images/symbolic_pool_diagram.png` is relative to the file containing the directive.

```
.. image:: images/symbolic_pool_diagram.png
   :alt: This text will be displayed if the image is broken
```

Rendered as:



Notes

The image path
: A relative path should be relative to the `.rst` file. An absolute path is treated as relative to the top `source/` directory. See Sphinx Image Directive for more on this.

Additional options
: Options, such as alternative text for missing images and scaling, are described in Docutils Image Directive.

8.3 Adding a video clip

To add a video clip file we use the sphinxcontrib-video extension. Note that the preferred folder to place the video file is the `source/_static/` folder. For example:

```
.. video:: ../_static/lesson4_invariants/ball_game/InvariantsClip_subtitles.mp4
   :alt: The text shown when the video cannot be displayed
   :height: 250
```

Rendered as:

See also:

See sphinxcontrib-video Quickstart for additional options.

Combining closed captions

You cannot use a separate file for the closed captions (subtitles). Instead you must embed the closed captions inside the video itself.

Here is one recipe to include a closed captions file in your video. Suppose you have an mp4 video `InvariantsClip.mp4` and a closed captions file named `InvariantsClip.srt`, you can combine them using the FFmpeg package with the following command:

```
ffmpeg -i InvariantsClip.mp4 -vf subtitles=InvariantsClip.srt InvariantsClip_
↪subtitles.mp4
```

Todo: Missing topics to add:

- table ot contents (mainly the `hidden` option)
- tabs (from sphinx-design)
- footnotes
- `.. rubric`, `.. centered` and `.. hlist`

Index