

```
# linguist override file
# Github uses linguist to detect the languages used in a repo for statistical purposes
and to show on the repo's home page.
# OnlyDust links to this info for the project's stats.
# This file should make AsciiDoc visible to linguist, so that ultimately OnlyDust will show
it.
# List of supported languages includes AsciiDoc: https://github.com/github-linguist/linguist/blob/master/lib/linguist/languages.yml
# See: https://github.com/github-linguist/linguist/blob/master/docs/overrides.md#detectable
#
```

```
# Detect .adoc and .yml files
*.adoc linguist-detectable
*.yml linguist-detectable
```

```
---
name: Bug report
about: Create a report to help us improve
title: ""
labels: bug
assignees: stoobie
---
```

****Describe the bug****
A clear and concise description of what the bug is.

****To Reproduce****
Steps to reproduce the behavior:

1. Go to '...'
2. Click on '....'
3. Scroll down to '...'
4. See error

****Expected behavior****
A clear and concise description of what you expected to happen.

****Screenshots****
If applicable, add screenshots to help explain your problem.

****Desktop (please complete the following information):****

- OS: [e.g. iOS]
- Browser [e.g. chrome, safari]
- Version [e.g. 22]

****Additional context****

Add any other context about the problem here.

name: Feature request

about: Suggest an idea for this project

title: ""

labels: enhancement

assignees: stoobie

****Is your feature request related to a problem? Please describe.****

A clear and concise description of what the problem is. Ex. I'm always frustrated when [...]

****Describe the solution you'd like****

A clear and concise description of what you want to happen.

****Describe alternatives you've considered****

A clear and concise description of any alternative solutions or features you've considered.

****Additional context****

Add any other context or screenshots about the feature request here.

Description of the Changes

Please add a detailed description of the change, whether it's an enhancement or a bugfix.

If the PR is related to an open issue please link to it.

PR Preview URL

After you push a commit to this PR, a preview is built and a URL to the root of the preview appears in the comment feed.

Paste here the specific URL(s) of the content that this PR addresses.

Check List

- [] Changes have been done against main branch, and PR does not conflict
- [] PR title follows the convention: `(optional scope): <description>`, e.g: `fix: minor typos in code`
ErrorDocument 404 /404.html

<IfModule mod_headers.c>

Header set Cache-Control "no-cache, no-store, must-revalidate"

```
Header set Pragma "no-cache"  
Header set Expires 0  
</IfModule>
```

```
# Code generated by www.301-redirect.online  
RewriteEngine on  
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^docs/$ /documentation? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^docs/$ /documentation? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^docs/Blocks/header/$ /documentation/architecture_and_concepts/  
Network_Architecture/header? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^documentation/develop/Blocks/header/$ /documentation/  
architecture_and_concepts/Network_Architecture/header? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^docs/Blocks/header/index\.html$ /documentation/  
architecture_and_concepts/Network_Architecture/header? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^/develop/Blocks/header/index\.html$ /documentation/  
architecture_and_concepts/Network_Architecture/header? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^docs/Blocks/transaction\-life\-cycle/$ /documentation/  
architecture_and_concepts/Network_Architecture/transaction-life-cycle? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^/develop/Blocks/transaction\-life\-cycle/$ /documentation/  
architecture_and_concepts/Network_Architecture/transaction-life-cycle? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^docs/Blocks/transaction\-life\-cycle/index\.html$ /documentation/  
architecture_and_concepts/Network_Architecture/transaction-life-cycle? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^/develop/Blocks/transaction\-life\-cycle/index\.html$ /documentation/  
architecture_and_concepts/Network_Architecture/transaction-life-cycle? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$  
RewriteRule ^docs/Blocks/transactions/$ /documentation/architecture_and_concepts/
```

Network_Architecture/transactions? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/Blocks/transactions/\$ /documentation/
architecture_and_concepts/Network_Architecture/transactions? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Blocks/transactions/index\.html\$ /documentation/
architecture_and_concepts/Network_Architecture/transactions? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/Blocks/transactions/index\.html\$ /documentation/
architecture_and_concepts/Network_Architecture/transactions? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/CLI/commands/\$ /documentation/cli/starkli? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/CLI/commands/\$ /documentation/cli/starkli? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/CLI/commands/index\.html\$ /documentation/cli/starkli? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/CLI/commands/index\.html\$ /documentation/cli/starkli? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Contracts/contract\-abi/\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-abi? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/Contracts/contract\-abi/\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-abi? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Contracts/contract\-abi/index\.html\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-abi? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/Contracts/contract\-abi/index\.html\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-abi? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Contracts/contract\-address/\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-address? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/Contracts/contract\-address/\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-address? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Contracts/contract\-address/index\.html\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-address? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/Contracts/contract\-address/index\.html\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-address? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Contracts/contract\-classes/\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-classes? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/Contracts/contract\-classes/\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-classes? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Contracts/contract\-classes/index\.html\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-classes? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/Contracts/contract\-classes/index\.html\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-classes? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Contracts/contract\-hash/\$ /documentation/
architecture_and_concepts/Smart_Contracts/class-hash? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/Contracts/contract\-hash/\$ /documentation/
architecture_and_concepts/Smart_Contracts/class-hash? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Contracts/contract\-hash/index\.html\$ /documentation/
architecture_and_concepts/Smart_Contracts/class-hash? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/Contracts/contract\-hash/index\.html\$ /documentation/
architecture_and_concepts/Smart_Contracts/class-hash? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Contracts/contract\-storage/\$ /documentation/

architecture_and_concepts/Smart_Contracts/contract-storage? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/Contracts/contract\storage/\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-storage? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Contracts/contract\storage/index\html\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-storage? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/Contracts/contract\storage/index\html\$ /documentation/
architecture_and_concepts/Smart_Contracts/contract-storage? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Data\ Availabilty/on\chain\data/\$ /documentation/
architecture_and_concepts/Network_Architecture/on-chain-data? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/Data_Availabilty/on\chain-data/\$ /documentation/
architecture_and_concepts/Network_Architecture/on-chain-data? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Data\ Availabilty/on\chain\data/index\html\$ /documentation/
architecture_and_concepts/Network_Architecture/on-chain-data? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/Data_Availabilty/on\chain\data/index\html\$ /documentation/
architecture_and_concepts/Network_Architecture/on-chain-data? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Events/starknet\events/\$ /documentation/
architecture_and_concepts/Smart_Contracts/starknet-events? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/Events/starknet\events/\$ /documentation/
architecture_and_concepts/Smart_Contracts/starknet-events? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Events/starknet\events/index\html\$ /documentation/
architecture_and_concepts/Smart_Contracts/starknet-events? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^/develop/Events/starknet\events/index\html\$ /documentation/
architecture_and_concepts/Smart_Contracts/starknet-events? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Fees/fee\-mechanism/\$ /documentation/architecture_and_concepts/
Network_Architecture/fee-mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/Fees/fee\-mechanism/\$ /documentation/
architecture_and_concepts/Network_Architecture/fee-mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Fees/fee\-mechanism/index\.html\$ /documentation/
architecture_and_concepts/Network_Architecture/fee-mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/Fees/fee\-mechanism/index\.html\$ /documentation/
architecture_and_concepts/Network_Architecture/fee-mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Hashing/hash\-functions/\$ /documentation/
architecture_and_concepts/Cryptography/hash-functions? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/Hashing/hash\-functions/\$ /documentation/
architecture_and_concepts/Cryptography/hash-functions? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Hashing/hash\-functions/index\.html\$ /documentation/
architecture_and_concepts/Cryptography/hash-functions? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/Hashing/hash\-functions/index\.html\$ /documentation/
architecture_and_concepts/Cryptography/hash-functions? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/intro/\$ /documentation? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/intro/index\.html\$ /documentation? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/L1\-L2_Communication/messaging\-mechanism/\$ /documentation/
architecture_and_concepts/Network_Architecture/messaging-mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/L1\-L2_Communication/messaging\-mechanism/\$ /
documentation/architecture_and_concepts/Network_Architecture/messaging-
mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/L1\L2\ Communication/messaging\-mechanism/index\.html\$ /
documentation/architecture_and_concepts/Network_Architecture/messaging-
mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/L1\L2\ Communication/messaging\-mechanism/index\.html\$\$ /
documentation/architecture_and_concepts/Network_Architecture/messaging-
mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/L1\L2\ Communication/token\-bridge/\$ /documentation/tools/
starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/L1\L2\ Communication/token\-bridge/\$ /documentation/tools/
starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/develop/L1\L2\ Communication/token\-bridge\.html\$ /
documentation/tools/starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/L1\L2\ Communication/token\-bridge/index\.html\$ /documentation/
tools/starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/L1\L2\ Communication/token\-bridge/index\.html\$ /
documentation/tools/starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/L1<\>L2\ Communication/token\-bridge/\$ /documentation/tools/
starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/L1\L2\ Communication/token\-bridge/\$ /documentation/tools/
starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/L1<\>L2\ Communication/token\-bridge/index\.html\$ /
documentation/tools/starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/L1\L2\ Communication/token\-bridge/index\.html\$ /
documentation/tools/starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/State/starknet\~state/\$ /documentation/architecture_and_concepts/
Network_Architecture/starknet-state? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/State/starknet\~state/\$ /documentation/
architecture_and_concepts/Network_Architecture/starknet-state? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/State/starknet\~state/index\.html\$ /documentation/
architecture_and_concepts/Network_Architecture/starknet-state? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^/develop/State/starknet\~state/index\.html\$ /documentation/
architecture_and_concepts/Network_Architecture/starknet-state? [R=301,L]

Added Dec 3, 2023

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Blocks/header/\$ /
documentation/architecture_and_concepts/Network_Architecture/header? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Blocks/header/index\.html\$ /
documentation/architecture_and_concepts/Network_Architecture/header? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Blocks/transaction\~life\~cycle/
\$ /documentation/architecture_and_concepts/Network_Architecture/transaction-life-
cycle? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Blocks/transaction\~life\~cycle/
index\.html\$ /documentation/architecture_and_concepts/Network_Architecture/
transaction-life-cycle? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Blocks/transactions/\$ /
documentation/architecture_and_concepts/Network_Architecture/transactions?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Blocks/transactions/
index\.html\$ /documentation/architecture_and_concepts/Network_Architecture/
transactions? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Fees/fee\-mechanism/\$ /
documentation/architecture_and_concepts/Network_Architecture/fee-mechanism?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Fees/fee\-mechanism/
index\.html\$ /documentation/architecture_and_concepts/Network_Architecture/fee-
mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/L1\-L2_Communication/token\
bridge/\$ /documentation/tools/starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/L1\-L2_Communication/token\
bridge/index\.html\$ /documentation/tools/starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/token\
bridge/\$ /documentation/tools/starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/token\
bridge/index.html\$ /documentation/tools/starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/L1\-L2_Communication/
messaging\-mechanism/index\.html\$ /documentation/architecture_and_concepts/
Network_Architecture/messaging-mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/L1\-L2_Communication/
messaging\-mechanism/\$ /documentation/architecture_and_concepts/
Network_Architecture/messaging-mechanism? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Events/starknet\-events/\$ /
documentation/architecture_and_concepts/Smart_Contracts/starknet-events? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Events/starknet\~events/
index\.html\$ /documentation/architecture_and_concepts/Smart_Contracts/starknet-
events? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Data_Availability/on\~chain\
data/\$ /documentation/architecture_and_concepts/Network_Architecture/on-chain-
data? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Data_Availability/on\~chain\
data/index\.html\$ /documentation/architecture_and_concepts/Network_Architecture/on-
chain-data? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/State/starknet\~state/\$ /
documentation/architecture_and_concepts/Network_Architecture/starknet-state?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/State/starknet\~state/
index\.html\$ /documentation/architecture_and_concepts/Network_Architecture/starknet-
state? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Contracts/class\~hash/\$ /
documentation/architecture_and_concepts/Smart_Contracts/class-hash? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Contracts/class\~hash/
index\.html\$ /documentation/architecture_and_concepts/Smart_Contracts/class-hash?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/develop/L1\~L2_Communication/token\~bridge\.html\$ /
documentation/tools/starkgate-bridge? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Hashing/hash\~functions/\$ /
documentation/architecture_and_concepts/Cryptography/hash-functions? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Hashing/hash\~functions/
index\.html\$ /documentation/architecture_and_concepts/Cryptography/hash-functions?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Contracts/system\~calls\~cairo1\$ /documentation/architecture_and_concepts/Smart_Contracts/system-calls-cairo1? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Contracts/system\~calls\~cairo1/index\~html\$ /documentation/architecture_and_concepts/Smart_Contracts/system-calls-cairo1? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Contracts/system\~calls\~cairo0\$ /documentation/architecture_and_concepts/Smart_Contracts/system-calls-cairo1? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Contracts/system\~calls\~cairo0/index\~html\$ /documentation/architecture_and_concepts/Smart_Contracts/system-calls-cairo1? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Cairo_on_Starknet/cairo\~1\~and\~sierra/\$ /documentation/architecture_and_concepts/Smart_Contracts/cairo-and-sierra? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Cairo_on_Starknet/cairo\~1\~and\~sierra/index\~html\$ /documentation/architecture_and_concepts/Smart_Contracts/cairo-and-sierra? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Cairo_on_Starknet/contract\~syntax/\$ /documentation/architecture_and_concepts/Smart_Contracts/contract-syntax? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Cairo_on_Starknet/contract\~syntax/index\~html\$ /documentation/architecture_and_concepts/Smart_Contracts/contract-syntax? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/validate_and_execute/index\~html\$ /documentation/architecture_and_concepts/Accounts/account_functions? [R=301,L]

```
RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/architecture_and_concepts/Accounts/
validate_and_execute/$ /documentation/architecture_and_concepts/Accounts/
account_functions? [R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/getting_started/environment_setup/$ /documentation/
quick_start/environment_setup? [R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/getting_started/environment_setup/index\.html$ /
documentation/quick_start/environment_setup? [R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/getting_started/account_setup/$ /documentation/
quick_start/set_up_an_account? [R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/getting_started/account_setup/index\.html$ /
documentation/quick_start/set_up_an_account? [R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/tools/CLI/starkli/$ /documentation/tools/devtools/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/tools/CLI/starkli/index\.html$ /documentation/tools/
devtools/? [R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/tools/CLI/commands/$ /documentation/cli/starkli?
[R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/tools/CLI/commands/index\.html$ /documentation/cli/
starkli? [R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/cli/commands/$ /documentation/cli/starkli? [R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/cli/commands/index\.html$ /documentation/cli/starkli?
[R=301,L]

RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/tools/api_rpc/$ /documentation/tools/api-services?
```

[R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/api_rpc/index\.html\$ /documentation/tools/api-services? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/api/api-reference/\$ /documentation/tools/api-services? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/api/api-reference/index\.html\$ /documentation/tools/api-services? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/contribute/bug-bounty/index\.html\$ /documentation? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/contribute/bug-bounty/\$ /documentation? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/contribute/completed-audits/\$ /documentation? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/contribute/completed-audits/index\.html\$ /documentation? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/contribute/conduct/\$ /documentation? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/contribute/conduct/index\.html\$ /documentation? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Blocks/transactions/\$ /documentation/architecture_and_concepts/Network_Architecture/transactions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Blocks/transactions\$ /documentation/architecture_and_concepts/Network_Architecture/transactions/? [R=301,L]

June 3, 2024 SEO directory restructure

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/\$ /? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/index\.html\$ /index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/account_functions/\$ /architecture-and-concepts/accounts/account-functions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/account_functions/index\.html\$ /architecture-and-concepts/accounts/account-functions/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/approach/\$ /architecture-and-concepts/accounts/approach/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/approach/index\.html\$ /architecture-and-concepts/accounts/approach/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/deploying_new_accounts/\$ /architecture-and-concepts/accounts/deploying-new-accounts/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/deploying_new_accounts/index\.html\$ /architecture-and-concepts/accounts/deploying-new-accounts/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/introduction/\$ /architecture-and-concepts/accounts/introduction/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/introduction/index\.html\$ /architecture-and-concepts/accounts/introduction/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/

simplified_transaction_flow/\$ /architecture-and-concepts/accounts/simplified-transaction-flow/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/simplified_transaction_flow/index\..html\$ /architecture-and-concepts/accounts/simplified-transaction-flow/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/universal-deployer/\$ /architecture-and-concepts/accounts/universal-deployer/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Accounts/universal-deployer/index\..html\$ /architecture-and-concepts/accounts/universal-deployer/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Cryptography/hash-functions/\$ /architecture-and-concepts/cryptography/hash-functions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Cryptography/hash-functions/index\..html\$ /architecture-and-concepts/cryptography/hash-functions/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Cryptography/p-value/\$ /architecture-and-concepts/cryptography/p-value/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Cryptography/p-value/index\..html\$ /architecture-and-concepts/cryptography/p-value/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Cryptography/stark-curve/\$ /architecture-and-concepts/cryptography/stark-curve/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Cryptography/stark-curve/index\..html\$ /architecture-and-concepts/cryptography/stark-curve/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Economics-of-Starknet/\$ /architecture-and-concepts/economics-of-starknet/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Economics\-\of\-\Starknet/
index\.html\$ /architecture-and-concepts/economics-of-starknet/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/fee\
mechanism/\$ /architecture-and-concepts/network-architecture/fee-mechanism/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/fee\
mechanism/index\.html\$ /architecture-and-concepts/network-architecture/fee-
mechanism/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/header/
\$ /architecture-and-concepts/network-architecture/block-structure/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/header/
index\.html\$ /architecture-and-concepts/network-architecture/block-structure/
index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/
messaging\-\mechanism/\$ /architecture-and-concepts/network-architecture/messaging-
mechanism/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/
messaging\-\mechanism/index\.html\$ /architecture-and-concepts/network-architecture/
messaging-mechanism/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/on\
chain\-\data/\$ /architecture-and-concepts/network-architecture/data-availability/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/on\
chain\-\data/index\.html\$ /architecture-and-concepts/network-architecture/data-
availability/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/
starknet_architecture_overview/\$ /architecture-and-concepts/network-architecture/

starknet-architecture-overview/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/starknet_architecture_overview/index\.html\$ /architecture-and-concepts/network-architecture/starknet-architecture-overview/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/starknet\-state/\$ /architecture-and-concepts/network-architecture/starknet-state/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/starknet\-state/index\.html\$ /architecture-and-concepts/network-architecture/starknet-state/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/transaction\-life\-cycle/\$ /architecture-and-concepts/network-architecture/transaction-life-cycle/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/transaction\-life\-cycle/index\.html\$ /architecture-and-concepts/network-architecture/transaction-life-cycle/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/transactions/\$ /architecture-and-concepts/network-architecture/transactions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/transactions/index\.html\$ /architecture-and-concepts/network-architecture/transactions/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/cairo\-and\-sierra/\$ /architecture-and-concepts/smart-contracts/cairo-and-sierra/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/cairo\-and\-sierra/index\.html\$ /architecture-and-concepts/smart-contracts/cairo-and-sierra/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/cairo\-\nbuiltins/\$ /architecture-and-concepts/smart-contracts/cairo-builtins/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/cairo\-\nbuiltins/index\..html\$ /architecture-and-concepts/smart-contracts/cairo-builtins/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/class\-\hash/\$ /architecture-and-concepts/smart-contracts/class-hash/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/class\-\hash/index\..html\$ /architecture-and-concepts/smart-contracts/class-hash/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/contract\-\abi/\$ /architecture-and-concepts/smart-contracts/contract-abi/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/contract\-\abi/index\..html\$ /architecture-and-concepts/smart-contracts/contract-abi/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/contract\-\address/\$ /architecture-and-concepts/smart-contracts/contract-address/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/contract\-\address/index\..html\$ /architecture-and-concepts/smart-contracts/contract-address/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/contract\-\classes/\$ /architecture-and-concepts/smart-contracts/contract-classes/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/contract\-\classes/index\..html\$ /architecture-and-concepts/smart-contracts/contract-classes/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/contract\-

storage/\$ /architecture-and-concepts/smart-contracts/contract-storage/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/contract\-\nstorage/index\.html\$ /architecture-and-concepts/smart-contracts/contract-storage/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/contract\-\nsyntax/\$ /architecture-and-concepts/smart-contracts/contract-syntax/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/contract\-\nsyntax/index\.html\$ /architecture-and-concepts/smart-contracts/contract-syntax/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/execution_info/\$ /architecture-and-concepts/smart-contracts/execution-info/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/execution_info/index\.html\$ /architecture-and-concepts/smart-contracts/execution-info/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/serialization_of_Cairo_types/\$ /architecture-and-concepts/smart-contracts/serialization-of-cairo-types/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/serialization_of_Cairo_types/index\.html\$ /architecture-and-concepts/smart-contracts/serialization-of-cairo-types/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/starknet\-\nevents/\$ /architecture-and-concepts/smart-contracts/starknet-events/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/starknet\-\nevents/index\.html\$ /architecture-and-concepts/smart-contracts/starknet-events/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/system\-

calls\~cairo1/\$ /architecture-and-concepts/smart-contracts/system-calls-cairo1/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Smart_Contracts/system\~calls\~cairo1/index\.html\$ /architecture-and-concepts/smart-contracts/system-calls-cairo1/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/cli/starkli/\$ /cli/starkli/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/cli/starkli/index\.html\$ /cli/starkli/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/cli/starknet\~compiler\~options/\$ /cli/starknet-compiler-options/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/cli/starknet\~compiler\~options/index\.html\$ /cli/starknet-compiler-options/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/glossary/\$ /glossary/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/glossary/index\.html\$ /glossary/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/notational\~conventions/\$ /notational-conventions/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/notational\~conventions/index\.html\$ /notational-conventions/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/quick_start/declare_a_smart_contract/\$ /quick-start/declare-a-smart-contract/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/quick_start/declare_a_smart_contract/index\.html\$ /quick-start/declare-a-smart-contract/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/quick_start/deploy_a_smart_contract/\$ /quick-start/deploy-

a-smart-contract/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/quick_start/deploy_a_smart_contract/index\.html\$ /quick-start/deploy-a-smart-contract/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/quick_start/environment_setup/\$ /quick-start/environment-setup/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/quick_start/environment_setup/index\.html\$ /quick-start/environment-setup/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/quick_start/interact_with_a_smart_contract/\$ /quick-start/interact-with-a-smart-contract/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/quick_start/interact_with_a_smart_contract/index\.html\$ /quick-start/interact-with-a-smart-contract/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/quick_start/set_up_an_account/\$ /quick-start/set-up-an-account/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/quick_start/set_up_an_account/index\.html\$ /quick-start/set-up-an-account/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/starknet_versions/deprecated/\$ /starknet-versions/deprecated/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/starknet_versions/deprecated/index\.html\$ /starknet-versions/deprecated/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/starknet_versions/juno_versions/\$ /starknet-versions/juno-versions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/starknet_versions/juno_versions/index\.html\$ /starknet-versions/juno-versions/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/starknet_versions/pathfinder_versions/\$ /starknet-versions/pathfinder-versions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/starknet_versions/pathfinder_versions/index\.html\$ /starknet-versions/pathfinder-versions/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/starknet_versions/upcoming_versions/\$ /starknet-versions/upcoming-versions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/starknet_versions/upcoming_versions/index\.html\$ /starknet-versions/upcoming-versions/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/starknet_versions/version_notes/\$ /starknet-versions/version-notes/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/starknet_versions/version_notes/index\.html\$ /starknet-versions/version-notes/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/api-services/\$ /tools/api-services/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/api-services/index\.html\$ /tools/api-services/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/audit/\$ /tools/audit/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/audit/index\.html\$ /tools/audit/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/bridged_tokens/\$ /tools/bridged-tokens/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/bridged_tokens/index\.html\$ /tools/bridged-tokens/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/dai_token_migration/\$ /tools/dai-token-migration/?

