



# Sphinx showcase

*Release 0.1.0*

**Certora, Inc**

**Mar 10, 2024**

## Contents:

<b>1</b>	<b>Standard markup</b>	<b>2</b>
1.1	Basic inline markup . . . . .	2
1.2	Lists . . . . .	3
1.3	Links . . . . .	5
<b>2</b>	<b>Code blocks</b>	<b>7</b>
2.1	Best practice . . . . .	7
2.2	In-place code . . . . .	7
2.3	From external file . . . . .	10
<b>3</b>	<b>Indexing and glossary</b>	<b>12</b>
3.1	Indexing . . . . .	12
3.2	Glossary . . . . .	14
<b>4</b>	<b>Comments and TODOs</b>	<b>15</b>
4.1	RestructuredText comments . . . . .	15
4.2	Development build . . . . .	16
<b>5</b>	<b>Admonitions</b>	<b>17</b>
<b>6</b>	<b>Panels</b>	<b>19</b>
6.1	Single card . . . . .	19
6.2	Grid with two cards . . . . .	20
6.3	Drop-down . . . . .	21
<b>7</b>	<b>Using Latex</b>	<b>22</b>
7.1	In-line math . . . . .	22
7.2	Centered math . . . . .	22
7.3	Advanced use . . . . .	23
<b>8</b>	<b>Miscellaneous</b>	<b>24</b>
8.1	Tables . . . . .	24
8.2	Adding an image . . . . .	24
8.3	Adding a video clip . . . . .	25
	<b>Index</b>	<b>27</b>

---

This chapter describes the most useful Sphinx directives and roles.

If you're markdown files (.md) – these use the [MyST](#) package, and the MyST examples are what you need. Those using reStructuredText files (.rst) should refer to the reStructuredText examples.

For additional information see:

- [reStructuredText Primer](#)
- [Sphinx Directives](#)
- [MyST](#)

## 1 Standard markup

### 1.1 Basic inline markup

#### Fonts

##### MyST (.md)

```
* We can use _italic_ and *bold*.  
* We can even have a `mono font`.
```

##### reStructuredText (.rst)

```
* We can use *italic* and **bold**.  
* We can even have a `mono font`.
```

Rendered as:

- We can use *italic* and **bold**.
- We can even have a `mono font`.

#### Headings

##### MyST (.md)

Listing 1: Headings conventions

```
# Top level

## Second level

### Third level
```

## reStructuredText (.rst)

For a full list and explanation, see: [reStructuredText Sections](#). Note there are no levels assigned to particular heading characters. Sphinx deduces the levels in each .rst file.

Listing 2: Headings conventions

```
Section heading
=====

Sub-section
-----

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

Even lower level
^^^^^^^^^^^^^^^^
```

## 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

```
* Bullet item
* Can contain nested lists

  * Like this
  * And this
```

Rendered as:

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

## Numbered lists

### MyST (.md)

```
1. Numbered list
2. Second item
```

### reStructuredText (.rst)

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

Rendered as:

1. Numbered list
2. Second item

## Definition list

### MyST (.md)

```
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.
```

### reStructuredText (.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

#### MyST (.md)

```
* Simple external link: [Certora](https://www.certora.com/).
* Using predefined link: [Rick Astley][RickRolled] (defined below)

[RickRolled]: https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

#### reStructuredText (.rst)

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

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

Rendered as:

- Simple external link: [Certora](https://www.certora.com/).
  - Using predefined link: [Rick Astley](https://www.youtube.com/watch?v=dQw4w9WgXcQ) (defined below)
- 

### Embedding a Youtube video

#### MyST (.md)

```
```{youtube} VGSsPIsbb6U
:align: center
```
```

#### reStructuredText (.rst)

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

Rendered as:

<https://youtu.be/VGSsPIsbb6U>

---

### Internal links

Link anywhere inside the documentation.

## MyST (.md)

See [MyST Cross-referencing](#).

## reStructuredText (.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``.

## Link to code file on Github

Link to a code file 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.

Absolute paths will be considered as relative to the *absolute code path* – see `code_path_variable`. For complete documentation, see `codelink_extension`.

The basic syntax is:

## MyST (.md)

Listing 3: Syntax

```
{clink}`Optional name <relative-path-to-code-file>`  
{clink}`Optional name <absolute path relative to absolute code path>`
```

## reStructuredText (.rst)

Listing 4: Syntax

```
:clink:`Optional name <relative-path-to-code-file>`  
:clink:`Optional name <absolute path relative to absolute code path>`
```

For example:

## MyST (.md)

```
* 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`
```

## reStructuredText (.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*.

Including source code for CVL elements using the `includecvl` directive (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.

#### MyST (.md)

```
```{code-block} cvl

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

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

## reStructuredText (.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](#).

## MyST (.md)