[R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/dai_token_migration/index\.html\$ /tools/dai-token-migration/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/devtools/\$ /tools/devtools/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/devtools/index\.html\$ /tools/devtools/index.html?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/important_addresses/\$ /tools/important-addresses/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/important_addresses/index\.html\$ /tools/important-addresses/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/limits_and_triggers/\$ /tools/limits-and-triggers/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/limits_and_triggers/index\.html\$ /tools/limits-and-triggers/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/ref_block_explorers/\$ /tools/ref-block-explorers/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/ref_block_explorers/index\.html\$ /tools/ref-block-explorers/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/starkgate\-adding_a_token/\$ /starkgate/adding-a-token/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/starkgate\-adding_a_token/index\.html\$ /starkgate/adding-a-token/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/tools/starkgate_architecture/\$ /starkgate/architecture/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate_architecture/index\.html\$ /starkgate/
architecture/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-automated_actions_with_bridging/\$ /
starkgate/automated-actions-with-bridging/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-automated_actions_with_bridging/
index\.html\$ /starkgate/automated-actions-with-bridging/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-bridge/\$ /starkgate/overview/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-bridge/index\.html\$ /starkgate/overview/
index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-cancelling\ a\ deposit/\$ /starkgate/
cancelling-a-deposit/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-cancelling\ a\ deposit/index\.html\$ /
starkgate/cancelling-a-deposit/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-depositing/\$ /starkgate/depositing/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-depositing/index\.html\$ /starkgate/
depositing/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-estimating_fees/\$ /starkgate/estimating-
fees/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-estimating_fees/index\.html\$ /starkgate/
estimating-fees/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate_function_reference/\$ /starkgate/function-reference/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate_function_reference/index\.html\$ /starkgate/function-reference/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-withdrawing/\$ /starkgate/withdrawing/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starkgate\-withdrawing/index\.html\$ /starkgate/withdrawing/index.html? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starknet\-book/\$ /tools/starknet-book/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/tools/starknet\-book/index\.html\$ /tools/starknet-book/index.html? [R=301,L]

Fix broken backlinks June 3, 2024

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/intro\$ /? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Account_Abstraction/introduction/\$ /architecture-and-concepts/accounts/introduction/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Contracts/contract\-classes/\$ /architecture-and-concepts/smart-contracts/contract-classes/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/useful_info/\$ /tools/important-addresses/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Contracts/system\-calls\-cairo1/\$ /architecture-and-concepts/smart-contracts/system-calls-cairo1/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/L1\L2\ Communication/messaging\-mechanism\$ /architecture-and-concepts/network-architecture/messaging-mechanism/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Contracts/contract\-storage/\$ /architecture-and-concepts/smart-contracts/contract-storage/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Blocks/transactions\$ /architecture-and-concepts/network-architecture/transactions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/develop/Blocks/transactions/\$ /architecture-and-concepts/network-architecture/transactions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/getting_started/\$ /quick-start/environment-setup/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/CLI/commands\$ /cli/starkli/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/Fees/fee\-mechanism\$ /architecture-and-concepts/network-architecture/fee-mechanism/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^docs/L1\<\>L2\ Communication/token\-bridge\$ /starkgate/overview/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture\$ /architecture-and-concepts/network-architecture/starknet-architecture-overview/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Contracts/contract\-address/\$ /architecture-and-concepts/smart-contracts/contract-address/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/develop/State/starknet\-state/\$ /architecture-and-concepts/network-architecture/starknet-state/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$
RewriteRule ^documentation/architecture_and_concepts/Account_Abstraction/

approach/\$ /architecture-and-concepts/accounts/approach/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Contracts/contract\-abi/\$ /
architecture-and-concepts/smart-contracts/contract-abi/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/Blocks/
transactions/\$ /architecture-and-concepts/network-architecture/transactions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/transaction\-structure\-and\-hash/\$ /architecture-and-
concepts/network-architecture/transactions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/L1\-
L2_Communication/token\-bridge/\$ /starkgate/overview/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/develop/Hashing/hash\-functions\$ /architecture-and-
concepts/cryptography/hash-functions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/develop/Hashing/hash\-functions/\$ /architecture-and-
concepts/cryptography/hash-functions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Contracts/contract\-address\$ /architecture-and-concepts/smart-
contracts/contract-address/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^docs/Hashing/hash\-functions\$ /architecture-and-concepts/cryptography/
hash-functions/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Account_Abstraction/
validate_and_execute/\$ /architecture-and-concepts/accounts/account-functions/?
[R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Contracts/cairo\-1\-and\-sierra/
\$ /architecture-and-concepts/smart-contracts/cairo-and-sierra/? [R=301,L]

RewriteCond %{QUERY_STRING} ^\$

RewriteRule ^documentation/architecture_and_concepts/Contracts/system\-calls/\$ /
architecture-and-concepts/smart-contracts/system-calls-cairo1/? [R=301,L]

```
RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/Blocks/
header/$ /architecture-and-concepts/network-architecture/block-structure/? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/
Data_Availability/on\chain\data/$ /architecture-and-concepts/network-architecture/
data-availability/? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/Fees/
fee-mechanism/$ /architecture-and-concepts/network-architecture/fee-mechanism/?
[R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/architecture_and_concepts/Network_Architecture/L1\
L2_Communication/messaging-mechanism/$ /architecture-and-concepts/network-
architecture/messaging-mechanism/? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/getting_started/setting_up_the_environment/$ /quick-start/
environment-setup/? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/getting_started/unit_tests/$ /quick-start/environment-
setup/? [R=301,L]
```

```
RewriteCond %{QUERY_STRING} ^$
RewriteRule ^documentation/tools/CLI/starknet-compiler-options/$ /cli/starknet-
compiler-options/? [R=301,L]
# Changelog
```

All notable changes to this project will be documented in this file. See [standard-version] (<https://github.com/conventional-changelog/standard-version>) for commit guidelines.

[0.1.514](<https://github.com/starknet-io/starknet-docs/compare/v0.1.513...v0.1.514>)
(2024-03-14)

[0.1.513](<https://github.com/starknet-io/starknet-docs/compare/v0.1.512...v0.1.513>)
(2024-03-14)

[0.1.512](<https://github.com/starknet-io/starknet-docs/compare/v0.1.511...v0.1.512>)
(2024-03-14)

[0.1.511](<https://github.com/starknet-io/starknet-docs/compare/v0.1.510...v0.1.511>)

(2024-03-13)

[0.1.510](https://github.com/starknet-io/starknet-docs/compare/v0.1.509...v0.1.510)
(2024-03-13)

[0.1.509](https://github.com/starknet-io/starknet-docs/compare/v0.1.508...v0.1.509)
(2024-03-13)

[0.1.508](https://github.com/starknet-io/starknet-docs/compare/v0.1.507...v0.1.508)
(2024-03-12)

[0.1.507](https://github.com/starknet-io/starknet-docs/compare/v0.1.506...v0.1.507)
(2024-03-12)

Bug Fixes

- add felt size multiplier to data gas usage ([#1174](https://github.com/starknet-io/starknet-docs/issues/1174)) ([6e2a41e](https://github.com/starknet-io/starknet-docs/commit/6e2a41e60625c9d2f96e5b5b7ad739698294a18d))

[0.1.506](https://github.com/starknet-io/starknet-docs/compare/v0.1.505...v0.1.506)
(2024-03-11)

[0.1.505](https://github.com/starknet-io/starknet-docs/compare/v0.1.504...v0.1.505)
(2024-03-10)

Bug Fixes

- Fix example in emit_events ([#1169](https://github.com/starknet-io/starknet-docs/issues/1169)) ([75b08a1](https://github.com/starknet-io/starknet-docs/commit/75b08a17fff8e8b0ca07d303b8d29b55390d6588))

[0.1.504](https://github.com/starknet-io/starknet-docs/compare/v0.1.503...v0.1.504)
(2024-03-10)

[0.1.503](https://github.com/starknet-io/starknet-docs/compare/v0.1.502...v0.1.503)
(2024-03-07)

[0.1.502](https://github.com/starknet-io/starknet-docs/compare/v0.1.501...v0.1.502)
(2024-03-07)

[0.1.501](https://github.com/starknet-io/starknet-docs/compare/v0.1.500...v0.1.501)
(2024-03-06)

[0.1.500](https://github.com/starknet-io/starknet-docs/compare/v0.1.499...v0.1.500)
(2024-03-06)

[0.1.499](https://github.com/starknet-io/starknet-docs/compare/v0.1.498...v0.1.499)
(2024-03-05)

[0.1.498](https://github.com/starknet-io/starknet-docs/compare/v0.1.497...v0.1.498)
(2024-03-05)

[0.1.497](https://github.com/starknet-io/starknet-docs/compare/v0.1.496...v0.1.497)
(2024-02-29)

[0.1.496](https://github.com/starknet-io/starknet-docs/compare/v0.1.495...v0.1.496)
(2024-02-29)

Features

- Added callouts to example in Data availability topic. ([#1006](https://github.com/starknet-io/starknet-docs/issues/1006)) ([e1b8500](https://github.com/starknet-io/starknet-docs/commit/e1b850074a9d2fd8113bef7de48b0fa1cca8ac))

[0.1.495](https://github.com/starknet-io/starknet-docs/compare/v0.1.494...v0.1.495)
(2024-02-29)

Bug Fixes

- Testing highlightjs-cairo v4.0 in starknet-docs-antora-ui ([#1153](https://github.com/starknet-io/starknet-docs/issues/1153)) ([1668841](https://github.com/starknet-io/starknet-docs/commit/1668841717103e1b49d6221024c8f736419f6034))

[0.1.494](https://github.com/starknet-io/starknet-docs/compare/v0.1.493...v0.1.494)
(2024-02-28)

[0.1.493](https://github.com/starknet-io/starknet-docs/compare/v0.1.492...v0.1.493)
(2024-02-26)

Bug Fixes

- Update validation info in transaction_lifecycle.adoc ([#1151](https://github.com/starknet-io/starknet-docs/issues/1151)) ([2fc6bbf](https://github.com/starknet-io/starknet-docs/commit/2fc6bbfc6b8f90a059657aa3aa63c9067ec958c7))

[0.1.492](https://github.com/starknet-io/starknet-docs/compare/v0.1.491...v0.1.492)
(2024-02-26)

[0.1.491](https://github.com/starknet-io/starknet-docs/compare/v0.1.490...v0.1.491)
(2024-02-26)

Bug Fixes

- generate previews from forked repos ([#1097](https://github.com/starknet-io/starknet-docs/issues/1097)) ([f37a901](https://github.com/starknet-io/starknet-docs/commit/f37a9016c571c038740b50b8f2b5b27511450b62))

[0.1.490](https://github.com/starknet-io/starknet-docs/compare/v0.1.489...v0.1.490) (2024-02-25)

[0.1.489](https://github.com/starknet-io/starknet-docs/compare/v0.1.488...v0.1.489) (2024-02-25)

[0.1.488](https://github.com/starknet-io/starknet-docs/compare/v0.1.487...v0.1.488) (2024-02-25)

[0.1.487](https://github.com/starknet-io/starknet-docs/compare/v0.1.486...v0.1.487) (2024-02-25)

[0.1.486](https://github.com/starknet-io/starknet-docs/compare/v0.1.485...v0.1.486) (2024-02-22)

[0.1.485](https://github.com/starknet-io/starknet-docs/compare/v0.1.484...v0.1.485) (2024-02-22)

[0.1.484](https://github.com/starknet-io/starknet-docs/compare/v0.1.483...v0.1.484) (2024-02-21)

[0.1.483](https://github.com/starknet-io/starknet-docs/compare/v0.1.482...v0.1.483) (2024-02-21)

[0.1.482](https://github.com/starknet-io/starknet-docs/compare/v0.1.481...v0.1.482) (2024-02-21)

[0.1.481](https://github.com/starknet-io/starknet-docs/compare/v0.1.480...v0.1.481) (2024-02-21)

[0.1.480](https://github.com/starknet-io/starknet-docs/compare/v0.1.479...v0.1.480) (2024-02-19)

[0.1.479](https://github.com/starknet-io/starknet-docs/compare/v0.1.478...v0.1.479) (2024-02-15)

[0.1.478](https://github.com/starknet-io/starknet-docs/compare/v0.1.477...v0.1.478) (2024-02-15)

[0.1.477](https://github.com/starknet-io/starknet-docs/compare/v0.1.476...v0.1.477)

(2024-02-14)

[0.1.476](https://github.com/starknet-io/starknet-docs/compare/v0.1.475...v0.1.476)
(2024-02-14)

Bug Fixes

- Minor edits to Economics-of-Starknet.adoc ([#1130](https://github.com/starknet-io/starknet-docs/issues/1130)) ([d00cfa6](https://github.com/starknet-io/starknet-docs/commit/d00cfa6f9aa3f9ebb42bd77e6c1bfc3c0fb830ae))

[0.1.475](https://github.com/starknet-io/starknet-docs/compare/v0.1.474...v0.1.475)
(2024-02-14)

[0.1.474](https://github.com/starknet-io/starknet-docs/compare/v0.1.473...v0.1.474)
(2024-02-13)

[0.1.473](https://github.com/starknet-io/starknet-docs/compare/v0.1.472...v0.1.473)
(2024-02-11)

Bug Fixes

- ignore fee token contract is the fee formula ([#1123](https://github.com/starknet-io/starknet-docs/issues/1123)) ([3863f6f](https://github.com/starknet-io/starknet-docs/commit/3863f6f693e3f3d06ee53f67cff4e7afb4334e13))

[0.1.472](https://github.com/starknet-io/starknet-docs/compare/v0.1.471...v0.1.472)
(2024-02-11)

[0.1.471](https://github.com/starknet-io/starknet-docs/compare/v0.1.470...v0.1.471)
(2024-02-08)

[0.1.470](https://github.com/starknet-io/starknet-docs/compare/v0.1.469...v0.1.470)
(2024-02-08)

[0.1.469](https://github.com/starknet-io/starknet-docs/compare/v0.1.468...v0.1.469)
(2024-02-06)

[0.1.468](https://github.com/starknet-io/starknet-docs/compare/v0.1.467...v0.1.468)
(2024-02-05)

[0.1.467](https://github.com/starknet-io/starknet-docs/compare/v0.1.466...v0.1.467)
(2024-02-05)

[0.1.466](https://github.com/starknet-io/starknet-docs/compare/v0.1.465...v0.1.466)
(2024-02-04)

[0.1.465](https://github.com/starknet-io/starknet-docs/compare/v0.1.464...v0.1.465)
(2024-02-04)

Bug Fixes

- fee calculation formula ([#1092](https://github.com/starknet-io/starknet-docs/issues/1092)) ([0c36768](https://github.com/starknet-io/starknet-docs/commit/0c367688427c9bff7c4e74e60b7b4da2bc8009da))

[0.1.464](https://github.com/starknet-io/starknet-docs/compare/v0.1.463...v0.1.464)
(2024-01-30)

[0.1.463](https://github.com/starknet-io/starknet-docs/compare/v0.1.462...v0.1.463)
(2024-01-28)

[0.1.462](https://github.com/starknet-io/starknet-docs/compare/v0.1.461...v0.1.462)
(2024-01-21)

[0.1.461](https://github.com/starknet-io/starknet-docs/compare/v0.1.460...v0.1.461)
(2024-01-18)

[0.1.460](https://github.com/starknet-io/starknet-docs/compare/v0.1.459...v0.1.460)
(2024-01-17)

[0.1.459](https://github.com/starknet-io/starknet-docs/compare/v0.1.458...v0.1.459)
(2024-01-17)

[0.1.458](https://github.com/starknet-io/starknet-docs/compare/v0.1.457...v0.1.458)
(2024-01-16)

[0.1.457](https://github.com/starknet-io/starknet-docs/compare/v0.1.456...v0.1.457)
(2024-01-16)

Bug Fixes

- Block limit (Cairo steps) in Update limits_and_triggers.adoc ([#1082](https://github.com/starknet-io/starknet-docs/issues/1082)) ([59b1c82](https://github.com/starknet-io/starknet-docs/commit/59b1c82a322d618a6aff4859988457f25ccb6e09))

[0.1.456](https://github.com/starknet-io/starknet-docs/compare/v0.1.455...v0.1.456)
(2024-01-15)

[0.1.455](https://github.com/starknet-io/starknet-docs/compare/v0.1.454...v0.1.455)
(2024-01-15)

[0.1.454](https://github.com/starknet-io/starknet-docs/compare/v0.1.453...v0.1.454)
(2024-01-15)

[0.1.453](https://github.com/starknet-io/starknet-docs/compare/v0.1.452...v0.1.453)
(2024-01-15)

[0.1.452](https://github.com/starknet-io/starknet-docs/compare/v0.1.451...v0.1.452)
(2024-01-15)

[0.1.451](https://github.com/starknet-io/starknet-docs/compare/v0.1.450...v0.1.451)
(2024-01-14)

[0.1.450](https://github.com/starknet-io/starknet-docs/compare/v0.1.449...v0.1.450)
(2024-01-14)

Bug Fixes

- discrepancies in 0.13.0 fee docs ([#1078](https://github.com/starknet-io/starknet-docs/issues/1078)) ([0f0cf14](https://github.com/starknet-io/starknet-docs/commit/0f0cf14033748cdd90369a00e2a32bef6d882bd1))

[0.1.449](https://github.com/starknet-io/starknet-docs/compare/v0.1.448...v0.1.449)
(2024-01-10)

[0.1.448](https://github.com/starknet-io/starknet-docs/compare/v0.1.447...v0.1.448)
(2024-01-08)

Bug Fixes

- Update tab labels of examples from Cairo v1 to Cairo v2 ([#1072](https://github.com/starknet-io/starknet-docs/issues/1072)) ([105f798](https://github.com/starknet-io/starknet-docs/commit/105f798c865cfbe225af63af51096c499f8cedbc))

[0.1.447](https://github.com/starknet-io/starknet-docs/compare/v0.1.446...v0.1.447)
(2024-01-07)

[0.1.446](https://github.com/starknet-io/starknet-docs/compare/v0.1.445...v0.1.446)
(2024-01-04)

Bug Fixes

- Update Goerli end-of-support until at least end of Q1 2024 ([#1068](https://github.com/starknet-io/starknet-docs/issues/1068)) ([fb67be3](https://github.com/starknet-io/starknet-docs/commit/fb67be3b32b7301316d7a056c710fa81477b2750))

[0.1.445](https://github.com/starknet-io/starknet-docs/compare/v0.1.444...v0.1.445)

(2024-01-04)

Bug Fixes

- Date and percentages in RN ([#1066](https://github.com/starknet-io/starknet-docs/issues/1066)) ([4562306](https://github.com/starknet-io/starknet-docs/commit/456230694da343102bbd889ec7c5db88fac25dfd))

[0.1.444](https://github.com/starknet-io/starknet-docs/compare/v0.1.443...v0.1.444)
(2024-01-04)

[0.1.443](https://github.com/starknet-io/starknet-docs/compare/v0.1.442...v0.1.443)
(2024-01-02)

Bug Fixes

- Update title and add :description: to StarkGate function and event reference ([#1063](https://github.com/starknet-io/starknet-docs/issues/1063)) ([9a0b909](https://github.com/starknet-io/starknet-docs/commit/9a0b909f25496eadfb160b6745b95d9b164f067b))

[0.1.442](https://github.com/starknet-io/starknet-docs/compare/v0.1.441...v0.1.442)
(2024-01-02)

[0.1.441](https://github.com/starknet-io/starknet-docs/compare/v0.1.440...v0.1.441)
(2024-01-01)

[0.1.440](https://github.com/starknet-io/starknet-docs/compare/v0.1.439...v0.1.440)
(2023-12-31)

[0.1.439](https://github.com/starknet-io/starknet-docs/compare/v0.1.438...v0.1.439)
(2023-12-31)

[0.1.438](https://github.com/starknet-io/starknet-docs/compare/v0.1.437...v0.1.438)
(2023-12-31)

[0.1.437](https://github.com/starknet-io/starknet-docs/compare/v0.1.436...v0.1.437)
(2023-12-28)

[0.1.436](https://github.com/starknet-io/starknet-docs/compare/v0.1.435...v0.1.436)
(2023-12-28)

[0.1.435](https://github.com/starknet-io/starknet-docs/compare/v0.1.434...v0.1.435)
(2023-12-28)

[0.1.434](https://github.com/starknet-io/starknet-docs/compare/v0.1.433...v0.1.434)

(2023-12-28)

[0.1.433](https://github.com/starknet-io/starknet-docs/compare/v0.1.432...v0.1.433)
(2023-12-28)

[0.1.432](https://github.com/starknet-io/starknet-docs/compare/v0.1.431...v0.1.432)
(2023-12-27)

[0.1.431](https://github.com/starknet-io/starknet-docs/compare/v0.1.430...v0.1.431)
(2023-12-26)

[0.1.430](https://github.com/starknet-io/starknet-docs/compare/v0.1.429...v0.1.430)
(2023-12-26)

[0.1.429](https://github.com/starknet-io/starknet-docs/compare/v0.1.428...v0.1.429)
(2023-12-26)

[0.1.428](https://github.com/starknet-io/starknet-docs/compare/v0.1.427...v0.1.428)
(2023-12-26)

Bug Fixes

- Fixed bugs in formulas on Gas and transaction fees. Update fees and formulas
([#1027](https://github.com/starknet-io/starknet-docs/issues/1027)) ([69c272b](https://github.com/starknet-io/starknet-docs/commit/69c272b6f3badbef064d15b573db45402e0720ba))

[0.1.427](https://github.com/starknet-io/starknet-docs/compare/v0.1.426...v0.1.427)
(2023-12-25)

[0.1.426](https://github.com/starknet-io/starknet-docs/compare/v0.1.425...v0.1.426)
(2023-12-25)

[0.1.425](https://github.com/starknet-io/starknet-docs/compare/v0.1.424...v0.1.425)
(2023-12-25)

[0.1.424](https://github.com/starknet-io/starknet-docs/compare/v0.1.423...v0.1.424)
(2023-12-25)

[0.1.423](https://github.com/starknet-io/starknet-docs/compare/v0.1.422...v0.1.423)
(2023-12-25)

[0.1.422](https://github.com/starknet-io/starknet-docs/compare/v0.1.421...v0.1.422)
(2023-12-21)

[0.1.421](https://github.com/starknet-io/starknet-docs/compare/v0.1.420...v0.1.421)

(2023-12-21)

[0.1.420](https://github.com/starknet-io/starknet-docs/compare/v0.1.419...v0.1.420)
(2023-12-21)

[0.1.419](https://github.com/starknet-io/starknet-docs/compare/v0.1.418...v0.1.419)
(2023-12-21)

[0.1.418](https://github.com/starknet-io/starknet-docs/compare/v0.1.417...v0.1.418)
(2023-12-21)

[0.1.417](https://github.com/starknet-io/starknet-docs/compare/v0.1.416...v0.1.417)
(2023-12-19)

[0.1.416](https://github.com/starknet-io/starknet-docs/compare/v0.1.415...v0.1.416)
(2023-12-18)

[0.1.415](https://github.com/starknet-io/starknet-docs/compare/v0.1.414...v0.1.415)
(2023-12-18)

[0.1.414](https://github.com/starknet-io/starknet-docs/compare/v0.1.413...v0.1.414)
(2023-12-14)

[0.1.413](https://github.com/starknet-io/starknet-docs/compare/v0.1.412...v0.1.413)
(2023-12-14)

[0.1.412](https://github.com/starknet-io/starknet-docs/compare/v0.1.411...v0.1.412)
(2023-12-14)

[0.1.411](https://github.com/starknet-io/starknet-docs/compare/v0.1.410...v0.1.411)
(2023-12-14)

[0.1.410](https://github.com/starknet-io/starknet-docs/compare/v0.1.409...v0.1.410)
(2023-12-14)

[0.1.409](https://github.com/starknet-io/starknet-docs/compare/v0.1.408...v0.1.409)
(2023-12-14)

Features

- 0.13.0 now on Goerli and Sepolia ([#1024](https://github.com/starknet-io/starknet-docs/issues/1024)) ([5538549](https://github.com/starknet-io/starknet-docs/commit/553854989cd16886011d72d9703539bc61930d11)), closes [PR#1023](https://github.com/starknet-io/PR/issues/1023)

[0.1.408](https://github.com/starknet-io/starknet-docs/compare/v0.1.407...v0.1.408)

(2023-12-11)

[0.1.407](https://github.com/starknet-io/starknet-docs/compare/v0.1.406...v0.1.407)
(2023-12-11)

[0.1.406](https://github.com/starknet-io/starknet-docs/compare/v0.1.405...v0.1.406)
(2023-12-10)

[0.1.405](https://github.com/starknet-io/starknet-docs/compare/v0.1.404...v0.1.405)
(2023-12-10)

[0.1.404](https://github.com/starknet-io/starknet-docs/compare/v0.1.403...v0.1.404)
(2023-12-10)

[0.1.403](https://github.com/starknet-io/starknet-docs/compare/v0.1.402...v0.1.403)
(2023-12-10)

[0.1.402](https://github.com/starknet-io/starknet-docs/compare/v0.1.401...v0.1.402)
(2023-12-10)

[0.1.401](https://github.com/starknet-io/starknet-docs/compare/v0.1.400...v0.1.401)
(2023-12-07)

[0.1.400](https://github.com/starknet-io/starknet-docs/compare/v0.1.399...v0.1.400)
(2023-12-07)

[0.1.399](https://github.com/starknet-io/starknet-docs/compare/v0.1.398...v0.1.399)
(2023-12-07)

[0.1.398](https://github.com/starknet-io/starknet-docs/compare/v0.1.397...v0.1.398)
(2023-12-07)

[0.1.397](https://github.com/starknet-io/starknet-docs/compare/v0.1.396...v0.1.397)
(2023-12-07)

[0.1.396](https://github.com/starknet-io/starknet-docs/compare/v0.1.395...v0.1.396)
(2023-12-06)

Bug Fixes

- Clarifications in Starknet state ([#970](https://github.com/starknet-io/starknet-docs/issues/970)) ([c7360ba](https://github.com/starknet-io/starknet-docs/commit/c7360ba3c9bd1ac36329beb5712c052006ea03d6))

[0.1.395](https://github.com/starknet-io/starknet-docs/compare/v0.1.394...v0.1.395)
(2023-12-03)

[0.1.394](https://github.com/starknet-io/starknet-docs/compare/v0.1.393...v0.1.394)
(2023-12-03)

[0.1.393](https://github.com/starknet-io/starknet-docs/compare/v0.1.392...v0.1.393)
(2023-12-03)

[0.1.392](https://github.com/starknet-io/starknet-docs/compare/v0.1.391...v0.1.392)
(2023-12-03)

[0.1.391](https://github.com/starknet-io/starknet-docs/compare/v0.1.390...v0.1.391)
(2023-12-03)

[0.1.390](https://github.com/starknet-io/starknet-docs/compare/v0.1.389...v0.1.390)
(2023-11-30)

[0.1.389](https://github.com/starknet-io/starknet-docs/compare/v0.1.388...v0.1.389)
(2023-11-30)

[0.1.388](https://github.com/starknet-io/starknet-docs/compare/v0.1.387...v0.1.388)
(2023-11-30)

[0.1.387](https://github.com/starknet-io/starknet-docs/compare/v0.1.386...v0.1.387)
(2023-11-30)

[0.1.386](https://github.com/starknet-io/starknet-docs/compare/v0.1.385...v0.1.386)
(2023-11-30)

[0.1.385](https://github.com/starknet-io/starknet-docs/compare/v0.1.384...v0.1.385)
(2023-11-30)

[0.1.384](https://github.com/starknet-io/starknet-docs/compare/v0.1.383...v0.1.384)
(2023-11-30)

[0.1.383](https://github.com/starknet-io/starknet-docs/compare/v0.1.382...v0.1.383)
(2023-11-29)

[0.1.382](https://github.com/starknet-io/starknet-docs/compare/v0.1.381...v0.1.382)
(2023-11-29)

[0.1.381](https://github.com/starknet-io/starknet-docs/compare/v0.1.380...v0.1.381)
(2023-11-29)

[0.1.380](https://github.com/starknet-io/starknet-docs/compare/v0.1.379...v0.1.380)
(2023-11-29)

[0.1.379](https://github.com/starknet-io/starknet-docs/compare/v0.1.378...v0.1.379)
(2023-11-29)

[0.1.378](https://github.com/starknet-io/starknet-docs/compare/v0.1.377...v0.1.378)
(2023-11-29)

[0.1.377](https://github.com/starknet-io/starknet-docs/compare/v0.1.376...v0.1.377)
(2023-11-29)

Bug Fixes

- More minor edits in account interface ([#982](https://github.com/starknet-io/starknet-docs/issues/982)) ([44289fb](https://github.com/starknet-io/starknet-docs/commit/44289fb4683c6f9aed053c6173477d9a0fc5643))

[0.1.376](https://github.com/starknet-io/starknet-docs/compare/v0.1.375...v0.1.376)
(2023-11-29)

[0.1.375](https://github.com/starknet-io/starknet-docs/compare/v0.1.374...v0.1.375)
(2023-11-29)

[0.1.374](https://github.com/starknet-io/starknet-docs/compare/v0.1.373...v0.1.374)
(2023-11-29)

[0.1.373](https://github.com/starknet-io/starknet-docs/compare/v0.1.372...v0.1.373)
(2023-11-29)

Bug Fixes

- Minor edits in approach.adoc ([#978](https://github.com/starknet-io/starknet-docs/issues/978)) ([20196c1](https://github.com/starknet-io/starknet-docs/commit/20196c10d29e0a44341812922eed54987089eb0c))

[0.1.372](https://github.com/starknet-io/starknet-docs/compare/v0.1.371...v0.1.372)
(2023-11-29)

[0.1.371](https://github.com/starknet-io/starknet-docs/compare/v0.1.370...v0.1.371)
(2023-11-29)

[0.1.370](https://github.com/starknet-io/starknet-docs/compare/v0.1.369...v0.1.370)
(2023-11-29)

[0.1.369](https://github.com/starknet-io/starknet-docs/compare/v0.1.368...v0.1.369)
(2023-11-28)

Bug Fixes

- normalize file path in Algolia index file ([#939](https://github.com/starknet-io/starknet-docs/issues/939)) ([2586169](https://github.com/starknet-io/starknet-docs/commit/2586169a94f68132168cf32f52f6cc030133611c)), closes [#910](https://github.com/starknet-io/starknet-docs/issues/910)

[0.1.368](https://github.com/starknet-io/starknet-docs/compare/v0.1.367...v0.1.368) (2023-11-28)

Features

- Add Starknet version to UI header ([#976](https://github.com/starknet-io/starknet-docs/issues/976)) ([33c3517](https://github.com/starknet-io/starknet-docs/commit/33c3517c26bb4e4dde9510f007f5c9f7bcf7bea03))

[0.1.367](https://github.com/starknet-io/starknet-docs/compare/v0.1.366...v0.1.367) (2023-11-28)

[0.1.366](https://github.com/starknet-io/starknet-docs/compare/v0.1.365...v0.1.366) (2023-11-28)

Bug Fixes

- apply new cairo syntax for Contract storage (Previously [#959](https://github.com/starknet-io/starknet-docs/issues/959), then [#960](https://github.com/starknet-io/starknet-docs/issues/960)) ([#975](https://github.com/starknet-io/starknet-docs/issues/975)) ([76a3156](https://github.com/starknet-io/starknet-docs/commit/76a3156e3dfef2607a507c1463219c527f67dc2c))

[0.1.365](https://github.com/starknet-io/starknet-docs/compare/v0.1.364...v0.1.365) (2023-11-28)

Bug Fixes

- Rewrote Contract tree hash ([#972](https://github.com/starknet-io/starknet-docs/issues/972)) ([df04d0c](https://github.com/starknet-io/starknet-docs/commit/df04d0ca9c7c0a85e01e090a104ade8c3dab4007))

[0.1.364](https://github.com/starknet-io/starknet-docs/compare/v0.1.363...v0.1.364) (2023-11-28)

[0.1.363](https://github.com/starknet-io/starknet-docs/compare/v0.1.362...v0.1.363) (2023-11-27)

[0.1.362](https://github.com/starknet-io/starknet-docs/compare/v0.1.361...v0.1.362) (2023-11-27)

[0.1.361](https://github.com/starknet-io/starknet-docs/compare/v0.1.360...v0.1.361)
(2023-11-23)

[0.1.360](https://github.com/starknet-io/starknet-docs/compare/v0.1.359...v0.1.360)
(2023-11-23)

[0.1.359](https://github.com/starknet-io/starknet-docs/compare/v0.1.357...v0.1.359)
(2023-11-23)

[0.1.358](https://github.com/starknet-io/starknet-docs/compare/v0.1.357...v0.1.358)
(2023-11-23)

[0.1.357](https://github.com/starknet-io/starknet-docs/compare/v0.1.356...v0.1.357)
(2023-11-23)

[0.1.356](https://github.com/starknet-io/starknet-docs/compare/v0.1.355...v0.1.356)
(2023-11-22)

[0.1.355](https://github.com/starknet-io/starknet-docs/compare/v0.1.354...v0.1.355)
(2023-11-22)

[0.1.354](https://github.com/starknet-io/starknet-docs/compare/v0.1.353...v0.1.354)
(2023-11-21)

Bug Fixes

- typo in docs releasing-starknet-docs-on-github ([#931](https://github.com/starknet-io/starknet-docs/issues/931)) ([#950](https://github.com/starknet-io/starknet-docs/issues/950)) ([5447bd2](https://github.com/starknet-io/starknet-docs/commit/5447bd2a1bf439f1819084df7bc5d9343855ae25))

[0.1.353](https://github.com/starknet-io/starknet-docs/compare/v0.1.352...v0.1.353)
(2023-11-21)

Bug Fixes

- Minor edits in simplified_transaction_flow.adoc ([#944](https://github.com/starknet-io/starknet-docs/issues/944)) ([e7921a0](https://github.com/starknet-io/starknet-docs/commit/e7921a013fb7c3c92a53732ff2c74efb596a3882))

[0.1.352](https://github.com/starknet-io/starknet-docs/compare/v0.1.351...v0.1.352)
(2023-11-19)

Bug Fixes

- several typos in the documentation ([#929](https://github.com/starknet-io/starknet-docs/issues/929)) ([#937](https://github.com/starknet-io/starknet-docs/issues/937)) ([7caff6d](https://github.com/starknet-io/starknet-docs/commit/7caff6d6a2e1946a613e5939c8230ee6afb72f1))

[0.1.351](https://github.com/starknet-io/starknet-docs/compare/v0.1.350...v0.1.351) (2023-11-17)

[0.1.350](https://github.com/starknet-io/starknet-docs/compare/v0.1.349...v0.1.350) (2023-11-16)

[0.1.349](https://github.com/starknet-io/starknet-docs/compare/v0.1.348...v0.1.349) (2023-11-15)

[0.1.348](https://github.com/starknet-io/starknet-docs/compare/v0.1.347...v0.1.348) (2023-11-13)

Bug Fixes

- Add Pederson content back in ([#927](https://github.com/starknet-io/starknet-docs/issues/927)) ([1e25ab9](https://github.com/starknet-io/starknet-docs/commit/1e25ab9dec9bc94378eb3b0da528b91f55a72998))

- Edit get_block_hash syscall ([#909](https://github.com/starknet-io/starknet-docs/issues/909)) ([3e3c03d](https://github.com/starknet-io/starknet-docs/commit/3e3c03d2861bf382db0a6827afd1537218bf48f4))

[0.1.347](https://github.com/starknet-io/starknet-docs/compare/v0.1.346...v0.1.347) (2023-11-08)

[0.1.346](https://github.com/starknet-io/starknet-docs/compare/v0.1.345...v0.1.346) (2023-11-07)

[0.1.345](https://github.com/starknet-io/starknet-docs/compare/v0.1.344...v0.1.345) (2023-11-07)

[0.1.344](https://github.com/starknet-io/starknet-docs/compare/v0.1.343...v0.1.344) (2023-10-31)

[0.1.343](https://github.com/starknet-io/starknet-docs/compare/v0.1.342...v0.1.343) (2023-10-25)

[0.1.342](https://github.com/starknet-io/starknet-docs/compare/v0.1.341...v0.1.342) (2023-10-24)

[0.1.341](https://github.com/starknet-io/starknet-docs/compare/v0.1.340...v0.1.341) (2023-10-24)

[0.1.340](https://github.com/starknet-io/starknet-docs/compare/v0.1.339...v0.1.340)
(2023-10-24)

[0.1.339](https://github.com/starknet-io/starknet-docs/compare/v0.1.338...v0.1.339)
(2023-10-24)

[0.1.338](https://github.com/starknet-io/starknet-docs/compare/v0.1.337...v0.1.338)
(2023-10-10)

[0.1.337](https://github.com/starknet-io/starknet-docs/compare/v0.1.335...v0.1.337)
(2023-10-10)

[0.1.332](https://github.com/starknet-io/starknet-docs/compare/v0.1.331...v0.1.332)
(2023-09-29)

[0.1.332](https://github.com/starknet-io/starknet-docs/compare/v0.1.331...v0.1.332)
(2023-09-29)

[0.1.331](https://github.com/starknet-io/starknet-docs/compare/v0.1.330...v0.1.331)
(2023-09-29)

[0.1.325](https://github.com/starknet-io/starknet-docs/compare/v0.1.324...v0.1.325)
(2023-09-20)

[0.1.325](https://github.com/starknet-io/starknet-docs/compare/v0.1.324...v0.1.325)
(2023-09-20)

[0.1.324](https://github.com/starknet-io/starknet-docs/compare/v0.1.323...v0.1.324)
(2023-09-20)

[0.1.323](https://github.com/starknet-io/starknet-docs/compare/v0.1.322...v0.1.323)
(2023-09-19)

[0.1.322](https://github.com/starknet-io/starknet-docs/compare/v0.1.321...v0.1.322)
(2023-09-19)

[0.1.321](https://github.com/starknet-io/starknet-docs/compare/v0.1.320...v0.1.321)
(2023-09-19)

[0.1.320](https://github.com/starknet-io/starknet-docs/compare/v0.1.319...v0.1.320)
(2023-09-18)

[0.1.319](https://github.com/starknet-io/starknet-docs/compare/v0.1.318...v0.1.319)
(2023-09-15)

[0.1.318](https://github.com/starknet-io/starknet-docs/compare/v0.1.317...v0.1.318)
(2023-09-15)

[0.1.317](https://github.com/starknet-io/starknet-docs/compare/v0.1.316...v0.1.317)
(2023-09-15)

[0.1.316](https://github.com/starknet-io/starknet-docs/compare/v0.1.315...v0.1.316)
(2023-09-15)

[0.1.315](https://github.com/starknet-io/starknet-docs/compare/v0.1.314...v0.1.315)
(2023-09-14)

[0.1.314](https://github.com/starknet-io/starknet-docs/compare/v0.1.313...v0.1.314)
(2023-09-14)

[0.1.313](https://github.com/starknet-io/starknet-docs/compare/v0.1.312...v0.1.313)
(2023-09-14)

[0.1.312](https://github.com/starknet-io/starknet-docs/compare/v0.1.311...v0.1.312)
(2023-09-13)

[0.1.311](https://github.com/starknet-io/starknet-docs/compare/v0.1.310...v0.1.311)
(2023-09-13)

[0.1.310](https://github.com/starknet-io/starknet-docs/compare/v0.1.309...v0.1.310)
(2023-09-13)

[0.1.309](https://github.com/starknet-io/starknet-docs/compare/v0.1.308...v0.1.309)
(2023-09-13)

[0.1.308](https://github.com/starknet-io/starknet-docs/compare/v0.1.306...v0.1.308)
(2023-09-13)

[0.1.306](https://github.com/starknet-io/starknet-docs/compare/v0.1.305...v0.1.306)
(2023-09-13)

[0.1.305](https://github.com/starknet-io/starknet-docs/compare/v0.1.304...v0.1.305)
(2023-09-13)

[0.1.304](https://github.com/starknet-io/starknet-docs/compare/v0.1.303...v0.1.304)
(2023-09-13)

[0.1.303](https://github.com/starknet-io/starknet-docs/compare/v0.1.302...v0.1.303)
(2023-09-13)

[0.1.302](https://github.com/starknet-io/starknet-docs/compare/v0.1.301...v0.1.302)

(2023-09-13)

[0.1.301](https://github.com/starknet-io/starknet-docs/compare/v0.1.300...v0.1.301)
(2023-09-13)

[0.1.300](https://github.com/starknet-io/starknet-docs/compare/v0.1.299...v0.1.300)
(2023-09-12)

[0.1.299](https://github.com/starknet-io/starknet-docs/compare/v0.1.298...v0.1.299)
(2023-09-12)

[0.1.298](https://github.com/starknet-io/starknet-docs/compare/v0.1.297...v0.1.298)
(2023-09-12)

[0.1.297](https://github.com/starknet-io/starknet-docs/compare/v0.1.296...v0.1.297)
(2023-09-12)

[0.1.296](https://github.com/starknet-io/starknet-docs/compare/v0.1.295...v0.1.296)
(2023-09-12)

[0.1.295](https://github.com/starknet-io/starknet-docs/compare/v0.1.294...v0.1.295)
(2023-09-12)

[0.1.294](https://github.com/starknet-io/starknet-docs/compare/v0.1.293...v0.1.294)
(2023-09-12)

Bug Fixes

- Update token-bridge.adoc: Starkgate -> StarkGate ([#795](https://github.com/starknet-io/starknet-docs/issues/795)) ([c86c2a9](https://github.com/starknet-io/starknet-docs/commit/c86c2a995530d94a092d958f7d8f425a83d08046))

[0.1.293](https://github.com/starknet-io/starknet-docs/compare/v0.1.292...v0.1.293)
(2023-09-11)

[0.1.292](https://github.com/starknet-io/starknet-docs/compare/v0.1.291...v0.1.292)
(2023-09-11)

[0.1.291](https://github.com/starknet-io/starknet-docs/compare/v0.1.290...v0.1.291)
(2023-09-11)

[0.1.290](https://github.com/starknet-io/starknet-docs/compare/v0.1.289...v0.1.290)
(2023-09-08)

[0.1.289](https://github.com/starknet-io/starknet-docs/compare/v0.1.288...v0.1.289)
(2023-09-08)

[0.1.288](https://github.com/starknet-io/starknet-docs/compare/v0.1.287...v0.1.288)
(2023-09-08)

[0.1.287](https://github.com/starknet-io/starknet-docs/compare/v0.1.286...v0.1.287)
(2023-09-08)

[0.1.286](https://github.com/starknet-io/starknet-docs/compare/v0.1.285...v0.1.286)
(2023-09-08)

[0.1.285](https://github.com/starknet-io/starknet-docs/compare/v0.1.284...v0.1.285)
(2023-09-07)

[0.1.284](https://github.com/starknet-io/starknet-docs/compare/v0.1.283...v0.1.284)
(2023-09-05)

[0.1.283](https://github.com/starknet-io/starknet-docs/compare/v0.1.282...v0.1.283)
(2023-09-05)

[0.1.282](https://github.com/starknet-io/starknet-docs/compare/v0.1.281...v0.1.282)
(2023-09-05)

[0.1.281](https://github.com/starknet-io/starknet-docs/compare/v0.1.280...v0.1.281)
(2023-09-05)

[0.1.280](https://github.com/starknet-io/starknet-docs/compare/v0.1.279...v0.1.280)
(2023-09-04)

[0.1.279](https://github.com/starknet-io/starknet-docs/compare/v0.1.278...v0.1.279)
(2023-09-04)

[0.1.278](https://github.com/starknet-io/starknet-docs/compare/v0.1.277...v0.1.278)
(2023-09-04)

[0.1.277](https://github.com/starknet-io/starknet-docs/compare/v0.1.276...v0.1.277)
(2023-09-01)

[0.1.276](https://github.com/starknet-io/starknet-docs/compare/v0.1.275...v0.1.276)
(2023-08-29)