```
```{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];
}
```
```



## reStructuredText (.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 5: CVL2 code example

```
7 methods
8 {
9     function DataWarehouse.getRegisteredSlot(
10         bytes32 blockHash,
11         address account,
12         bytes32 slot
13     ) external returns (uint256) => _getRegisteredSlot(blockHash, account, slot);
14 }
15
16 ghost mapping(address => uint256) _exchangeRateSlotValue;
17
18 function _getRegisteredSlot(
19     bytes32 blockHash,
20     address account,
21     bytes32 slot
22 ) returns uint256 {
23     return _exchangeRateSlotValue[account];
24 }
```

## Inline CVL and solidity

### MyST (.md)

You can add inline *CVL* code using the `:cvl:` role, and inline Solidity using the `:solidity:` role. To do so you must first define these roles at the top of your `.md` file, like so:

```
```{role} cvl(code)
:language: cvl
```

```{role} solidity(code)
:language: solidity
```
```

Now we can use them, as in the following example:

```
Type casting between integers in *CVL* has two different forms,
{cvl}`assert_uint256` and {cvl}`require_uint256`.
In the {solidity}`constructor(uint256 x)` ...
```

### reStructuredText (.rst)

You can add inline *CVL* code using the `:cvl:` role, and inline Solidity using the `:solidity:` role. These roles are defined in the `conf.py` file. 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(uint256 x)` ...
```

Rendered as:

Type casting between integers in *CVL* has two different forms, **assert\_uint256** and **require\_uint256**. In the **constructor**(uint256 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

#### MyST (.md)

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

## reStructuredText (.rst)

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

Rendered as:

Listing 6: Voting rules

```
/// @title Count the number of times `_hasVoted` been written to
ghost mathint numVoted {
    init_state axiom numVoted == 0;
}

/// @title No illegal changes to `_hasVoted`
invariant onlyLegalVotedChanges()
    !illegalStore;

/// @title Sum of voter in favor and against equals total number of voted
invariant sumResultsEqualsTotalVotes()
    votesInFavor() + votesAgainst() == to_mathint(totalVotes());
```

- 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:

## MyST (.md)

```
```{literalinclude} ../../../../code/voting/Voting_solution.spec
:language: solidity
:lines: 4-
:emphasize-lines: 4-6
```
```

## reStructuredText (.rst)

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

Rendered as:

```
contract Voting {

    mapping(address => bool) internal _hasVoted;

    uint256 public votesInFavor;
    uint256 public votesAgainst;
    uint256 public totalVotes;

    function vote(bool isInFavor) public {
        require(!_hasVoted[msg.sender]);
        _hasVoted[msg.sender] = true;

        totalVotes += 1;
        if (isInFavor) {
            votesInFavor += 1;
        } else {
            votesAgainst += 1;
        }
    }

    function hasVoted(address voter) public view returns (bool) {
        return _hasVoted[voter];
    }
}
```

## 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.

### MyST (.md)

---

**Todo:** Test this one!

---

```
```{index} municipality, town, city
```
```

### reStructuredText (.rst)

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

## Adding single values

### MyST (.md)

```
```{eval-rst}
.. index::
   single: propositional logic
   single: logic; propositional
```
```

### reStructuredText (.rst)

```
.. 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:

### MyST (.md)

```
```{eval-rst}
.. index::
   :name: intro_to_formal
```

## Introduction to formal verification

See {ref}`intro_to_formal` ...
```

## reStructuredText (.rst)

```
.. index::
   :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](#):

## MyST (.md)

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

## reStructuredText (.rst)

```
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.

## MyST (.md)

```
{.glossary}
CVL
: The Certora Verification Language, used for writing specs for Solidity contracts.

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

## reStructuredText (.rst)

```
.. glossary::

    CVL
        The Certora Verification 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 Verification 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:

## MyST (.md)

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

## reStructuredText (.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:

#### MyST (.md)

```
```{only} is_dev_build
The following will only be included in dev builds.
```
```

#### reStructuredText (.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:

#### MyST (.md)

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



## reStructuredText (.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:

## MyST (.md)

```
```{todolist}
```
```

## reStructuredText (.rst)

```
.. todolist::
```

# 5 Admonitions

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

## MyST (.md)

```
```{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.
```
```

(continues on next page)

```
```{warning}
Warn about dangerous things.
```

```{seealso}
For providing more references.
```

```{admonition} General admonition - any title you want
The freedom to admonish.
```
```

## reStructuredText (.rst)

```
.. 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

**MyST (.md)**

```
```{card} Card Title

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

### reStructuredText (.rst)

```
.. 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

### MyST (.md)

```
````{grid} 2

````{grid-item-card} Title 1

Left card
````

````{grid-item-card} Title 2

Right card
````
````
```

### reStructuredText (.rst)

```
.. 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 7: Invariant

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

Solidity

Listing 8: 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:

### MyST (.md)

```
```{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.
```
```

### reStructuredText (.rst)

```
.. 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:

#### MyST (.md)

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

#### reStructuredText (.rst)

```
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  $f : G \rightarrow H$  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.