[0.1.275](https://github.com/starknet-io/starknet-docs/compare/v0.1.274...v0.1.275)
(2023-08-29)

[0.1.274](https://github.com/starknet-io/starknet-docs/compare/v0.1.273...v0.1.274)
(2023-08-29)

[0.1.273](https://github.com/starknet-io/starknet-docs/compare/v0.1.272...v0.1.273)
(2023-08-24)

[0.1.272](https://github.com/starknet-io/starknet-docs/compare/v0.1.271...v0.1.272)
(2023-08-24)

[0.1.271](https://github.com/starknet-io/starknet-docs/compare/v0.1.270...v0.1.271)
(2023-08-24)

[0.1.270](https://github.com/starknet-io/starknet-docs/compare/v0.1.269...v0.1.270)
(2023-08-24)

[0.1.269](https://github.com/starknet-io/starknet-docs/compare/v0.1.268...v0.1.269)
(2023-08-23)

[0.1.268](https://github.com/starknet-io/starknet-docs/compare/v0.1.267...v0.1.268)
(2023-08-23)

[0.1.267](https://github.com/starknet-io/starknet-docs/compare/v0.1.266...v0.1.267)
(2023-08-23)

Bug Fixes

- Edits to initial sentence. ([#741](https://github.com/starknet-io/starknet-docs/issues/741)) ([26ab9ab](https://github.com/starknet-io/starknet-docs/commit/26ab9abde2897a1fc8a811c16c366982dc2bb87e))
- remove cairo 0 examples ([#729](https://github.com/starknet-io/starknet-docs/issues/729)) ([327dc7b](https://github.com/starknet-io/starknet-docs/commit/327dc7b33d3976865837efe478065e5edfc7ca5))

[0.1.266](https://github.com/starknet-io/starknet-docs/compare/v0.1.265...v0.1.266)
(2023-08-23)

Bug Fixes

- Fix broken link, clarify sentence and request for clarification. ([#751](https://github.com/starknet-io/starknet-docs/issues/751)) ([139bb77](https://github.com/starknet-io/starknet-docs/commit/139bb77a6ed60fb0d396d0ca89f401c220c40ffb))

[0.1.265](https://github.com/starknet-io/starknet-docs/compare/v0.1.264...v0.1.265)
(2023-08-23)

[0.1.264](https://github.com/starknet-io/starknet-docs/compare/v0.1.263...v0.1.264)
(2023-08-22)

[0.1.263](https://github.com/starknet-io/starknet-docs/compare/v0.1.262...v0.1.263)

(2023-08-22)

[0.1.262](https://github.com/starknet-io/starknet-docs/compare/v0.1.261...v0.1.262)
(2023-08-22)

[0.1.261](https://github.com/starknet-io/starknet-docs/compare/v0.1.260...v0.1.261)
(2023-08-22)

Bug Fixes

- Fix broken link, clarify sentence and request for clarification. ([#751](https://github.com/starknet-io/starknet-docs/issues/751)) ([#759](https://github.com/starknet-io/starknet-docs/issues/759)) ([94255ef](https://github.com/starknet-io/starknet-docs/commit/94255effb43283f20b51f2f64250cbcedc775030))

[0.1.260](https://github.com/starknet-io/starknet-docs/compare/v0.1.259...v0.1.260)
(2023-08-22)

[0.1.259](https://github.com/starknet-io/starknet-docs/compare/v0.1.258...v0.1.259)
(2023-08-21)

[0.1.258](https://github.com/starknet-io/starknet-docs/compare/v0.1.257...v0.1.258)
(2023-08-21)

[0.1.257](https://github.com/starknet-io/starknet-docs/compare/v0.1.256...v0.1.257)
(2023-08-21)

[0.1.256](https://github.com/starknet-io/starknet-docs/compare/v0.1.255...v0.1.256)
(2023-08-21)

[0.1.255](https://github.com/starknet-io/starknet-docs/compare/v0.1.254...v0.1.255)
(2023-08-21)

[0.1.254](https://github.com/starknet-io/starknet-docs/compare/v0.1.253...v0.1.254)
(2023-08-21)

[0.1.253](https://github.com/starknet-io/starknet-docs/compare/v0.1.252...v0.1.253)
(2023-08-21)

[0.1.252](https://github.com/starknet-io/starknet-docs/compare/v0.1.251...v0.1.252)
(2023-08-15)

[0.1.251](https://github.com/starknet-io/starknet-docs/compare/v0.1.250...v0.1.251)
(2023-08-15)

[0.1.250](https://github.com/starknet-io/starknet-docs/compare/v0.1.249...v0.1.250)

(2023-08-15)

[0.1.249](https://github.com/starknet-io/starknet-docs/compare/v0.1.248...v0.1.249)
(2023-08-15)

[0.1.248](https://github.com/starknet-io/starknet-docs/compare/v0.1.247...v0.1.248)
(2023-08-15)

[0.1.247](https://github.com/starknet-io/starknet-docs/compare/v0.1.246...v0.1.247)
(2023-08-15)

[0.1.246](https://github.com/starknet-io/starknet-docs/compare/v0.1.245...v0.1.246)
(2023-08-15)

[0.1.245](https://github.com/starknet-io/starknet-docs/compare/v0.1.244...v0.1.245)
(2023-08-15)

[0.1.244](https://github.com/starknet-io/starknet-docs/compare/v0.1.243...v0.1.244)
(2023-08-09)

[0.1.243](https://github.com/starknet-io/starknet-docs/compare/v0.1.242...v0.1.243)
(2023-08-09)

[0.1.242](https://github.com/starknet-io/starknet-docs/compare/v0.1.241...v0.1.242)
(2023-08-07)

[0.1.241](https://github.com/starknet-io/starknet-docs/compare/v0.1.240...v0.1.241)
(2023-08-07)

[0.1.240](https://github.com/starknet-io/starknet-docs/compare/v0.1.239...v0.1.240)
(2023-08-04)

[0.1.239](https://github.com/starknet-io/starknet-docs/compare/v0.1.238...v0.1.239)
(2023-08-03)

[0.1.238](https://github.com/starknet-io/starknet-docs/compare/v0.1.237...v0.1.238)
(2023-08-03)

[0.1.237](https://github.com/starknet-io/starknet-docs/compare/v0.1.236...v0.1.237)
(2023-08-02)

[0.1.236](https://github.com/starknet-io/starknet-docs/compare/v0.1.235...v0.1.236)
(2023-08-02)

Bug Fixes

- removed image from L1-L2 communication ([#698](https://github.com/starknet-io/starknet-docs/issues/698)) ([84cac9a](https://github.com/starknet-io/starknet-docs/commit/84cac9adf162a1e40e53ee339f63aa5c930dc733))

[0.1.235](https://github.com/starknet-io/starknet-docs/compare/v0.1.234...v0.1.235) (2023-07-31)

[0.1.234](https://github.com/starknet-io/starknet-docs/compare/v0.1.233...v0.1.234) (2023-07-25)

Features

- Use remote UI ([#695](https://github.com/starknet-io/starknet-docs/issues/695)) ([49aa719](https://github.com/starknet-io/starknet-docs/commit/49aa719f20f1f216431176989e69991827d46bce))

[0.1.233](https://github.com/starknet-io/starknet-docs/compare/v0.1.232...v0.1.233) (2023-07-18)

Bug Fixes

- typos ([#693](https://github.com/starknet-io/starknet-docs/issues/693)) ([e428572](https://github.com/starknet-io/starknet-docs/commit/e428572828788a36c006d61e3d8a010d4eb38395))

[0.1.232](https://github.com/starknet-io/starknet-docs/compare/v0.1.231...v0.1.232) (2023-07-18)

[0.1.231](https://github.com/starknet-io/starknet-docs/compare/v0.1.230...v0.1.231) (2023-07-14)

[0.1.230](https://github.com/starknet-io/starknet-docs/compare/v0.1.229...v0.1.230) (2023-07-14)

[0.1.229](https://github.com/starknet-io/starknet-docs/compare/v0.1.228...v0.1.229) (2023-07-13)

[0.1.228](https://github.com/starknet-io/starknet-docs/compare/v0.1.227...v0.1.228) (2023-07-13)

Features

- Add style guide and more details for contributing. ([#519](https://github.com/starknet-io/starknet-docs/issues/519)) ([a52bf4f](https://github.com/starknet-io/starknet-docs/commit/a52bf4ff9488ab06542a9f30e5c687149fdb7db8))

[0.1.227](https://github.com/starknet-io/starknet-docs/compare/v0.1.226...v0.1.227)
(2023-07-12)

[0.1.226](https://github.com/starknet-io/starknet-docs/compare/v0.1.225...v0.1.226)
(2023-07-12)

[0.1.225](https://github.com/starknet-io/starknet-docs/compare/v0.1.224...v0.1.225)
(2023-07-12)

[0.1.224](https://github.com/starknet-io/starknet-docs/compare/v0.1.223...v0.1.224)
(2023-07-12)

[0.1.223](https://github.com/starknet-io/starknet-docs/compare/v0.1.222...v0.1.223)
(2023-07-12)

[0.1.222](https://github.com/starknet-io/starknet-docs/compare/v0.1.221...v0.1.222)
(2023-07-12)

[0.1.221](https://github.com/starknet-io/starknet-docs/compare/v0.1.220...v0.1.221)
(2023-07-12)

[0.1.220](https://github.com/starknet-io/starknet-docs/compare/v0.1.219...v0.1.220)
(2023-07-12)

[0.1.219](https://github.com/starknet-io/starknet-docs/compare/v0.1.218...v0.1.219)
(2023-07-12)

[0.1.218](https://github.com/starknet-io/starknet-docs/compare/v0.1.217...v0.1.218)
(2023-07-12)

[0.1.217](https://github.com/starknet-io/starknet-docs/compare/v0.1.216...v0.1.217)
(2023-07-11)

[0.1.216](https://github.com/starknet-io/starknet-docs/compare/v0.1.215...v0.1.216)
(2023-07-11)

[0.1.215](https://github.com/starknet-io/starknet-docs/compare/v0.1.214...v0.1.215)
(2023-07-10)

Features

- add pathfinder versions ([#651](https://github.com/starknet-io/starknet-docs/issues/651)) ([b9f5834](https://github.com/starknet-io/starknet-docs/commit/b9f5834ee46027034d16e47a4e234d5de1ae50ef))

[0.1.214](https://github.com/starknet-io/starknet-docs/compare/v0.1.213...v0.1.214)

(2023-07-10)

Features

- add pathfinder versions ([#651](https://github.com/starknet-io/starknet-docs/issues/651)) ([#652](https://github.com/starknet-io/starknet-docs/issues/652)) ([c2a90bf](https://github.com/starknet-io/starknet-docs/commit/c2a90bf4ba72e807f3baec14399778a2ac35963f))

[0.1.213](https://github.com/starknet-io/starknet-docs/compare/v0.1.212...v0.1.213)
(2023-07-10)

Bug Fixes

- data availability href fix ([#639](https://github.com/starknet-io/starknet-docs/issues/639)) ([6d0514c](https://github.com/starknet-io/starknet-docs/commit/6d0514c0e81ba4b0ea933500c915382c682c6292))

[0.1.212](https://github.com/starknet-io/starknet-docs/compare/v0.1.211...v0.1.212)
(2023-07-10)

[0.1.211](https://github.com/starknet-io/starknet-docs/compare/v0.1.210...v0.1.211)
(2023-07-07)

[0.1.210](https://github.com/starknet-io/starknet-docs/compare/v0.1.209...v0.1.210)
(2023-07-07)

[0.1.209](https://github.com/starknet-io/starknet-docs/compare/v0.1.208...v0.1.209)
(2023-07-07)

[0.1.208](https://github.com/starknet-io/starknet-docs/compare/v0.1.207...v0.1.208)
(2023-07-07)

[0.1.207](https://github.com/starknet-io/starknet-docs/compare/v0.1.206...v0.1.207)
(2023-07-07)

[0.1.206](https://github.com/starknet-io/starknet-docs/compare/v0.1.205...v0.1.206)
(2023-07-07)

Bug Fixes

- Minor style fixes. ([#631](https://github.com/starknet-io/starknet-docs/issues/631)) ([47bf7eb](https://github.com/starknet-io/starknet-docs/commit/47bf7ebfec9199c6135eaeb214fa2d0a7e6f4a8a))

[0.1.205](https://github.com/starknet-io/starknet-docs/compare/v0.1.204...v0.1.205)

(2023-07-06)

[0.1.204](https://github.com/starknet-io/starknet-docs/compare/v0.1.203...v0.1.204)
(2023-07-06)

[0.1.203](https://github.com/starknet-io/starknet-docs/compare/v0.1.202...v0.1.203)
(2023-07-05)

[0.1.202](https://github.com/starknet-io/starknet-docs/compare/v0.1.201...v0.1.202)
(2023-07-05)

[0.1.201](https://github.com/starknet-io/starknet-docs/compare/v0.1.200...v0.1.201)
(2023-07-05)

[0.1.200](https://github.com/starknet-io/starknet-docs/compare/v0.1.199...v0.1.200)
(2023-07-05)

[0.1.199](https://github.com/starknet-io/starknet-docs/compare/v0.1.198...v0.1.199)
(2023-07-05)

[0.1.198](https://github.com/starknet-io/starknet-docs/compare/v0.1.197...v0.1.198)
(2023-07-05)

[0.1.197](https://github.com/starknet-io/starknet-docs/compare/v0.1.196...v0.1.197)
(2023-07-04)

[0.1.196](https://github.com/starknet-io/starknet-docs/compare/v0.1.195...v0.1.196)
(2023-06-30)

[0.1.195](https://github.com/starknet-io/starknet-docs/compare/v0.1.194...v0.1.195)
(2023-06-28)

[0.1.194](https://github.com/starknet-io/starknet-docs/compare/v0.1.193...v0.1.194)
(2023-06-22)

[0.1.193](https://github.com/starknet-io/starknet-docs/compare/v0.1.192...v0.1.193)
(2023-06-22)

[0.1.192](https://github.com/starknet-io/starknet-docs/compare/v0.1.191...v0.1.192)
(2023-06-21)

[0.1.191](https://github.com/starknet-io/starknet-docs/compare/v0.1.190...v0.1.191)
(2023-06-21)

[0.1.190](https://github.com/starknet-io/starknet-docs/compare/v0.1.189...v0.1.190)
(2023-06-21)

Bug Fixes

- minor fixes in "Starknet full-nodes and API providers" ([#600](https://github.com/starknet-io/starknet-docs/issues/600)) ([05b20e3](https://github.com/starknet-io/starknet-docs/commit/05b20e31dee389bc0b0bae81af61d686922d0842))

[0.1.189](https://github.com/starknet-io/starknet-docs/compare/v0.1.188...v0.1.189) (2023-06-21)

[0.1.188](https://github.com/starknet-io/starknet-docs/compare/v0.1.187...v0.1.188) (2023-06-21)

[0.1.187](https://github.com/starknet-io/starknet-docs/compare/v0.1.186...v0.1.187) (2023-06-21)

[0.1.186](https://github.com/starknet-io/starknet-docs/compare/v0.1.185...v0.1.186) (2023-06-21)

[0.1.185](https://github.com/starknet-io/starknet-docs/compare/v0.1.184...v0.1.185) (2023-06-21)

[0.1.184](https://github.com/starknet-io/starknet-docs/compare/v0.1.183...v0.1.184) (2023-06-21)

[0.1.183](https://github.com/starknet-io/starknet-docs/compare/v0.1.182...v0.1.183) (2023-06-21)

[0.1.182](https://github.com/starknet-io/starknet-docs/compare/v0.1.181...v0.1.182) (2023-06-21)

[0.1.181](https://github.com/starknet-io/starknet-docs/compare/v0.1.180...v0.1.181) (2023-06-21)

[0.1.180](https://github.com/starknet-io/starknet-docs/compare/v0.1.179...v0.1.180) (2023-06-21)

[0.1.179](https://github.com/starknet-io/starknet-docs/compare/v0.1.178...v0.1.179) (2023-06-21)

[0.1.178](https://github.com/starknet-io/starknet-docs/compare/v0.1.177...v0.1.178) (2023-06-21)

[0.1.177](https://github.com/starknet-io/starknet-docs/compare/v0.1.176...v0.1.177) (2023-06-21)

[0.1.176](https://github.com/starknet-io/starknet-docs/compare/v0.1.175...v0.1.176) (2023-06-21)

[0.1.175](https://github.com/starknet-io/starknet-docs/compare/v0.1.174...v0.1.175) (2023-06-20)

Bug Fixes

- minor fixes in "System Calls/Cairo 0" ([#567](https://github.com/starknet-io/starknet-docs/issues/567)) ([4c946c1](https://github.com/starknet-io/starknet-docs/commit/4c946c12569f361530c25e7fac45cecf280b28b1)), closes [#545](https://github.com/starknet-io/starknet-docs/issues/545) [#540](https://github.com/starknet-io/starknet-docs/issues/540) [#544](https://github.com/starknet-io/starknet-docs/issues/544)

[0.1.174](https://github.com/starknet-io/starknet-docs/compare/v0.1.173...v0.1.174) (2023-06-20)

[0.1.173](https://github.com/starknet-io/starknet-docs/compare/v0.1.172...v0.1.173) (2023-06-20)

[0.1.172](https://github.com/starknet-io/starknet-docs/compare/v0.1.171...v0.1.172) (2023-06-20)

Bug Fixes

- minor fixes in "Class hash" ([#556](https://github.com/starknet-io/starknet-docs/issues/556)) ([a0b4743](https://github.com/starknet-io/starknet-docs/commit/a0b4743a172889cd014d562ef0affcddaaed837e))
- minor fixes in "Starknet state" ([#555](https://github.com/starknet-io/starknet-docs/issues/555)) ([d03ec6a](https://github.com/starknet-io/starknet-docs/commit/d03ec6a45a10d4798e9b633199a0532ed43712c8))

[0.1.171](https://github.com/starknet-io/starknet-docs/compare/v0.1.169...v0.1.171) (2023-06-19)

[0.1.159](https://github.com/starknet-io/starknet-docs/compare/v0.1.158...v0.1.159) (2023-06-13)

[0.1.158](https://github.com/starknet-io/starknet-docs/compare/v0.1.157...v0.1.158) (2023-06-13)

[0.1.157](https://github.com/starknet-io/starknet-docs/compare/v0.1.156...v0.1.157) (2023-06-12)

[0.1.156](https://github.com/starknet-io/starknet-docs/compare/v0.1.155...v0.1.156)

(2023-06-12)

[0.1.155](https://github.com/starknet-io/starknet-docs/compare/v0.1.154...v0.1.155)
(2023-06-09)

Bug Fixes

- minor fixes in "Block structure" ([#526](https://github.com/starknet-io/starknet-docs/issues/526)) ([0ea5756](https://github.com/starknet-io/starknet-docs/commit/0ea57568f0859c8baf3d8fa36728755375ab5651))
- minor fixes in "Contract address" ([#525](https://github.com/starknet-io/starknet-docs/issues/525)) ([244666e](https://github.com/starknet-io/starknet-docs/commit/244666e851792648e0ddeb58b7227a8900510d87))
- minor fixes in "L1-L2 messaging" ([#532](https://github.com/starknet-io/starknet-docs/issues/532)) ([d287023](https://github.com/starknet-io/starknet-docs/commit/d287023e236adb073a34edea79576554b8393fe9))
- minor fixes in "Transaction lifecycle" ([#527](https://github.com/starknet-io/starknet-docs/issues/527)) ([872e18c](https://github.com/starknet-io/starknet-docs/commit/872e18c0fad091b0dc674a37b8bb6dd79ab0cd03))

[0.1.154](https://github.com/starknet-io/starknet-docs/compare/v0.1.153...v0.1.154)
(2023-06-09)

[0.1.153](https://github.com/starknet-io/starknet-docs/compare/v0.1.152...v0.1.153)
(2023-06-09)

[0.1.152](https://github.com/starknet-io/starknet-docs/compare/v0.1.151...v0.1.152)
(2023-06-06)

[0.1.151](https://github.com/starknet-io/starknet-docs/compare/v0.1.150...v0.1.151)
(2023-06-05)

[0.1.150](https://github.com/starknet-io/starknet-docs/compare/v0.1.149...v0.1.150)
(2023-06-05)

[0.1.149](https://github.com/starknet-io/starknet-docs/compare/v0.1.148...v0.1.149)
(2023-06-03)

[0.1.148](https://github.com/starknet-io/starknet-docs/compare/v0.1.147...v0.1.148)
(2023-05-26)

[0.1.147](https://github.com/starknet-io/starknet-docs/compare/v0.1.146...v0.1.147)
(2023-05-26)

[0.1.146](https://github.com/starknet-io/starknet-docs/compare/v0.1.145...v0.1.146)
(2023-05-25)

[0.1.145](https://github.com/starknet-io/starknet-docs/compare/v0.1.144...v0.1.145)
(2023-05-25)

[0.1.144](https://github.com/starknet-io/starknet-docs/compare/v0.1.143...v0.1.144)
(2023-05-24)

[0.1.143](https://github.com/starknet-io/starknet-docs/compare/v0.1.142...v0.1.143)
(2023-05-24)

[0.1.142](https://github.com/starknet-io/starknet-docs/compare/v0.1.141...v0.1.142)
(2023-05-24)

[0.1.141](https://github.com/starknet-io/starknet-docs/compare/v0.1.140...v0.1.141)
(2023-05-24)

[0.1.140](https://github.com/starknet-io/starknet-docs/compare/v0.1.139...v0.1.140)
(2023-05-24)

[0.1.139](https://github.com/starknet-io/starknet-docs/compare/v0.1.138...v0.1.139)
(2023-05-24)

[0.1.138](https://github.com/starknet-io/starknet-docs/compare/v0.1.137...v0.1.138)
(2023-05-23)

[0.1.137](https://github.com/starknet-io/starknet-docs/compare/v0.1.136...v0.1.137)
(2023-05-23)

[0.1.136](https://github.com/starknet-io/starknet-docs/compare/v0.1.135...v0.1.136)
(2023-05-22)

[0.1.135](https://github.com/starknet-io/starknet-docs/compare/v0.1.133...v0.1.135)
(2023-05-19)

[0.1.134](https://github.com/starknet-io/starknet-docs/compare/v0.1.133...v0.1.134)
(2023-05-19)

[0.1.132](https://github.com/starknet-io/starknet-docs/compare/v0.1.131...v0.1.132)
(2023-05-19)

[0.1.131](https://github.com/starknet-io/starknet-docs/compare/v0.1.130...v0.1.131)
(2023-05-18)

[0.1.130](https://github.com/starknet-io/starknet-docs/compare/v0.1.129...v0.1.130)
(2023-05-18)

[0.1.129](https://github.com/starknet-io/starknet-docs/compare/v0.1.128...v0.1.129)
(2023-05-17)

[0.1.128](https://github.com/starknet-io/starknet-docs/compare/v0.1.127...v0.1.128)
(2023-05-16)

[0.1.127](https://github.com/starknet-io/starknet-docs/compare/v0.1.126...v0.1.127)
(2023-05-11)

[0.1.126](https://github.com/starknet-io/starknet-docs/compare/v0.1.125...v0.1.126)
(2023-05-11)

[0.1.125](https://github.com/starknet-io/starknet-docs/compare/v0.1.124...v0.1.125)
(2023-05-10)

[0.1.124](https://github.com/starknet-io/starknet-docs/compare/v0.1.123...v0.1.124)
(2023-05-10)

[0.1.123](https://github.com/starknet-io/starknet-docs/compare/v0.1.122...v0.1.123)
(2023-05-10)

[0.1.122](https://github.com/starknet-io/starknet-docs/compare/v0.1.121...v0.1.122)
(2023-05-10)

[0.1.121](https://github.com/starknet-io/starknet-docs/compare/v0.1.120...v0.1.121)
(2023-05-09)

[0.1.120](https://github.com/starknet-io/starknet-docs/compare/v0.1.119...v0.1.120)
(2023-04-21)

[0.1.119](https://github.com/starknet-io/starknet-docs/compare/v0.1.118...v0.1.119)
(2023-04-21)

[0.1.118](https://github.com/starknet-io/starknet-docs/compare/v0.1.117...v0.1.118)
(2023-04-20)

[0.1.117](https://github.com/starknet-io/starknet-docs/compare/v0.1.116...v0.1.117)
(2023-04-20)

[0.1.116](https://github.com/starknet-io/starknet-docs/compare/v0.1.115...v0.1.116)
(2023-04-18)

[0.1.115](https://github.com/starknet-io/starknet-docs/compare/v0.1.114...v0.1.115)
(2023-04-17)

[0.1.114](https://github.com/starknet-io/starknet-docs/compare/v0.1.113...v0.1.114)

(2023-04-13)

[0.1.113](https://github.com/starknet-io/starknet-docs/compare/v0.1.112...v0.1.113)
(2023-04-11)

[0.1.112](https://github.com/starknet-io/starknet-docs/compare/v0.1.111...v0.1.112)
(2023-04-11)

[0.1.111](https://github.com/starknet-io/starknet-docs/compare/v0.1.110...v0.1.111)
(2023-04-06)

[0.1.110](https://github.com/starknet-io/starknet-docs/compare/v0.1.109...v0.1.110)
(2023-04-06)

[0.1.109](https://github.com/starknet-io/starknet-docs/compare/v0.1.108...v0.1.109)
(2023-04-06)

[0.1.108](https://github.com/starknet-io/starknet-docs/compare/v0.1.107...v0.1.108)
(2023-04-06)

[0.1.107](https://github.com/starknet-io/starknet-docs/compare/v0.1.106...v0.1.107)
(2023-04-05)

[0.1.106](https://github.com/starknet-io/starknet-docs/compare/v0.1.105...v0.1.106)
(2023-03-22)

[0.1.105](https://github.com/starknet-io/starknet-docs/compare/v0.1.104...v0.1.105)
(2023-03-21)

[0.1.104](https://github.com/starknet-io/starknet-docs/compare/v0.1.103...v0.1.104)
(2023-03-21)

[0.1.103](https://github.com/starknet-io/starknet-docs/compare/v0.1.102...v0.1.103)
(2023-03-21)

[0.1.102](https://github.com/starknet-io/starknet-docs/compare/v0.1.101...v0.1.102)
(2023-03-20)

[0.1.101](https://github.com/starknet-io/starknet-docs/compare/v0.1.100...v0.1.101)
(2023-03-15)

[0.1.100](https://github.com/starknet-io/starknet-docs/compare/v0.1.99...v0.1.100)
(2023-03-13)

[0.1.99](https://github.com/starknet-io/starknet-docs/compare/v0.1.98...v0.1.99)
(2023-03-13)

[0.1.98](https://github.com/starknet-io/starknet-docs/compare/v0.1.97...v0.1.98)
(2023-03-13)

[0.1.97](https://github.com/starknet-io/starknet-docs/compare/v0.1.96...v0.1.97)
(2023-03-09)

[0.1.96](https://github.com/starknet-io/starknet-docs/compare/v0.1.95...v0.1.96)
(2023-03-07)

[0.1.95](https://github.com/starknet-io/starknet-docs/compare/v0.1.94...v0.1.95)
(2023-03-06)

[0.1.94](https://github.com/starknet-io/starknet-docs/compare/v0.1.93...v0.1.94)
(2023-03-06)

[0.1.93](https://github.com/starknet-io/starknet-docs/compare/v0.1.92...v0.1.93)
(2023-03-05)

[0.1.92](https://github.com/starknet-io/starknet-docs/compare/v0.1.91...v0.1.92)
(2023-03-02)

[0.1.91](https://github.com/starknet-io/starknet-docs/compare/v0.1.90...v0.1.91)
(2023-03-01)

[0.1.90](https://github.com/starknet-io/starknet-docs/compare/v0.1.89...v0.1.90)
(2023-03-01)

[0.1.89](https://github.com/starknet-io/starknet-docs/compare/v0.1.88...v0.1.89)
(2023-03-01)

[0.1.88](https://github.com/starknet-io/starknet-docs/compare/v0.1.87...v0.1.88)
(2023-02-27)

[0.1.87](https://github.com/starknet-io/starknet-docs/compare/v0.1.86...v0.1.87)
(2023-02-26)

[0.1.86](https://github.com/starknet-io/starknet-docs/compare/v0.1.85...v0.1.86)
(2023-02-24)

[0.1.85](https://github.com/starknet-io/starknet-docs/compare/v0.1.84...v0.1.85)
(2023-02-23)

[0.1.84](https://github.com/starknet-io/starknet-docs/compare/v0.1.83...v0.1.84)
(2023-02-23)

[0.1.83](https://github.com/starknet-io/starknet-docs/compare/v0.1.82...v0.1.83)
(2023-02-22)

[0.1.82](https://github.com/starknet-io/starknet-docs/compare/v0.1.81...v0.1.82)
(2023-02-22)

[0.1.81](https://github.com/starknet-io/starknet-docs/compare/v0.1.80...v0.1.81)
(2023-02-22)

[0.1.80](https://github.com/starknet-io/starknet-docs/compare/v0.1.78...v0.1.80)
(2023-02-22)

[0.1.77](https://github.com/starknet-io/starknet-docs/compare/v0.1.76...v0.1.77)
(2023-02-22)

[0.1.76](https://github.com/starknet-io/starknet-docs/compare/v0.1.75...v0.1.76)
(2023-02-22)

[0.1.75](https://github.com/starknet-io/starknet-docs/compare/v0.1.74...v0.1.75)
(2023-02-21)

[0.1.74](https://github.com/starknet-io/starknet-docs/compare/v0.1.73...v0.1.74)
(2023-02-21)

[0.1.73](https://github.com/starknet-io/starknet-docs/compare/v0.1.72...v0.1.73)
(2023-02-21)

[0.1.72](https://github.com/starknet-io/starknet-docs/compare/v0.1.71...v0.1.72)
(2023-02-21)

[0.1.71](https://github.com/starknet-io/starknet-docs/compare/v0.1.70...v0.1.71)
(2023-02-21)

[0.1.70](https://github.com/starknet-io/starknet-docs/compare/v0.1.69...v0.1.70)
(2023-02-21)

[0.1.69](https://github.com/starknet-io/starknet-docs/compare/v0.1.68...v0.1.69)
(2023-02-16)

[0.1.68](https://github.com/starknet-io/starknet-docs/compare/v0.1.67...v0.1.68)
(2023-02-14)

[0.1.67](https://github.com/starknet-io/starknet-docs/compare/v0.1.66...v0.1.67)
(2023-02-14)

[0.1.66](https://github.com/starknet-io/starknet-docs/compare/v0.1.65...v0.1.66)

(2023-02-08)

[0.1.65](https://github.com/starknet-io/starknet-docs/compare/v0.1.64...v0.1.65)
(2023-01-19)

[0.1.64](https://github.com/starknet-io/starknet-docs/compare/v0.1.63...v0.1.64)
(2023-01-19)

[0.1.63](https://github.com/starknet-io/starknet-docs/compare/v0.1.62...v0.1.63)
(2023-01-19)

[0.1.62](https://github.com/starknet-io/starknet-docs/compare/v0.1.61...v0.1.62)
(2023-01-19)

[0.1.61](https://github.com/starknet-io/starknet-docs/compare/v0.1.60...v0.1.61)
(2023-01-19)

[0.1.60](https://github.com/starknet-io/starknet-docs/compare/v0.1.59...v0.1.60)
(2023-01-17)

[0.1.59](https://github.com/starknet-io/starknet-docs/compare/v0.1.58...v0.1.59)
(2023-01-17)

[0.1.58](https://github.com/starknet-io/starknet-docs/compare/v0.1.57...v0.1.58)
(2023-01-17)

[0.1.57](https://github.com/starknet-io/starknet-docs/compare/v0.1.56...v0.1.57)
(2023-01-16)

[0.1.56](https://github.com/starknet-io/starknet-docs/compare/v0.1.55...v0.1.56)
(2023-01-13)

[0.1.55](https://github.com/starknet-io/starknet-docs/compare/v0.1.54...v0.1.55)
(2023-01-13)

[0.1.54](https://github.com/starknet-io/starknet-docs/compare/v0.1.53...v0.1.54)
(2023-01-12)

[0.1.53](https://github.com/starknet-io/starknet-docs/compare/v0.1.52...v0.1.53)
(2023-01-11)

[0.1.52](https://github.com/starknet-io/starknet-docs/compare/v0.1.51...v0.1.52)
(2023-01-11)

[0.1.51](https://github.com/starknet-io/starknet-docs/compare/v0.1.50...v0.1.51)
(2023-01-10)

[0.1.50](https://github.com/starknet-io/starknet-docs/compare/v0.1.49...v0.1.50)
(2023-01-09)

[0.1.49](https://github.com/starknet-io/starknet-docs/compare/v0.1.48...v0.1.49)
(2023-01-09)

[0.1.48](https://github.com/starknet-io/starknet-docs/compare/v0.1.46...v0.1.48)
(2023-01-09)

[0.1.46](https://github.com/starknet-io/starknet-docs/compare/v0.1.45...v0.1.46)
(2023-01-06)

[0.1.45](https://github.com/starknet-io/starknet-docs/compare/v0.1.44...v0.1.45)
(2023-01-06)

[0.1.44](https://github.com/starknet-io/starknet-docs/compare/v0.1.43...v0.1.44)
(2023-01-06)

[0.1.43](https://github.com/starknet-io/starknet-docs/compare/v0.1.42...v0.1.43)
(2023-01-04)

[0.1.42](https://github.com/starknet-io/starknet-docs/compare/v0.1.41...v0.1.42)
(2023-01-04)

[0.1.41](https://github.com/starknet-io/starknet-docs/compare/v0.1.40...v0.1.41)
(2023-01-04)

[0.1.40](https://github.com/starknet-io/starknet-docs/compare/v0.1.39...v0.1.40)
(2023-01-03)

[0.1.39](https://github.com/starknet-io/starknet-docs/compare/v0.1.38...v0.1.39)
(2023-01-03)

[0.1.38](https://github.com/starknet-io/starknet-docs/compare/v0.1.37...v0.1.38)
(2023-01-03)

[0.1.37](https://github.com/starknet-io/starknet-docs/compare/v0.1.36...v0.1.37)
(2022-12-28)

[0.1.36](https://github.com/starknet-io/starknet-docs/compare/v0.1.35...v0.1.36)
(2022-12-28)

[0.1.35](https://github.com/starknet-io/starknet-docs/compare/v0.1.34...v0.1.35)
(2022-12-28)

[0.1.34](https://github.com/starknet-io/starknet-docs/compare/v0.1.33...v0.1.34)
(2022-12-23)

[0.1.33](https://github.com/starknet-io/starknet-docs/compare/v0.1.32...v0.1.33)
(2022-12-23)

[0.1.32](https://github.com/starknet-io/starknet-docs/compare/v0.1.31...v0.1.32)
(2022-12-23)

[0.1.31](https://github.com/starknet-io/starknet-docs/compare/v0.1.30...v0.1.31)
(2022-12-23)

[0.1.30](https://github.com/starknet-io/starknet-docs/compare/v0.1.29...v0.1.30)
(2022-12-21)

[0.1.29](https://github.com/starknet-io/starknet-docs/compare/v0.1.28...v0.1.29)
(2022-12-14)

[0.1.28](https://github.com/starknet-io/starknet-docs/compare/v0.1.27...v0.1.28)
(2022-12-14)

[0.1.27](https://github.com/starknet-io/starknet-docs/compare/v0.1.26...v0.1.27)
(2022-12-13)

[0.1.26](https://github.com/starknet-io/starknet-docs/compare/v0.1.25...v0.1.26)
(2022-12-06)

[0.1.25](https://github.com/starknet-io/starknet-docs/compare/v0.1.24...v0.1.25)
(2022-12-06)

[0.1.24](https://github.com/starknet-io/starknet-docs/compare/v0.1.23...v0.1.24)
(2022-12-06)

[0.1.23](https://github.com/starknet-io/starknet-docs/compare/v0.1.22...v0.1.23)
(2022-12-06)

[0.1.22](https://github.com/starknet-io/starknet-docs/compare/v0.1.21...v0.1.22)
(2022-12-06)

[0.1.21](https://github.com/starknet-io/starknet-docs/compare/v0.1.20...v0.1.21)
(2022-12-06)

[0.1.20](https://github.com/starknet-io/starknet-docs/compare/v0.1.19...v0.1.20)
(2022-12-06)

[0.1.19](https://github.com/starknet-io/starknet-docs/compare/v0.1.18...v0.1.19)

(2022-12-06)

[0.1.18](https://github.com/starknet-io/starknet-docs/compare/v0.1.17...v0.1.18)
(2022-12-06)

[0.1.17](https://github.com/starknet-io/starknet-docs/compare/v0.1.16...v0.1.17)
(2022-12-06)

[0.1.16](https://github.com/starknet-io/starknet-docs/compare/v0.1.15...v0.1.16)
(2022-12-06)

[0.1.15](https://github.com/starknet-io/starknet-docs/compare/v0.1.14...v0.1.15)
(2022-11-29)

[0.1.14](https://github.com/starknet-io/starknet-docs/compare/v0.1.13...v0.1.14)
(2022-11-23)

[0.1.13](https://github.com/starknet-io/starknet-docs/compare/v0.1.12...v0.1.13)
(2022-11-22)

[0.1.12](https://github.com/starknet-io/starknet-docs/compare/v0.1.11...v0.1.12)
(2022-11-18)

[0.1.11](https://github.com/starknet-io/starknet-docs/compare/v0.1.10...v0.1.11)
(2022-11-18)

[0.1.10](https://github.com/starknet-io/starknet-docs/compare/v0.1.9...v0.1.10)
(2022-11-17)

[0.1.9](https://github.com/starknet-io/starknet-docs/compare/v0.1.8...v0.1.9)
(2022-11-17)

[0.1.8](https://github.com/starknet-io/starknet-docs/compare/v0.1.7...v0.1.8)
(2022-11-17)

[0.1.7](https://github.com/starknet-io/starknet-docs/compare/v0.1.4...v0.1.7)
(2022-11-16)

[0.1.6](https://github.com/starknet-community-libs/starknet-docs/compare/v0.1.4...v0.1.6) (2022-11-07)

Bug Fixes

- Fixed a few typos and Contract Hash -> Class Hash ([#89](https://github.com/starknet-community-libs/starknet-docs/issues/89)) ([c266e63](https://github.com/starknet-community-libs/starknet-docs/commit/c266e63f68e06b14cbfd267f22f53fd44ce8bf9f))

[0.1.5](https://github.com/starknet-community-libs/starknet-docs/compare/v0.1.4...v0.1.5) (2022-10-27)

[0.1.4](https://github.com/starknet-community-libs/starknet-docs/compare/v0.1.3-5...v0.1.4) (2022-10-27)

Bug Fixes

- Steve/test deploy ([#93](https://github.com/starknet-community-libs/starknet-docs/issues/93)) ([8548725](https://github.com/starknet-community-libs/starknet-docs/commit/8548725b53dbd0674511477cc8db19d3ff6ac50d))

[0.1.3](https://github.com/starknet-community-libs/starknet-docs/compare/v0.1.3-5...v0.1.3) (2022-10-27)

[0.1.2](https://github.com/starknet-community-libs/starknet-docs/compare/v0.1.1...v0.1.2) (2022-10-19)

[0.1.1-2](https://github.com/starknet-community-libs/starknet-docs/compare/v0.1.1-1...v0.1.1-2) (2022-10-18)

[0.1.1](https://github.com/starknet-community-libs/starknet-docs/compare/v0.1.1-1...v0.1.1) (2022-10-18)

[0.0.22](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.18-13...v0.0.22) (2022-09-04)

[0.0.20](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.18-7...v0.0.20) (2022-06-15)

[0.0.18](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.18-0...v0.0.18) (2022-06-06)

[0.0.16](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-18...v0.0.16) (2022-05-18)

[0.0.14](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-17...v0.0.14) (2022-05-11)

[0.0.12](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-14...v0.0.12) (2022-05-01)

[0.0.10](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-9...v0.0.10) (2022-04-06)

[0.0.8](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.6...v0.0.8) (2022-03-31)

[0.0.6](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-8...v0.0.6) (2022-03-31)

[0.0.4](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-0...v0.0.4) (2022-02-22)

[0.0.22](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.18-13...v0.0.22) (2022-09-04)

[0.0.20](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.18-7...v0.0.20) (2022-06-15)

[0.0.18](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.18-0...v0.0.18) (2022-06-06)

[0.0.16](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-18...v0.0.16) (2022-05-18)

[0.0.14](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-17...v0.0.14) (2022-05-11)

[0.0.12](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-14...v0.0.12) (2022-05-01)

[0.0.10](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-9...v0.0.10) (2022-04-06)

[0.0.8](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.6...v0.0.8) (2022-03-31)

[0.0.6](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-8...v0.0.6) (2022-03-31)

[0.0.4](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-0...v0.0.4) (2022-02-22)

[0.0.20](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.18-7...v0.0.20) (2022-06-15)

[0.0.18](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.18-0...v0.0.18) (2022-06-06)

[0.0.16](https://github.com/starknet-community-libs/starknet-docs/compare/

v0.0.4-18...v0.0.16) (2022-05-18)

[0.0.14](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-17...v0.0.14) (2022-05-11)

[0.0.12](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-14...v0.0.12) (2022-05-01)

[0.0.10](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-9...v0.0.10) (2022-04-06)

[0.0.8](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.6...v0.0.8) (2022-03-31)

[0.0.6](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-8...v0.0.6) (2022-03-31)

[0.0.4](https://github.com/starknet-community-libs/starknet-docs/compare/v0.0.4-0...v0.0.4) (2022-02-22)

[0.0.18](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-24...v0.0.18) (2022-06-06)

[0.0.16](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-18...v0.0.16) (2022-05-18)

[0.0.14](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-17...v0.0.14) (2022-05-11)

[0.0.12](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-14...v0.0.12) (2022-05-01)

[0.0.10](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-9...v0.0.10) (2022-04-06)

[0.0.8](https://github.com/starkware-libs/starknet-docs/compare/v0.0.6...v0.0.8) (2022-03-31)

[0.0.6](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-8...v0.0.6) (2022-03-31)

[0.0.4](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-0...v0.0.4) (2022-02-22)

[0.0.16](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-18...v0.0.16) (2022-05-18)

[0.0.14](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-17...v0.0.14) (2022-05-11)

[0.0.12](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-14...v0.0.12) (2022-05-01)

[0.0.10](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-9...v0.0.10) (2022-04-06)

[0.0.8](https://github.com/starkware-libs/starknet-docs/compare/v0.0.6...v0.0.8) (2022-03-31)

[0.0.6](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-8...v0.0.6) (2022-03-31)

[0.0.4](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-0...v0.0.4) (2022-02-22)

[0.0.14](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-16...v0.0.14) (2022-05-11)

[0.0.12](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-14...v0.0.12) (2022-05-01)

[0.0.10](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-9...v0.0.10) (2022-04-06)

[0.0.8](https://github.com/starkware-libs/starknet-docs/compare/v0.0.6...v0.0.8) (2022-03-31)

[0.0.6](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-8...v0.0.6) (2022-03-31)

[0.0.4](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-0...v0.0.4) (2022-02-22)

[0.0.12](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-13...v0.0.12) (2022-05-01)

[0.0.10](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-9...v0.0.10) (2022-04-06)

[0.0.8](https://github.com/starkware-libs/starknet-docs/compare/v0.0.6...v0.0.8) (2022-03-31)

[0.0.6](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-8...v0.0.6)
(2022-03-31)

[0.0.4](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-0...v0.0.4)
(2022-02-22)

[0.0.10](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-9...v0.0.10)
(2022-04-06)

[0.0.8](https://github.com/starkware-libs/starknet-docs/compare/v0.0.6...v0.0.8)
(2022-03-31)

[0.0.6](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-8...v0.0.6)
(2022-03-31)

[0.0.4](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-0...v0.0.4)
(2022-02-22)

[0.0.8](https://github.com/starkware-libs/starknet-docs/compare/v0.0.6...v0.0.8)
(2022-03-31)

[0.0.4-8](https://github.com/starkware-libs/starknet-docs/compare/
v0.0.4-7...v0.0.4-8) (2022-03-31)

[0.0.6](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-7...v0.0.6)
(2022-03-31)

[0.0.4](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-0...v0.0.4)
(2022-02-22)