#### MyST (.md)

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

#### reStructuredText (.rst)

```
.. 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.

### MyST (.md)

Listing 9: 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}
```
```

### reStructuredText (.rst)

Listing 10: 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:

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

**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

**Attention:** This section is currently only available for reStructuredText. See [MyST Roles and Directives](#) for how to use these directives in MyST.

### 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 |

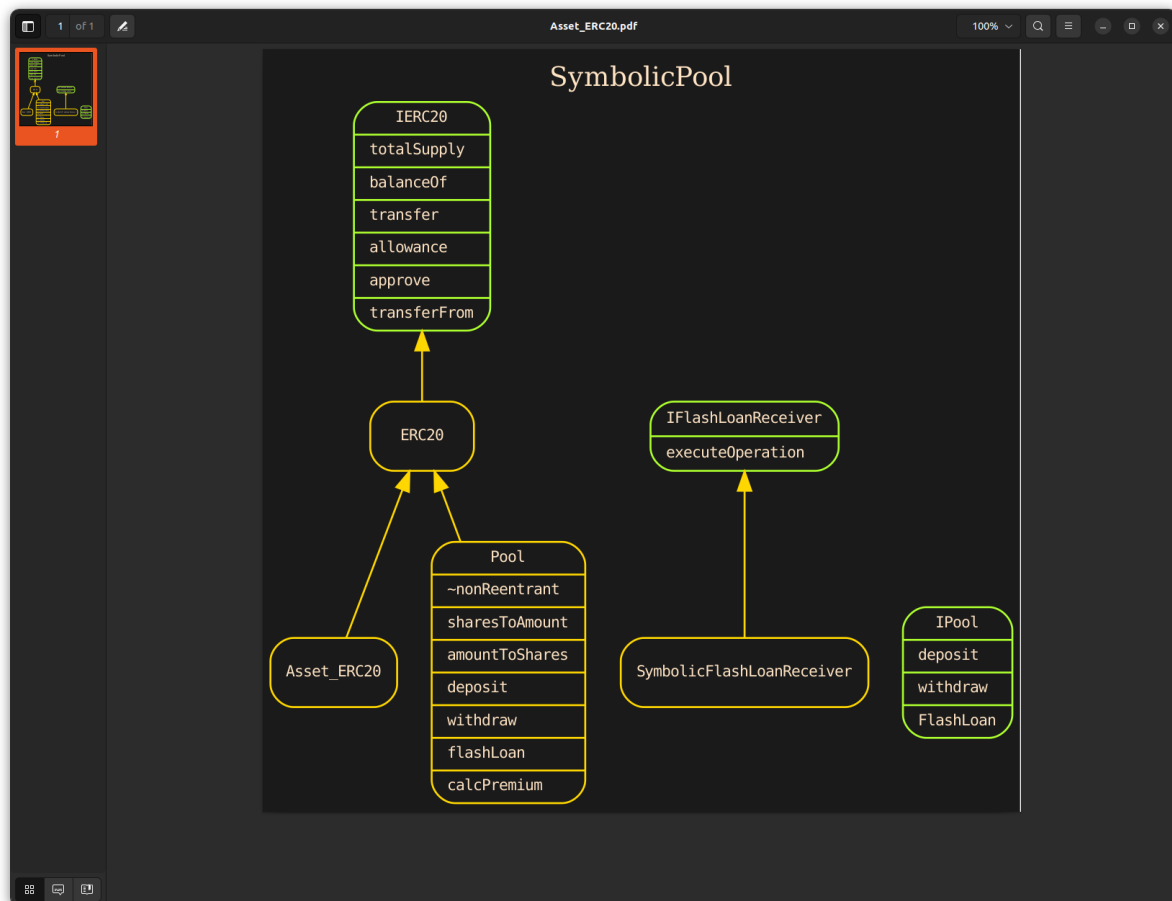
### 8.2 Adding an image

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_
↪subtitles.mp4
```

---

**Todo:** Missing topics to add:

- table of contents (mainly the `hidden` option)
  - adding images and using the `only-light` and `only-dark` classes in furo
  - tabs (from `sphinx-design`)
  - footnotes
  - `.. rubric`, `.. centered` and `.. hlist`
-

## Index

### C

Certora Prover, [15](#)  
closed captions, [25](#)  
CVL, [15](#)

### D

dev-build, [15](#)

### I

image, [24](#)

### P

picture, [24](#)  
Prover, [15](#)

### S

subtitles, [25](#)

### T

table, [24](#)  
todo, [16](#)

### V

video, [25](#)  
    youtube, [5](#)

### Y

youtube, [5](#)