[0.0.4](https://github.com/starkware-libs/starknet-docs/compare/v0.0.4-0...v0.0.4)
(2022-02-22)

```
pipeline {
  agent { label "gcp-hodor-slave-generic" }
  options {
    // Add timestamps to output.
    timestamps()
    timeout(time: 10, unit: 'MINUTES')
  }
  environment {
    repoOwner = "starknet-io"
    repoName = "starknet-docs"
    prId = "${env.CHANGE_ID}"
    prPrefix = "pr"
    comment = "Your preview build is ready! '( Check the following link in 1-2 minutes:  
https://${repoOwner}.github.io/${repoName}/${prPrefix}-${prId}/ ."
  }
}
```



```

    commitMessage = "Adding or updating preview build for PR: ${prId}"
    githubCredentials = credentials('gh-starknet-docs-pat')
}
stages {
    stage('Typos Tests') {
        agent {
            docker {
                image 'rust:latest'
                reuseNode true
                args '--user root'
            }
        }
        steps {
            script {
                sh label:"Install typos-cli", script:
                """
                    cargo install typos-cli
                """
                sh label:"Run typos test", script:
                """
                    set +e

                    res=$(typos)

                    if [ \ $? != 0 ]
                    then
                        apt update
                        apt install -y python3-requests
                        python3 CI/scripts/comment_pr.py -a "$githubCredentials_PSW" -o
"$repoOwner" -r "$repoName" -p "$prId" -c "Oops, your pull request failed to pass the
Typo tests stage :( . see the result: \n$res\n\nPlease fix the typo, commit and push!"
                        exit 1
                    fi
                """
            }
        }
    }
    stage('Build') {
        agent {
            docker {
                image 'node:16.20.2'
                reuseNode true
            }
        }
        environment {
            ANTORA_CACHE_DIR="${WORKSPACE}/.cache"

```

```

    }
    steps {
      script {
        sh label:"Install node modules", script:
        """
          yarn
        """
        sh label:"Generate site", script:
        """
          yarn generate
        """
        sh label:"Copy artifacts", script:
        """
          mkdir ${prPrefix}-${prId}
          cp -r ./public_html/* ${prPrefix}-${prId}
        """
        stash includes: "${prPrefix}-${prId}/**/*", name: "artifact"
      }
    }
  }
}
stage('Publish') {
  steps {
    script {
      sh label:"Git clone", script:
      """
        git clone https://$githubCredentials_PSW@github.com/starknet-io/
starknet-docs.git
      """
      sh label:"Git config", script:
      """
        git config --global user.name "Starknet Bot"
        git config --global user.email "starknet-bot@noreply.com"
      """
      dir('starknet-docs') {
        sh label:"Checkout gh-pages", script:
        """
          git checkout gh-pages
        """
        sh label:"Remove PR folder if exist", script:
        """
          if [ -d ${prPrefix}-${prId} ]
          then
            rm -rf ${prPrefix}-${prId}
          fi
        """
        unstash 'artifact'
      }
    }
  }
}

```

```
def result = sh (
    label:"Push to branch gh-pages",
    returnStatus: true,
    script:
        ""
        git add ${prPrefix}-${prId}
        if ! git diff-index --quiet HEAD --; then
            echo "Changes detected. Committing and pushing changes."
            echo "Commit message: $commitMessage"
            git commit -m "$commitMessage"
            echo "Commit successful. Pushing changes."
            git pull
            git push
            exit 0
        else
            echo "No changes found."
            exit 10
        fi
    ""
)
if (result == 10) {
    echo 'Pipeline succeeded because there were no changes to push.'
    currentBuild.result = 'SUCCESS'
    return
} else if (result != 0) {
    echo 'Push to branch gh-pages step failed. Marking pipeline as
FAILURE.'

    currentBuild.result = 'FAILURE'
    return
}
sh label:"Checkout ${env.CHANGE_TARGET}", script:
""
git checkout "${env.CHANGE_TARGET}"
""
sh label:"Comment preview link on contributor pull request", script:
""
echo "Commenting preview link on PR: #${prId}"
python3 CI/scripts/comment_pr.py -a "$githubCredentials_PSW" -o
"$repoOwner" -r "$repoName" -p "${prId}" -c "$comment"
""
}
}
}
```

```

    post {
        cleanup {
            deleteDir()
        }
    }
}
import requests
import argparse

```

```

def comment_pr(api_url, auth_token, owner, repo, pr_id, comment):
    url = f"{api_url}/repos/{owner}/{repo}/issues/{pr_id}/comments"

```

```

    headers = {
        'Authorization': f'Bearer {auth_token}',
        'Content-Type': 'application/json',
        'Accept': 'application/vnd.github.v3+json',
        'X-GitHub-API-Version': '2022-11-28'
    }

```

```

    payload = {
        "body": comment
    }

```

```

    try:
        response = requests.post(url=url, headers=headers, json=payload)
        if response.status_code in (200, 201):
            print('Comment posted successfully!')
            print('Response:', response.json())
        else:
            print('Request failed with status code:', response.status_code)
    except Exception as e:
        print(e)

```

```

def main():
    parser = argparse.ArgumentParser(description="Create a comment on Github pull request.")
    parser.add_argument("-u", "--api_url", required=False, default="https://api.github.com", help="Github api url.")
    parser.add_argument("-a", "--auth_token", required=True, help="The Github auth token. Used to authenticate with Github.")
    parser.add_argument("-o", "--owner", required=True, help="The repo owner")
    parser.add_argument("-r", "--repo", required=True, help="The name of the repo")
    parser.add_argument("-p", "--pr", required=True, help="The pull request id/number")
    parser.add_argument("-c", "--comment", required=True, help="The message to

```

```

comment")
    args = parser.parse_args()

    comment_pr(args.api_url, args.auth_token, args.owner, args.repo, args.pr,
args.comment)

    exit(0)

if __name__ == "__main__":
    main()
CI/*    @idan-stark
[id="readme"]
+++++
<div align="center">
+++++
// For info on these shields see https://shields.io/badges/
image:https://img.shields.io/badge/PRs-welcome-ff69b4.svg?style=flat-square[https://
github.com/starknet-io/starknet-docs/issues]
image:https://img.shields.io/badge/Read_the_Starknet_docs!-2f6df2[Static Badge,
link=https://docs.starknet.io]
image:https://img.shields.io/badge/Telegram-2AABEE[Static Badge,link=https://t.me/
+9VC94EQA8dY4ZTZh]
+++++
</div>
+++++

= Starknet documentation repository
:toc:
:toclevels: 1

The Starknet docs website, https://docs.starknet.io, is written in https://asciidoc.org/
[AsciiDoc] and is built using link:https://antora.org/[Antora], a static website generator
for AsciiDoc.

== Contributing to Starknet documentation
If you are interested in contributing to Starknet technical documentation, the following
table provides quick links to help you get started.

|===

|*Question* |*Resource*

|I'm interested, how do I contribute?
|For information on how you can contribute, see
xref:#different_ways_to_contribute[Different ways to contribute].

```

|Are there any basic guidelines to help me?

| For basic guidelines to help us keep our content consistent, see [link:/contributing_to_docs/doc_guidelines.adoc](#)[Documentation guidelines].

| Is there a style guide and writing guide I should use?

| See the [xref:contributing_to_docs/starknet_docs_style_guide.adoc](#)[Starknet documentation supplementary style guide].

|How do I set up my workstation?

|See [xref:contributing_to_docs/setting_up_environment.adoc](#)[Setting up your environment].

|===

[#different_ways_to_contribute]

== Different ways to contribute

There are a few different ways you can contribute to Starknet documentation:

- * Create a [link:https://github.com/starknet-io/starknet-docs/issues](https://github.com/starknet-io/starknet-docs/issues)[GitHub issue].

- * Submit a pull request (PR). You can create a local clone of your own fork of the [link:https://github.com/starknet-io/starknet-docs](https://github.com/starknet-io/starknet-docs)[starknet-docs repository], make your changes, and submit a PR. This option is best if you have substantial changes, or to help the changes you want to be added more quickly.

What happens when you submit a PR?

When you submit a PR, the <https://github.com/orgs/starknet-io/teams/starknet-docs>[Starknet Docs team] reviews the PR and arranges further technical reviews as necessary. If the PR requires changes, the reviewers add comments to the PR. We might request that you make the changes, or let you know that we incorporated your content in a different PR. Occasionally, we might add commits to the original PR directly. When the PR has been reviewed and all updates are complete, the documentation team merges the PR and applies it to the valid version(s).

== Tips on authoring

- * [xref:contributing_to_docs/starknet_docs_style_guide.adoc](#)[Starknet documentation supplementary style guide]: General style guidance and writing guidance.

For information on writing in AsciiDoc, see:

- * [link:https://docs.asciidoctor.org/asciidoc/latest/](https://docs.asciidoctor.org/asciidoc/latest/)[AsciiDoc Language Documentation]

- * [link:http://asciidoctor.org/docs/asciidoc-syntax-quick-reference/](http://asciidoctor.org/docs/asciidoc-syntax-quick-reference/)[AsciiDoc Syntax Quick

Reference]

[NOTE]

====

There are multiple ways of coding ids, source code blocks, cross-references, and links. In general we use the most explicit coding conventions for coding in order to prioritize code readability. Most of these coding conventions are listed in [link:https://redhat-documentation.github.io/asciidoc-markup-conventions/](https://redhat-documentation.github.io/asciidoc-markup-conventions/) [AsciiDoc Mark-up Quick Reference for Red Hat Documentation]

====

== Before you begin

- . Install ``yarn`` if it's not already installed.
- . Install ``npm`` if it's not already installed.
- . Clone this repo, either from a fork, or if you are an official collaborator, then directly from ``starknet-io/starknet-docs``.
- . Change to the ``starknet-docs`` directory.
- . Run the ``yarn`` command to prepare the environment:

+

yarn

``yarn`` should prepare your environment by installing the required modules based on ``package-lock.json`` and ``package.json``. If it was successful, you should be able to build and preview content.

== Building and previewing content locally

After writing or editing content, to preview your changes:

- . Build the content by running the ``build_local_site.sh`` build script:

+

[source,bash]

./build_local_site.sh

+

This command generates the website in the directory ``public_html``.

- . Open the website by doing one of the following:

+

* Open the start page: ``<repo_root>/public_html/index.html``.

* Run the `xref:http_server[http server]` packaged with Antora:

+

[source,bash]

```
npx http-server public_html -c-1
```

+

The server runs, and gives you one or more local URLs that you can use to view the website. For example:

+

```
[source,bash,subs="+quotes,+macros"]
```

Starting up http-server, serving *public_html*

...

Available on:

\http://127.0.0.1:8080

\http://192.168.68.111:8080

\http://192.168.14.3:8080

\http://10.14.0.2:8080

Hit CTRL-C to stop the server

== Releasing Starknet docs on GitHub (for Admins only)

The high-level process for releasing documentation changes in this repository.

During the course of content development, writers merge branches with changes either directly into `main`, into a secondary branch as needed, where these changes wait until we are ready to release them—that is, post them to link:<https://docs.starknet.io>[docs.starknet.io].

GitHub actions create Git tags and releases that appear at the repo's link:<https://github.com/starknet-community-libs/starknet-docs/releases>[Releases] and link:<https://github.com/starknet-community-libs/starknet-docs/tags>[Tags] pages.

When a feature branch is merged into the `main` branch, a GitHub action creates a release tag in the format `v<version>.<major_update>.<minor_update>` and updates `CHANGELOG.md`. It then publishes the new content to docs.starknet.io.

=== Procedure

Merging a feature branch to `main` automatically publishes changes in the feature branch. No additional steps are required.

GitHub increments the version numbers in `package.json` and `package-lock.json`, and updates `CHANGELOG.md` with the descriptions of each PR that was just merged into

`main`.

. Update your local `main` branch from the remote `main` branch using one of the following:

* Pull the changes:

+

[source,bash]

starknet-docs (main) git pull

* Do a rebase from `git@github.com:starknet-io/starknet-docs.git`:

+

[source,bash]

starknet-docs (main) git fetch origin

starknet-docs (main) git rebase origin/main

+

[NOTE]

=====

If you are using a fork, then your forked repo is `origin` by default, in which case you should assign the name `upstream` to `git@github.com:starknet-io/starknet-docs.git`. So when rebasing, use `upstream` instead of `origin` in the above commands.

=====

[default]

check-file = false

[type.adoc]

extend-glob = ["*.adoc"]

check-file = true

[default.extend-words]

Don't correct the word

afe = "afe"

npx antora --stacktrace --extension ./linked-worktree-as-content-source.js playbook.yml

About Starknet

* xref:index.adoc[]

// * xref:notational-conventions.adoc[]

* xref:glossary.adoc[]include::1.0.0@docs-common-content:ROOT:partial\$partial_glossary.adoc[]

// The source for the included file is in a separate repository.

//

// To edit the source page directly, go to:

```
//  
// `https://github.com/starknet-io/docs-common-content/edit/main/modules/ROOT/  
partials/partial_glossary.adoc`[id="overview"]  
= Overview
```

Starknet is a permissionless Validity-Rollup, also known as a zero-knowledge rollup (ZK rollup) for Ethereum. As a Layer 2 (L2) blockchain, Starknet enables any dApp to achieve massive computation scale without compromising on Ethereum's composability and security.

Starknet aims to achieve secure, low-cost transactions and high performance by using the STARK cryptographic proof system. Starknet contracts and the Starknet OS are written in link:<https://github.com/starkware-libs/cairo>[Cairo], a custom-built and specialized programming language.

== Explore Starknet

```
[pass]  
++++  
<div class="home-cta-container">  
  <a href="https://docs.starknet.io/documentation/quick_start/environment_setup/"  
class="home-cta home-cta-first" id="cta1">  
    <div class="image-container">  
        
    </div>  
    <h2>Quick start</h2>  
    <p class="chakra-card__body css-jintet" id="text1">Set up your environment and get  
started with Starknet.</p>  
    <p class="chakra-card__body css-jintet" id="text2"><b>Explore ></b></p>  
  </a>  
  
  <a href="https://docs.starknet.io/documentation/architecture_and_concepts/  
Network_Architecture/header/" class="home-cta" id="cta2">  
    <div class="image-container">  
        
    </div>  
    <h2>Architecture</h2>  
    <p class="chakra-card__body css-jintet" id="text3">Learn about Starknet's  
architecture including contracts and accounts.</p>  
    <p class="chakra-card__body css-jintet" id="text4"><b>Explore ></b></p>  
  </a>  
  
  <a href="https://docs.starknet.io/documentation/starknet_versions/version_notes/"  
class="home-cta" id="cta3">
```

```

<div class="image-container">
  
</div>
<h2>Releases</h2>
<p class="chakra-card__body css-jintet" id="text5">Learn more about the current
version of Starknet.</p>
<p class="chakra-card__body css-jintet" id="text6"><b>Explore ></b></p>
</a>
</div>
++++

```

== Developer tools and resources

[pass]

```

++++
<div class="no-background no-border">
  <div class="column-container">
    <div class="column">
      <p><a href="https://book.starknet.io/">The Starknet Book</a></p>
      <p><a href="https://docs.cairo-lang.org/">The Cairo docs</a></p>
      <p><a href="https://docs.starknet.io/documentation/tools/devtools/">Starknet
development tools</a></p>
      <p><a href="https://docs.starknet.io/documentation/tools/api-services/">Full nodes
and API services</a></p>
    </div>
    <div class="column">
      <p><a href="https://docs.starknet.io/documentation/tools/
limits_and_triggers/">Limits and triggers</a></p>
      <p><a href="https://docs.starknet.io/documentation/cli/starkli/">Cairo 0 tools</a></
p>
      <p><a href="https://docs.starknet.io/documentation/tools/
ref_block_explorers/">Block explorers</a></p>
      <p><a href="https://docs.starknet.io/documentation/tools/audit/">Audit providers</
a></p>
    </div>
  </div>
</div>
++++

```

== Contribute to the Starknet docs

Want to contribute to Starknet documentation? Get started here:

* See the link:https://github.com/starknet-io/starknet-docs/blob/dev/README.adoc#different_ways_to_contribute[different ways to contribute].

* Review these link:https://github.com/starknet-io/starknet-docs/blob/dev/contributing_to_docs/doc_guidelines.adoc[basic writing guidelines] to help you write better and be a more valuable contributor.

* Use the link:https://github.com/starknet-io/starknet-docs/blob/dev/contributing_to_docs/starknet_docs_style_guide.adoc[Starknet documentation supplementary style guide] to help you write with the Starknet voice.

```
[pass]
++++
<html>
<head>
<style>
*::before, ::after {
  border-color: var(--chakra-colors-gray-200);
}

:where(*, *::before, *::after) {
  border-width: 0;
  border-style: solid;
  box-sizing: border-box;
  word-wrap: break-word;
}

@media (max-width: 768px) {
  .home-cta-container {
    flex-direction: column; /* Switch back to a column layout for mobile */
  }

  .home-cta-container .home-cta {
    margin: 8px 8px 0 8px; /* Reset margin for mobile */
    width: 100%; /* Make each box take up the full width of the screen */
  }
}

.home-cta-container {
  display: flex;
}

.cta-image-container {
  background-image: url('_images/developers.svg');
  background-size: cover;
  background-repeat: no-repeat;
  background-position: center center;
  width: 100%;
  height: 100%;
}
```

```
.image-container {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  height: 8em;
  margin-bottom: 10px;
  margin: 8px;
  border-radius: 20px 20px 0 0;
  position: relative;
  background-image: linear-gradient(180.15deg, var(--chakra-colors-chakra-body-text)
0.2%, var(--chakra-colors-chakra-body-bg) 105.43%);
  overflow: hidden;
}
```

```
.image-container img {
  z-index: -1;
}
```

```
.cta-image {
  max-width: 464px;
  max-height: 100%;
}
```

```
.home-cta {
  flex: 1;
  margin: 8px 8px 0 8px;
  padding: 20px;
  background-color: var(--chakra-colors-chakra-body-bg);
  border: 1px solid rgb(226, 232, 240);
  border-bottom: 1px solid rgb(226, 232, 240);
  border-radius: 20px;
  color: var(--chakra-colors-card-link-fg);
  font-size: 18px;
  font-weight: var(--chakra-fontWeights-medium);
  text-decoration: none;
  transition: background-color 0.15s, border-color 0.15s, color 0.15s;
  box-sizing: border-box;
  position: relative;
  background-image: linear-gradient(180.15deg, var(--chakra-colors-gradient-blue-
default-a) 0.2%, var(--chakra-colors-gradient-blue-default-b) 105.43%);
  overflow: hidden;
  transition-property: var(--chakra-transition-property-common);
  transition-duration: var(--chakra-transition-duration-fast);
  transition-timing-function: var(--chakra-transition-easing-ease-out);
}
```

```
    cursor: pointer;
    -webkit-text-decoration: none;

    outline: 2px solid transparent;
    outline-offset: 2px;
}

.home-cta-first {
    margin-left: 0;
}

.chakra-card__body.css-jintet {
    font-size: 15px;
    color: #363636;
}

.column-container {
    display: flex;
}

.column {
    flex: 1;
    padding: 10px;
    margin: 5px;
    border-radius: 5px;
}

.home-cta-container .home-cta:hover {
    text-decoration: none;
    color: #363636;
    border-color: #C506E4;
}

.home-cta a {
    text-decoration: none;
    color: #363636;
}
</style>

<script>

document.addEventListener('DOMContentLoaded', function() {
    const themeSwitch = document.querySelector('[data-theme="dark"]');
    if (!themeSwitch) {
        console.error('Theme switch element not found');
        return;
    }
});
```

```
}
```

```
const image1 = document.getElementById('img_1');  
const image2 = document.getElementById('img_2');  
const image3 = document.getElementById('img_3');
```

```
const text1 = document.getElementById('text1');  
const text2 = document.getElementById('text2');  
const text3 = document.getElementById('text3');  
const text4 = document.getElementById('text4');  
const text5 = document.getElementById('text5');  
const text6 = document.getElementById('text6');
```

```
let initialThemeSet = false; // To track if the initial theme has been set
```

```
themeSwitch.addEventListener('click', () => {  
  const currentTheme = themeSwitch.getAttribute('data-theme');  
  // Toggle the theme  
  themeSwitch.setAttribute('data-theme', currentTheme === 'dark' ? 'light' : 'dark');  
  // Toggle the theme  
  toggleTheme(currentTheme);  
});
```

```
function toggleTheme(currentTheme) {  
  // Handle theme switching and image/text updates  
  if (currentTheme === 'dark' || !initialThemeSet) {  
    image1.src = '_images/developers_dark.svg';  
    image2.src = '_images/how_SN_works_dark.svg';  
    image3.src = '_images/roadmap_dark.svg';  
    text1.style.color = 'white';  
    text2.style.color = 'white';  
    text3.style.color = 'white';  
    text4.style.color = 'white';  
    text5.style.color = 'white';  
    text6.style.color = 'white';  
  } else {  
    image1.src = '_images/developers.svg';  
    image2.src = '_images/how_SN_works.svg';  
    image3.src = '_images/roadmap.svg';  
    text1.style.color = '';  
    text2.style.color = '';  
    text3.style.color = '';  
    text4.style.color = '';  
    text5.style.color = '';  
    text6.style.color = '';  
  }  
}
```

```

    if (!initialThemeSet) {
        initialThemeSet = true;
    }
}

// Set the initial state based on the themeSwitch value
toggleTheme(themeSwitch.getAttribute('data-theme'));
});

```

```

</script>
</head>
</html>

```

```

++++
[id="notational-conventions"]
= Notational conventions

```

```

[cols="1,2", stripes=even]
|===
| Notation | Explanation

```

| ***bold* a** | * Indicates GUI elements, such as button text or menu names.
 * Highlights text for added importance or to draw attention to it.

| *_italics_*
 | Indicates the first occurrence of a new term, titles of documents or topics, and user-supplied or variable values.

| ``monospace`` | Indicates code, commands, file paths, or other text that should be displayed in a fixed-width font to differentiate it from the surrounding text.

| ``<item>``
 | Indicates user-supplied or variable values in code, commands, file paths, or other text that should be displayed in a monospace font.

Can also refer to generic types. For example: ``List<felt252>``

| $\text{stem:}[\text{text}\{\text{string_in_math_notation}\}]$
 | A normal (non-italics) serif typeface indicates a fixed element or constant specified in mathematical notation.

| ...
 | An ellipsis indicates that the preceding element can repeat multiple times.

| [item, ...]

a| Square brackets indicate that the enclosed items are optional. Also can indicate a range of numbers, where the enclosed items are included. For example, in [0,100], 0 and 100 are part of the specified range.

a| (item1, item2)

| Parentheses indicate a range of numbers, where the enclosed items are not included. For example, in (0,100), 0 and 100 are not part of the specified range.

| item1 \| item2

| A vertical bar (\|) indicates a choice between *item1* and *item2*.

a| +{item1, item2}+

| Curly brackets indicates a list or set of possible values.

| /

| A forward slash indicates a division operator or a path separator in URLs and file paths.

|===

[NOTE]

====

This topic previously appeared in the Starknet Book.

====[IMPORTANT]

====

Sepolia testnet replaces Goerli testnet.

Goerli testnet support is now removed.

For more information, including bridge support for Sepolia, see link:<http://eepurl.com/iK0YTE>[Starknet Goerli Deprecation] in the Starknet Dev News newsletter.

====

* Architecture

** xref:network-architecture/starknet-architecture-overview.adoc[Overview]

** xref:provers-overview.adoc[Provers]

** xref:solidity-verifier.adoc[Solidity verifier]

** xref:nodes.adoc[Nodes]

** xref:network-architecture/block-structure.adoc[Block structure]

** xref:network-architecture/transaction-life-cycle.adoc[Transaction lifecycle]

** xref:network-architecture/transactions.adoc[Transaction types]

** xref:network-architecture/fee-mechanism.adoc[]

** State

```

*** xref:network-architecture/starknet-state.adoc[Starknet state]
*** xref:network-architecture/data-availability.adoc[Data availability]

** L1-L2 messaging
*** xref:network-architecture/messaging-mechanism.adoc[L1-L2 messaging
mechanism]
*** xref:network-architecture/messaging-reference.adoc[Messaging reference]

** Accounts
*** xref:accounts/introduction.adoc[What is an account?]
*** xref:accounts/approach.adoc[Starknet's account interface]
*** xref:accounts/account-functions.adoc[Account interface function reference]
*** xref:accounts/deploying-new-accounts.adoc[Deploying new accounts]
*** xref:accounts/universal-deployer.adoc[Universal Deployer Contract]
*** xref:accounts/simplified-transaction-flow.adoc[Simplified transaction flow]

** Contracts
*** xref:smart-contracts/contract-classes.adoc[Contract classes and instances]
*** xref:smart-contracts/class-hash.adoc[Class hash]
*** xref:smart-contracts/compiled-class-hash.adoc[Compiled class hash]
*** xref:smart-contracts/contract-address.adoc[Contract address]
*** xref:smart-contracts/contract-storage.adoc[Contract storage]
*** xref:smart-contracts/contract-abi.adoc[Contract ABI]
*** xref:smart-contracts/starknet-events.adoc[Events]
*** xref:smart-contracts/contract-syntax.adoc[Migrating a contract from Cairo v1 to
Cairo v2]
*** xref:smart-contracts/cairo-and-sierra.adoc[Cairo and Sierra]
*** xref:smart-contracts/cairo-builtins.adoc[Cairo builtins]
*** xref:smart-contracts/serialization-of-cairo-types.adoc[Serialization of Cairo types]
*** xref:smart-contracts/system-calls-cairo1.adoc[System calls]
*** xref:smart-contracts/execution-info.adoc[Execution information for the current block]

** Cryptography
*** xref:cryptography/p-value.adoc[The STARK field]
*** xref:cryptography/stark-curve.adoc[The STARK curve]
*** xref:cryptography/hash-functions.adoc[Hash functions]

* xref:economics-of-starknet.adoc[The Economics of Starknet]
[id="account_interface_functions"]
= Account interface function reference

```

== Overview

The functions in the table `xref:#starknet_account_interface_functions[]` are part of account contracts. Where required, you must include these functions within your account contract. The logic of these functions can be mostly arbitrary, with a few

limitations. For information on these limitations, see `xref:#limitations_of_validation[]`.

[#starknet_account_interface_functions]

.Starknet account interface functions

[cols="1,3a"]

====

| Function name | When required

| ``+__validate__+`` | Always required

| ``+__execute__+`` | Always required. The signatures of ``+__validate__+`` and ``+__execute__+`` must be identical.

[WARNING]

====

At the moment of writing (Starknet 0.13.2), two critical validations must happen in ``+__execute__+``, and their absence can lead to draining of the account's funds:

(1) ``assert!(get_caller_address().is_zero())``

This asserts that the account's ``+__execute__+`` is not called from another contract, thus skipping validations (in later versions we may disallow calling ``__execute__`` from another contract at the protocol level)

(2) ``assert!(get_tx_info().unbox().version.into() >= 1_u32)``

This asserts that the transaction's version is at least 1, preventing the account from accepting ``INVOKE`` v0 transactions. It is critical to explicitly disallow the deprecated v0 transaction type, as v0 transactions assume that the signature verification happens in ``+__execute__+``, and are thus skipping ``+__validate__+`` entirely.

====

| ``+__validate_declare__+`` | Required for the account to be able to send a ``DECLARE`` transaction. This function must receive exactly one argument, which is the class hash of the declared class.

| ``+__validate_deploy__+`` a | Required to allow deploying an instance of the account contract with a ``DEPLOY_ACCOUNT`` `_transaction_`. The arguments of ``+__validate_deploy__+`` must be the class hash of the account to be deployed, the salt used for computing the account's contract address, followed by the constructor arguments.

[NOTE]

====

You can only use the ``+__validate_deploy__+`` function in an account contract to validate the ``DEPLOY_ACCOUNT`` transaction for that same contract. That is, this function runs at most once throughout the lifecycle of the account.

====

| ``constructor`` | All contracts have a ``constructor`` function. It can be explicitly defined in

the contract, or if not explicitly defined, the sequencer uses a default ``constructor`` function, which is empty.

|===

When the sequencer receives a transaction, it calls the corresponding validation function with the appropriate input from the transaction's data, as follows:

- * For an ``INVOKE`` transaction, the sequencer calls the ``+__validate__+`` function with the transaction's calldata as input. The transaction's calldata will be deserialized to the arguments in the ``+__validate__+`` function's signature, it is up to the sender to make sure that the calldata is encoded appropriately according to validate's signature. After successfully completing validation, the sequencer calls the ``+__execute__+`` function with the same arguments.

- * For a ``DEPLOY_ACCOUNT`` transaction, the sequencer calls the ``constructor`` function with the transaction's ``constructor_calldata`` as input (as above, it is expected that the constructor's calldata successfully deserializes to the arguments in the constructor signature). After the successful execution of the constructor, the sequencer validates the transaction by calling the ``+__validate_deploy__+`` function.

- * For a ``DECLARE`` transaction, the sequencer validates the transaction by calling the ``+__validate_declare__+`` function.

- * For more information on the available transaction types and their fields, see [xref:architecture-and-concepts:network-architecture/transactions.adoc\[Transaction types\]](#).

- * For more information on the validation and execution stages, see [xref:architecture-and-concepts:network-architecture/transaction-life-cycle.adoc\[Transaction lifecycle\]](#).

Separating the validation and execution stages guarantees payment to sequencers for work completed and protects them from Denial of Service (DoS) attacks.

[#attacks_that_validation_limitations_prevent]
== Potential DoS

The validation functions have limitations, described below, that are designed to prevent the following DoS attacks on the sequencer:

- * An attacker could cause the sequencer to perform a large amount of work before a transaction fails validation. Two examples of such attacks are:

- ** Spamming ``INVOKE`` transactions whose ``+__validate__+`` requires many steps, but eventually fails

- ** Spamming ``DEPLOY_ACCOUNT`` transactions that are invalid as a result of the constructor or ``+__validate_deploy__+`` failing.

- * The above attacks are solved by making sure that the validation step is not resource-intensive, e.g. by keeping the maximal number of steps low. However, even if the validation is simple, the following "mempool pollution" attack could still be possible:

- . An attacker fills the mempool with transactions that are valid at the time they are sent.

. The sequencer is ready to execute them, thinking that by the time it includes them in a block, they will still be valid.

. Shortly after the transactions are sent, the attacker sends one transaction that somehow invalidates all the previous ones and makes sure it's included in a block, e.g. by offering higher fees for this one transaction.

An example of such an attack is having the implementation of ``+__validate__+`` checks that the value of a storage slot is ``1``, and the attacker's transaction later sets it to ``0``.

Restricting validation functions from calling external contracts prevents this attack.

[#limitations_of_validation]

== Limitations on validation

The limitations listed here apply to the following validation functions:

* ``+__validate__+``, ``+__validate_deploy__+``, and ``+__validate_declare__+``.

* A constructor, when run in a ``DEPLOY_ACCOUNT`` transaction. That is, if an account is deployed from an existing class via the ``deploy`` syscall, these limitations do not apply.

The validation functions have the following limitations:

* You cannot call functions in external contracts, only in your account contract.

+

[NOTE]

====

This restriction enforces a single storage update being able to invalidate only transactions from a single account. However, be aware that an account can always invalidate its own past transactions by e.g. changing its public key.

This limitation implies that the fees you need to pay to invalidate transactions in the mempool are directly proportional to the number of unique accounts whose transactions you want to invalidate.

====

* The maximum number of computational steps, measured in Cairo steps, for a validation function is 1,000,000.

* A builtin can be applied a limited number of times. For specific limits for each builtin, see [xref:tools:limits-and-triggers.adoc\[\]](#).

* The ``get_execution_info`` syscall behaves differently When raised from one of the ``validate`` functions:

** ``sequencer_address`` is set to zero

** ``block_timestamp`` returns the time (in UTC), rounded to the most recent hour.

** ``block_number`` returns the block number, rounded down to the nearest multiple of 100.

* The following syscalls cannot be called:

** ``get_block_hash``

** ``get_sequencer_address`` (this syscall is only supported for Cairo 0 contracts).

[id="invalid_transactions"]
== Invalid transactions

When the `+__validate__+`, `+__validate_deploy__+`, or `+__validate_declare__+`, function fails, the account in question does not pay any fee, and the transaction's status is `REJECTED`.

[id="reverted_transactions"]
== Reverted transactions

A transaction has the status `REVERTED` when the `+__execute__+` function fails. A reverted transaction is included in a block, and the sequencer is eligible to charge a fee for the work done up to the point of failure, similar to Ethereum.

== Implementation reference

Thanks to account abstraction, the logic of `+__execute__+` and the different validation functions is up to the party implementing the account.

To see a concrete implementation, see OpenZeppelin's link:<https://github.com/OpenZeppelin/cairo-contracts/blob/v0.14.0/src/account/account.cairo#L72>[account component].

This implementation adheres to link:<https://github.com/starknet-io/SNIPs/blob/main/SNIPS/snip-6.md>[SNIP6], which defines a standard for account interfaces.

[id="starknet_account_structure"]
= Starknet's account interface

Starknet's account structure is inspired by Ethereum's EIP-4337, where instead of EOAs, you use smart contract accounts with arbitrary verification logic.

While not mandatory at the protocol level, you can use a richer standard interface for accounts, defined in link:<https://github.com/starknet-io/SNIPs/blob/main/SNIPS/snip-6.md>[Starknet Improvement Proposal #6 (SNIP-6)]. SNIP-6 was developed by community members at OpenZeppelin, in close collaboration with wallet teams and other Core Starknet developers.

[#account_functions]
== Account functions

A valid account contract includes specific functions, depending on the type of the contract. For more information, see [xref:architecture-and-concepts:accounts/account-functions.adoc](#)].

[#replay_protection]
== Replay protection

In Starknet, similar to Ethereum, every contract has a nonce, including an account

contract. This nonce is sequential. The nonce of a transaction sent from an account must match the nonce of that account. After the transaction is executed, whether or not it is reverted, the nonce is incremented by one.

[NOTE]

====

In Starknet, only the nonce of account contracts, that is, those adhering to the above structure, can be non-zero.

In contrast, in Ethereum, regular smart contracts, known as `_Contract Accounts_`, as opposed to `_Externally Owned Accounts_` can increment their nonce by deploying smart contracts, that is, executing the ``CREATE`` and ``CREATE2`` opcodes.

For more information on accounts in Ethereum, see link:<https://ethereum.org/en/developers/docs/accounts/>[Ethereum Accounts] in the Ethereum documentation.

====

A nonce serves two important roles:

- * It guarantees transaction hash uniqueness, which is important for a good user experience.
- * It provides replay protection to the account: Because the signature is bound to a particular nonce, a malicious party cannot replay the transaction.

Starknet currently determines the nonce structure at the protocol level to be sequential. In the future, Starknet will consider a more flexible design, extending account abstraction to nonce management, previously referred to as `_nonce abstraction_`.
[id="deploying_new_accounts"]
= Deploying a new account

You can deploy a new account in the following ways:

- * Send a ``DEPLOY_ACCOUNT`` transaction. This method does not require a preexisting account.
- * Using the Universal Deployer Contract (UDC). This method requires an existing account to send the ``INVOKE`` transaction.

Upon receiving one of these transactions, the sequencer performs the following steps:

- . Runs the respective validation function in the contract, as follows:
 - ** When deploying with the ``DEPLOY_ACCOUNT`` transaction type, the sequencer executes the ``+__validate_deploy__+`` function in the deployed contract.
 - ** When deploying using the UDC, the sequencer executes the ``+__validate__+`` function in the contract of the sender's address.
- . Executes the constructor with the given arguments.
- . Charges fees from the new account address.

+

[NOTE]

====

If you use a ``DEPLOY_ACCOUNT`` transaction, the fees are paid from the address of the deployed account. If you use the UDC, which requires an ``INVOKE`` transaction, the fees are paid from the sender's account. For information on the differences between V1 and V3 ``INVOKE`` transactions, see [xref:network-architecture/transactions.adoc#invoke_transaction\[INVOKE` transaction\]](#) in `_Transaction types_`.

====

. Sets the account's nonce as follows:

** ``1``, when deployed with a ``DEPLOY_ACCOUNT`` transaction

** ``0``, when deployed with the UDC

== Deploying a new account with Starkli

Starkli simplifies account creation, whether you create an account as a Starknet wallet account, or using the UDC.

To create and deploy a new account, use Starkli's ``starkli account`` command.

For more information on creating a new account as a Starknet wallet account with a ``DEPLOY_ACCOUNT`` transaction, or by using the UDC with an ``INVOKE`` transaction, see [link:https://book.starkli.rs/accounts\[Accounts\]](https://book.starkli.rs/accounts[Accounts]) in the Starkli Book.

[#DEPLOY_ACCOUNT_restrictions]

== ``DEPLOY_ACCOUNT`` constructor restrictions

The constructor of the ``DEPLOY_ACCOUNT`` transaction has the following limitations:

* Restricted access to ``sequencer_address`` in the ``get_execution_info`` syscall. The syscall returns zero values for ``sequencer_address``

* Restricted access to the following syscalls:

** ``get_block_hash`` for Cairo contracts

** ``get_sequencer_address`` for Cairo 0 contracts

== Additional resources

* [link:https://book.starkli.rs/accounts\[Accounts\]](https://book.starkli.rs/accounts[Accounts]) in the Starkli Book

* [xref:accounts/universal-deployer.adoc\[\]](#)

* [xref:network-architecture/transactions.adoc\[\]\[id="what_is_an_account"\]](#)

= What is an account?

An account represents a user onchain, and enables that user to interact with the blockchain.

Through an account, you can send transactions and interact with other contracts. In

order for you to own an onchain asset, such as an ERC-20 token or an NFT, that asset must be associated with your account address.

[id="ethereum_account_structure"]
== Ethereum account structure

Within Ethereum individual user accounts are known as Externally Owned Accounts (EOAs).

EOAs differ from smart contracts in that EOAs are not controlled by code, but rather by a pair of private and public keys.

The account's address is derived from those keys and only by possessing the private key can you initiate transactions from an account. While Ethereum contracts are `_passive_`, that is, they can only change if they were called inside a transaction, EOAs can initiate transactions.

While simple, because the signature scheme is fixed, EOAs have some drawbacks, including the following:

- * Control over the private key gives complete control over the account, so you must keep your seed phrase secure yet accessible.
- * Limited flexibility surrounding wallet functionality

EIP-4337 is a design proposal for Ethereum that outlines `_account abstraction_`, whereby all accounts are managed via a dedicated smart contract on the Ethereum network, as a way to increase flexibility and usability. You can add custom logic on top of the basic EOA functionality, thereby bringing account abstraction into Ethereum.

[id="account_abstraction"]
== What is Account Abstraction?

Account abstraction enables more flexible account management. Rather than the protocol determining an account's behavior, an `_account contract_`, which is a smart contract with programmable logic, defines a user's account.

Using account abstraction you can now program how your account functions.

For example, you can:

- * Determine what it means for a signature to be valid, or what contracts your account is allowed to interact with. This is known as `_signature abstraction_`.
- * Pay transaction fees in different tokens. This is known as `_fee abstraction_`.
- * Design your own replay protection mechanism and allow sending multiple uncoupled

transactions in parallel.

+

In Ethereum, you cannot send two transactions in parallel, you must wait for confirmation of the first before sending the second. Otherwise, the second transaction can be rejected due to an invalid nonce. With account abstraction, a sequential nonce is not required. This is known as `_nonce abstraction_`.

Today, Starknet offers signature abstraction. In the future, we will enrich the current account abstraction design. For example, see the link:<https://community.starknet.io/t/starknet-account-abstraction-model-part-1/781>[paymaster proposal] for fee abstraction in the Starknet Community Forum.

[id="examples"]

== Examples of customizing account functionality

Two examples of how you might program an account to function using account abstraction are:

[horizontal]

Social recovery:: A process where if you lose your wallet, you are able to retrieve it via a selected social network, vastly improving the typical experience of wallet recovery.

Operating your account via facial recognition:: With signature abstraction, you can use your phone's native hardware to sign transactions, making it practically impossible to take control of another user's account, even if your phone is stolen.

[id="simplified_transaction_flow"]

= Simplified transaction flow

The key stages of transaction lifetime are:

. The sequencer selects a transaction from the mempool and calls the ``+__validate__+`` function.

. If the transaction is valid, the sequencer calls the ``+__execute__+`` function.

. If ``+__execute__+`` runs successfully, the sequencer includes the transaction in the block, charges the fee, and proceeds to work on the next transaction.

. After completing the block, the sequencer sends the block to the prover.

// Why is this section relevant to this topic?

[id="a_payment_mechanism"]

== The payment mechanism

The sequencer receives fees in ETH in return for including transactions in a block.

For more details on how the transaction fee is computed, see [xref:../network-](#)

architecture/fee-mechanism.adoc[Gas and transaction fees].
[id="universal_deployer_contract"]
= Universal Deployer Contract (UDC)

:deploy-syscall: xref:architecture-and-concepts:smart-contracts/system-calls-cairo1.adoc#deploy[deploy syscall]

The Universal Deployer Contract (UDC) is a singleton smart contract that wraps the {deploy-syscall} to expose it to any contract that doesn't implement it, such as account contracts. You can think of it as a standardized generic factory for Starknet contracts.

And since Starknet has no deployment transaction type, it offers a standardized way to deploy smart contracts by following the <https://community.starknet.io/t/snip-deployer-contract-interface/2772>[standard deployer interface] and emitting a `ContractDeployed` event.

For more information see the <https://community.starknet.io/t/snip-deployer-contract-interface/2772>[proposal for the standard deployer interface].
For details on the motivation and the decision making process, see the <https://community.starknet.io/t/universal-deployer-contract-proposal/1864>[Universal Deployer Contract proposal].

== UDC address

The UDC address is
`0x041a78e741e5af2fec34b695679bc6891742439f7afb8484ecd7766661ad02bf` in
Mainnet, Sepolia testnet, and starknet-devnet. This address might change in the future
when it is migrated to a modern version of Cairo.

== Interface

```
[source,cairo]
----
trait IUniversalDeployer {
    fn deployContract(
        class_hash: ClassHash,
        salt: felt252,
        unique: bool,
        calldata: Span<felt252>
    ) -> ContractAddress;
}
----
```

== Deploying a contract with the UDC

.Procedure

. Declare the contract with a `DECLARE` transaction, or ensure that the contract has been declared.

+

For more information, see the <xref:architecture-and-concepts:network-architecture/transactions.adoc#declare-transaction> [`DECLARE` transaction].

. Call the `deployContract` function in the UDC.

.Example implementation in Cairo:

[source,cairo]

#[starknet::interface]

```
trait IUniversalDeployer<TContractState> {  
    fn deployContract(  
        ref self: TContractState,  
        class_hash: ClassHash,  
        salt: felt252,  
        unique: bool,  
        calldata: Span<felt252>  
    ) -> ContractAddress; }
```

const UDC_ADDRESS: felt252 =

0x041a78e741e5af2fec34b695679bc6891742439f7afb8484ecd7766661ad02bf;

```
fn deploy() -> ContractAddress {  
    let dispatcher = IUniversalDeployerDispatcher {  
        contract_address: UDC_ADDRESS.try_into().unwrap()  
    };  
};
```

// deployment parameters

let class_hash = class_hash_const::<

0x5c478ee27f2112411f86f207605b2e2c58cdb647bac0df27f660ef2252359c6
>();

let salt = 1234567879;

let unique = false;

let mut calldata = array![];

// the UDC returns the deployed contract address

dispatcher.deployContract(class_hash, salt, unique, calldata.span())

}

== Deployment types

The Universal Deployer Contract offers two types of addresses to deploy: origin-

dependent and origin-independent.

As the names suggest, the origin-dependent type includes the deployer's address in the address calculation,

whereas, the origin-independent type does not.

The `unique` boolean parameter ultimately determines the type of deployment.

[IMPORTANT]

====

When deploying a contract that uses `get_caller_address` in the constructor calldata, remember that the UDC, not the account, deploys that contract.

Therefore, querying `get_caller_address` in a contract's constructor returns the UDC's address, not the account's address.

====

=== Origin-dependent

By making deployments dependent upon the origin address, users can reserve a whole address space to prevent someone else from taking ownership of the address.

Only the owner of the origin address can deploy to those addresses.

Achieving this type of deployment necessitates that the origin sets `unique` to `true` in the `deployContract` call.

Under the hood, the function call leverages the origin's address and creates a hashchain by hashing the origin's address with the given salt.

To deploy a unique contract address pass:

[,js]

```
let deployed_addr = udc.deployContract(class_hash, salt, true, calldata.span());
```

=== Origin-independent

Origin-independent contract deployments create contract addresses independent of the deployer and the UDC instance.

Instead, only the class hash, salt, and constructor arguments determine the address.

This type of deployment enables redeployments of accounts and known systems across multiple networks.

To deploy a reproducible deployment, set `unique` to `false`.

[source,cairo]

```
let deployed_addr = udc.deployContract(class_hash, salt, false, calldata.span());
```

== Deploying the UDC

[NOTE]

=====

The UDC has already been deployed on most networks and development environments. The standard requires the UDC to be deployed passing ``deploy_from_zero=true`` and ``salt=0`` as arguments to the `{deploy-syscall}`. This results in a deterministic and predictable address across all instances of Starknet, facilitating SDK integration and reproducibility of deployments.

=====

== API specification

=== ``deployContract`` method

Deploy a contract through the Universal Deployer Contract.

[source,cairo]

```
fn deployContract(  
    classHash: ClassHash,  
    salt: felt252,  
    unique: bool,  
    calldata: Span<felt252>  
) -> ContractAddress
```

=== ``ContractDeployed`` event

Emitted when ``deployer`` deploys a contract through the Universal Deployer Contract.

[source,cairo]

```
#[derive(Drop, starknet::Event)]  
struct ContractDeployed {  
    address: ContractAddress,  
    deployer: ContractAddress,  
    unique: bool,  
    classHash: ClassHash,  
    calldata: Span<felt252>,  
    salt: felt252,  
}
```

[id="hash_functions"]

= Hash functions
// :stem: latexmath

[id="domain_and_range"]
== Domain and range

All hashes outputs are eventually mapped to elements in stem: \mathbb{F}_P , where $\text{stem}: P = 2^{251} + 17 \cdot 2^{192} + 1$.

There are three hash functions used throughout Starknet's specifications:

- * stem: $\text{sn_keccak}: \{0,1\}^* \rightarrow \mathbb{F}_P$
- * stem: $\text{pedersen}: \mathbb{F}_P^2 \rightarrow \mathbb{F}_P$
- * stem: $\text{poseidon}: \mathbb{F}_P^* \rightarrow \mathbb{F}_P$

[id="starknet_keccak"]
== Starknet Keccak

Starknet Keccak, usually denoted by stem: sn_keccak , is defined as the first 250 bits of the Keccak256 hash. For Starknet Keccak, Keccak256 is augmented in order to fit into a field element.

[id="pedersen_hash"]
== Pedersen hash

Pedersen hash makes use of the following STARK friendly elliptic curve over stem: \mathbb{F}_P :

[stem]
++++
 $y^2 = x^3 + \alpha x + \beta$
++++

where

- * stem: $\alpha = 1$
- * stem: $\beta = 3141592653589793238462643383279502884197169399375105820974944592307816406665$

[id="definition"]
=== Definition

Given an input stem: $(a,b) \in \mathbb{F}_P^2$, we begin by breaking it into stem: $(a_{\text{low}}, a_{\text{high}}, b_{\text{low}}, b_{\text{high}})$, where the low part consists of the low 248 bits of the element and the high part consists of the high 4 bits of the element. Our Pedersen hash is then defined by:

[stem]

++++

$$h(a,b) = \text{left}[\text{shift_point} + a_{\text{low}} \cdot P_0 + a_{\text{high}} \cdot P_1 + b_{\text{low}} \cdot P_2 + b_{\text{high}} \cdot P_3]_x$$

++++

where the values of the constants stem:[shift_point, P_0, P_1, P_2, P_3] can be found in link:https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/crypto/signature/fast_pedersen_hash.py[fast_pedersen_hash.py^], and stem:[P_x] denotes the stem:[x] coordinate of the point stem:[P].

For more information, see xref: cryptography/stark-curve.adoc[STARK curve].

[id="poseidon_hash"]

== Poseidon hash

`_Poseidon_` is a family of hash functions designed to be very efficient as algebraic circuits. As such, they can be very useful in ZK-proving systems such as STARKs.

Poseidon is a sponge construction based on the Hades permutation. Starknet's version of Poseidon is based on a three-element state permutation.

A Poseidon hash of up to 2 elements is defined as follows.

[stem]

++++

$$\text{poseidon}_1(x) := \text{left}[\text{hades_permutation}(x, 0, 1)]_0$$

++++

[stem]

++++

$$\text{poseidon}_2(x,y) := \text{left}[\text{hades_permutation}(x,y,2)]_0$$

++++

Where $\text{left}[\cdot]_j$ denotes taking the stem:[j]'th coordinate of a tuple.

.Additional resources

- * xref:#poseidon_array_hash[Poseidon hash with an arbitrary number of inputs]
- * link:<https://github.com/starkware-industries/poseidon/blob/main/poseidon3.txt>[Parameters for defining the Poseidon permutation used in Starknet]
- * link:<https://github.com/CryptoExperts/poseidon>[Reference implementation in C and assembly of the above by CryptoExperts]

[id="array_hashing"]
== Array hashing

[id="pedersen_array_hash"]
=== Pedersen

Let $\text{stem}:[h]$ denote the pedersen hash function, then given an array $\text{stem}:[a_1, \dots, a_n]$ of $\text{stem}:[n]$ field elements we define $\text{stem}:[h(a_1, \dots, a_n)]$ to be:

[stem]
++++
 $h(\dots h(h(0, a_1), a_2), \dots, a_n), n)$
++++

[id="poseidon_array_hash"]
=== Poseidon

Let $\text{stem}:[\text{hades}:\mathbb{F}_{P^3} \rightarrow \mathbb{F}_{P^3}]$ denote the Hades permutation, with Starknet's parameters, then given an array $\text{stem}:[a_1, \dots, a_n]$ of $\text{stem}:[n]$ field elements we define $\text{stem}:[\text{poseidon}(a_1, \dots, a_n)]$ to be the first coordinate of $\text{stem}:[H(a_1, \dots, a_n; 0, 0, 0)]$, where:

[stem]
++++
 $H(a_1, \dots, a_n; s_1, s_2, s_3) = \begin{cases} H(\text{big}(a_3, \dots, a_n; \text{hades}(s_1 + a_1, s_2 + a_2, s_3)), & \& \text{if } n \geq 2 \\ \text{hades}(s_1 + a_1, s_2 + 1, s_3), & \& \text{if } n = 1 \\ \text{hades}(s_1 + 1, s_2, s_3), & \& \text{if } n = 0 \end{cases}$
++++

For an implementation of the above in Python, see link:https://github.com/starkware-libs/cairo-lang/blob/12ca9e91bbdc8a423c63280949c7e34382792067/src/starkware/cairo/common/poseidon_hash.py#L46[poseidon_hash.py], and for an equivalent Cairo implementation, see link:https://github.com/starkware-libs/cairo-lang/blob/12ca9e91bbdc8a423c63280949c7e34382792067/src/starkware/cairo/common/builtin_poseidon/poseidon.cairo#L28[poseidon.cairo] in the cairo-lang GitHub repository.
= The STARK field

// The field element type in Starknet is based on the STARK field in the underlying Cairo VM. In other words, a value $\text{stem}:[x]$ of a field element type is an integer in the range of $\text{stem}:[0 \leq x < P]$.

The `_STARK field_` is the finite field `stem:[$$\mathbb{F}_P$]`, where `stem:[$$P$]` is a prime number, calculated as follows:

```
[stem]
++++
P = 2^{251} + 17*2^{192} + 1
++++
```

The Cairo VM uses the STARK field, referred to as a field element, or `_felt_`. The ``felt252`` type in Cairo refers to elements of this field.`[id="stark_curve"]`
= The STARK curve
`[id="stark_curve"]`

`:stem: latexmath`

The STARK curve is an elliptic curve defined as follows:

```
[stem]
++++
y^2 \equiv x^3 + \alpha \cdot x + \beta \pmod{p}
++++
```

where:

```
[stem]
++++
\begin{align*} \alpha &= 1 \quad \beta = 3141592653589793238462643383279502884197 \\ &169399375105820974944592307816406665 \quad \\ p &= 36185027886661312136973227830950701056231072153315966999730920561 \\ &35872020481 \quad \&= 2^{251} + 17 \cdot 2^{192} + 1 \\ \end{align*}
++++
```

The Generator point used in the ECDSA scheme is:

```
[stem]
++++
\begin{split} G = (87473945107800776645746498977432208364927860753324948115 \\ 1382481072868806602, \quad 152666792071518830868575557812948353041420400780 \\ 739481342941381225525861407) \end{split}
++++
```

The STARK curve is commonly used in smart contracts but not distinguished by the Starknet protocol.`[id="economics_of_starknet"]`
= The token economics of Starknet

:description: How token economics work in a block chain in general, and specifically in Starknet. The purpose of the Starknet token, its supply, and distribution.

:keywords: tokenomics, STRK, token economics, economic mechanisms of Starknet

Starknet is a developing decentralized protocol and the economic mechanisms described here, also known as tokenomics, are subject to change based on governance decisions made by the larger community of Starknet. For more details on Starknet's governance processes see the link:<https://governance.starknet.io/>[Starknet Governance Hub]. This document describes certain economic fundamentals of the Starknet token. This document is intended for informational purposes only and is meant to outline the usage and functionalities of the asset within Starknet. It is important to understand that the primary purpose of the Starknet token, STRK, is to facilitate operations and activities on Starknet and it is not intended to serve as an investment.

[#why_are_economics_relevant]

== Why are economics relevant?

Blockchains work through a combination of cryptography and economic incentives. Cryptography limits what actors in the system can do, for example, transactions must be validly signed to be included in the chain. Economic incentives encourage actors to voluntarily perform actions that maintain the network's capabilities when spending their own resources, for example, miners or stakers actively publish new blocks to the chain because they can receive fees and new tokens as a reward. Blockchains are valuable because they are data structures maintained by diverse and, ideally, large groups of otherwise unaffiliated persons. This gives them resilience: Any one participant can disappear, but the data structure is preserved. This also gives them censorship resistance: No single person can unilaterally decide to forbid certain persons from using the network.

Starknet operates as a Layer 2 (L2) network on top of Ethereum. Today, Starknet achieves secure low-cost transactions by using the STARK cryptographic proof system to reduce the size of transaction data while preserving and verifying the integrity of that data. Still under development, Starknet will achieve resilience and censorship resistance by using a token, the Starknet token (STRK), to incentivize network participants to sequence transactions for users of the network and to ensure that there is a provably fair mechanism, a proof-of-stake mechanism, to determine who should sequence and submit a proof for the network blocks. A proof-of-stake mechanism might also be used to facilitate data availability solutions and other significant services required for network operations.

[#purpose_of_the_token]

== The purpose of the STRK token

STRK is the mechanism for paying fees to enable operation of the network, maintaining and securing the network by enabling staking for consensus, and deciding on

Starknet's values and technology goals by voting for governance proposals.

* *Transaction fees:* Originally, fees in Starknet were paid only in Ether (ETH). As of v.0.13.0, fees for transactions on the network can be paid using STRK, as well as ETH.

+

A portion of the fees paid in STRK are converted to ETH by the receiving sequencer, in order to cover Ethereum L1 gas costs, which, due to the specifications of the Ethereum protocol, must be paid in ETH.

* *Staking:* Certain services that are critical to the liveness and security of Starknet may require the staking of Starknet tokens. These services might be offered by multiple providers, and could include sequencing, reaching temporary L2 consensus before L1 finality is reached, STARK-proving services, and data availability provisioning, to name a few examples. These [https://starkware.co/resource/starknet-decentralization-a-roadmap-in-broad-strokes/\[protocol changes\]](https://starkware.co/resource/starknet-decentralization-a-roadmap-in-broad-strokes/[protocol%20changes]) are still under discussion within the larger governance community and are targeted for [https://starkware.co/resource/starknet-decentralization-a-roadmap-in-broad-strokes/\[2024 -2025\]](https://starkware.co/resource/starknet-decentralization-a-roadmap-in-broad-strokes/[2024%20-2025]).

* *Governance:* Proposals for improving Starknet might require a minimal token support threshold. Voting, either directly or via delegation, will be required for changes to the protocol that are essential to Starknet's liveness, security, and maintenance. Today, for example, major updates to the Starknet Operating System require the approval of token holders.

For more information about Governance see the <https://governance.starknet.io/> [Starknet Governance Hub]

[IMPORTANT]

====

As discussed above, the Starknet tokens are digital assets intended to support the operation and usage of Starknet and are not offered as an investment. As such, the Starknet tokens do not represent any equity in StarkWare or the Starknet Foundation, nor do they provide any participation right in StarkWare or grant any right of claim from StarkWare or the Starknet Foundation.

====

[#supply_and_distribution]

== Supply and distribution

Ten billion Starknet tokens were initially created by StarkWare in May 2022 and minted onchain on November 30, 2022.

The existing ten billion tokens have been or are planned to be distributed according to the following:

[cols="1,2",]

|===

// |Percentage of total| Recipients | Details

|*20.04%: Early Contributors* |Tokens allocated for StarkWare's team members and early contributors. These tokens are subject to a lock-up schedule, as further detailed below.

|*18.17%: Investors* |Tokens allocated for StarkWare's investors. These tokens are subject to a lock-up schedule, as further detailed below.

|*10.76%: StarkWare* |Tokens allocated for StarkWare for operation services such as to pay fees, provide other services on Starknet, and engage other service providers.

|*12.93%: Grants including Development Partners (aka DPs)* |Tokens allocated for grants for research or work done to develop, test, deploy and maintain the Starknet protocol. The process for applications and allocations related to Starknet Foundation Grants will be outlined in a post at a later date.

|*9.00%: Community Provisions* |Tokens distributed to those who contributed to Starknet and powered or developed its underlying technology.

|*9.00%: Community Rebates* |Tokens allocated for rebates in Starknet tokens to partially cover the costs of onboarding to Starknet from Ethereum. Community rebates are not yet available and will be announced in 2024 in a subsequent post.

|*10.00%: Foundation Strategic Reserves* |Tokens allocated for the Starknet Foundation to fund ecosystem activities that are aligned with the <https://www.starknet.io/en/content/introducing-the-starknet-foundation>[Foundation's mission].

|*8.10%: Foundation Treasury* |Token allocated for the Starknet Foundation's treasury available for operations and other future initiatives by the Starknet Foundation.

|*2.00%: Donations* |Tokens reserved for donations to institutions and organizations, such as universities, NGOs, etc, as decided by the Starknet Foundation.

|===

image:planned_distribution_STRK.jpg[Planned distribution of STRK]

To align long-term incentives of the Investors and Early Contributors with the interests of the Starknet community, and following common practice in decentralized ecosystems, all tokens allocated to Investors and Early Contributors is subject to the following lock-up schedule, where percentages are based on the total token supply:

* Up to 0.64% (64 million tokens) will be unlocked on the 15th of each month, starting April 15, 2024, and going through March 15, 2025, for a total of 7.68% (768 million tokens) unlocked by March 15, 2025. +

* Up to 1.27% (127 million tokens) will be unlocked on the 15th of each month, starting April 15, 2025, and going through March 15, 2027, for a total of 30.48% (3.048 billion tokens) unlocked by March 15, 2027.

.Estimated supply of STRK in circulation

[#estimated_supply]

image::STRK_estimated_circulating_supply.jpg[Estimated supply of STRK in circulation]

====

_The graph, xref:#estimated_supply[], excludes newly circulating tokens resulting from inflation or staking (see below)._

Token allotments currently retained by the Starknet Foundation, while contractually unlocked, are not considered circulating unless granted, donated, or otherwise allocated out of originating wallets through future grants, provisions, donations, developer initiatives, or other programs.

====

Through this lock-up period, token holders cannot transfer, sell, or pledge their STRK tokens. Delegation of voting is permitted with locked tokens and, when available, staking might also be permitted.

The total circulating supply of tokens is planned to increase over time with the minting of new tokens by the protocol, as staking rewards, block rewards, or other rewards associated with the staking process. Such minting will be made pursuant to a schedule that will be determined with the community at a later point, not before Starknet services are more decentralized. The supply in circulation might not, therefore, remain fixed. However, as long as StarkWare is the sole operator of the Starknet sequencer, there will be no issuance of new tokens for the purpose of block rewards. For more information, see link:<https://starkware.co/resource/a-token-minting-proposal-to-manage-inflation/>[_A token-minting proposal to manage inflation_].

[#risks_and_disclaimers]

== Risks and disclaimers

Starknet is a developing decentralized protocol and the economic mechanisms described herein are subject to change based on decisions made by the larger community of Starknet builders and users. Starknet relies upon third parties to adopt and implement software and protocols as users and contributors of Starknet. It also relies, in whole or partly, on third parties to develop, supply and otherwise support it. There is no assurance or guarantee that such third parties will continue to participate in the network or that the network will continue to function as intended.

The technical documents provided herein describe certain planned and specified economic fundamentals of a digital asset, STRK. These materials are intended for informational purposes only and are meant to outline the usage and functionalities of the asset within Starknet. It is important to understand that the primary purpose of STRK is to pay for fees, provide a mechanism for securing consensus, and allow for decentralized governance on Starknet; it is not intended to serve as an investment.

Starknet relies upon third parties to adopt and implement the software and protocols as users of Starknet. It also relies, in whole or partly, on third parties to develop, supply and otherwise support it. As a Layer 2 network over Ethereum, Starknet also relies

upon third parties maintaining and operating the Ethereum network. There is no assurance or guarantee that those third parties will complete their work, properly carry out their obligations, and/or otherwise meet anyone's needs.

STRK, as the native token of Starknet, may be subject to the risks of the Starknet network, including, without limitation, the following: (i) the technology associated with Starknet may not function as intended; (ii) the details of the Starknet token economics including the total supply and distribution schedule may be changed due to decisions made by the consensus of participants of the Starknet network; (iii) Starknet may fail to attract sufficient interest from key stakeholders or users; (iv) Starknet may not progress satisfactorily and Starknet tokens may not be useful or valuable; (v) Starknet may suffer from attacks by hackers or other individuals; and (vi) Starknet is comprised of open-source technologies that depend on a network of computers to run certain software programs to process transactions, and because of this model StarkWare and the Starknet Foundation have limited control over Starknet.

Risks related to blockchain technology in general and Starknet in particular may impact the usefulness of Starknet, and, in turn, the utility or value of STRK. The software and hardware, technology and technical concepts and theories applicable to Starknet and STRK are still in an early development stage and unproven, there is no warranty that Starknet will achieve any specific level of functionality or success, nor that the underlying technology will be uninterrupted or error-free, and there is an inherent risk that the technology could contain weaknesses, vulnerabilities or bugs causing, potentially, the complete loss of any Starknet tokens held by Starknet users.

As with most commonly used public blockchains, STRK is accessed using a private key that corresponds to the address at which they are stored. If the private key, or the "seed" used to create the address and corresponding private key are lost or stolen, the tokens associated with that address might be unrecoverable and will be permanently lost.

Public blockchain-based systems, including Starknet and the underlying Ethereum network, depend on independent verifiers, and therefore may be vulnerable to consensus attacks including, but not limited to, double-spend attacks, majority voting power attacks, race condition attacks, and censorship attacks. These attacks, if successful, could result in the permanent loss of STRK.

Starknet, STRK, and blockchain technology are nascent, and there may be additional risks not described above or that may be new or unanticipated. We recommend only using Starknet or holding STRK if you are familiar with the technology and aware of the risks.

This document and its contents are not, and should not be construed as, an offer to sell, or the solicitation of an offer to buy, any tokens, nor should it or any part of it form the basis or be relied on in connection with any contract or commitment whatsoever. This document is not advice of any kind, including legal, investment, financial, tax, or

any other professional advice. Nothing in this document should be read or interpreted as a guarantee or promise of how the Starknet network or its STRK will develop, be utilized, or accrue value.

All information in this document is provided on an “as is” basis without any representation or warranty of any kind. This document only outlines current plans, which could change at the discretion of various parties, and the success of which will depend on many factors outside of Starknet Foundation’s control. Such future statements necessarily involve known and unknown risks, which may cause actual performance and results in future periods to differ materially from what we have described or implied in this document. StarkWare and the Starknet Foundation disclaim all warranties, express or implied, to the fullest extent permitted by law with respect to the functionality of Starknet and STRK.

= Architecture

[id="block_structure"]

= Block structure

A Starknet block is a list of transactions and a block header that contains the following fields:

[%autowidth]

|===

| Name | Type | Description

|`block_number`|`u64`|The number, that is, the height, of this block.

|`parent_block_hash`|`felt252`|The hash of the block’s parent.

|`global_state_root`|`felt252`|The xref:../network-architecture/starknet-state.adoc#state_commitment[state commitment] after the block.

|`sequencer_address`|`ContractAddress`|The Starknet address of the sequencer that created the block.

|`block_timestamp`|`Timestamp`|The time at which the sequencer began building the block, in seconds since the Unix epoch.

|`transaction_count`|`u32`|The number of transactions in the block.

|`events_count`|`u32`|The number of events in the block.

|`state_diff_length`|`u32`|The sum of number of storage diffs, nonce updates, deployed contracts and declared classes.

|`state_diff_commitment`|`felt252`|The poseidon hash of the state diff of the block, see below for more details.

| `transactions_commitment` | `felt252` | A commitment to the transactions included in the block.

The root of a height-64 binary Merkle Patricia trie. The leaf at index stem:[\$i\$] corresponds to stem:[\$\$\{h(\text{transaction_hash}), \text{signature}\})\$\$].

| `events_commitment` | `felt252` | The root of a height-64 binary Merkle Patricia trie.

The leaf at index stem:[\$i\$] corresponds to the hash of the stem:[\$i^{th}\$] event emitted in the block.

See below for a description on how event hashes are computed.

| `receipts_commitment` | `felt252` | The root of a height-64 Merkle-Patricia trie.

The leaf at index stem:[\$i\$] corresponds to the hash of the stem:[\$i^{th}\$] transaction receipt.

See below for a description on how receipt hashes are computed.

| `l1_gas_price` | `(u128, u128)` | The price of L1 gas that was used while constructing the block. L1 gas prices apply to storage updates and L1->L2 messages. As of March 2023, computation is also priced in terms of L1 gas, but this will change in the future. The first `u128` value is the price in wei. The second is the price in fri.

| `l1_data_gas_price` | `(u128, u128)` | The price of L1 blob gas that was used while constructing the block. If the `l1_da_mode` of the block is set to `BLOB`, L1 blob gas prices determines the storage update cost.

The first `u128` value is the price in wei. The second is the price in fri.

| `l1_da_mode` | `String` | `CALLDATA` or `BLOB`, depending on how Starknet state diffs are sent to L1.

| `protocol_version` | `String` | The version of the Starknet protocol used when creating the block.

|===

[#block_hash]
== Block hash

A block hash is defined as the Poseidon hash of the header's fields, as follows:

[, , subs="quotes"]

```
_h_(0x5) = _h_(  
  "STARKNET_BLOCK_HASH0",  
  block_number,  
  global_state_root,  
  sequencer_address,
```

```

    block_timestamp,
    transaction_count || event_count || state_diff_length || l1_da_mode,
    state_diff_commitment,
    transactions_commitment
    events_commitment,
    l1_gas_price_in_wei,
    l1_gas_price_in_fri,
    l1_data_gas_price_in_wei,
    l1_data_gas_price_in_fri
    receipts_commitment
    0,
    parent_block_hash
)
----

```

Where:

- ``_h_`` is the xref:../cryptography/hash-functions.adoc#poseidon-hash[Poseidon hash].
- ``||`` denotes concatenation, ``transaction_count``, ``event_count`` and ``state_diff_length`` are given 64 bits each, and ``l1_da_mode`` is one bit where 0 denotes ``CALLDATA`` and 1 denotes ``BLOB``.

For a reference implementation, see the link:https://github.com/starkware-libs/sequencer/blob/bb361ec67396660d5468fd088171913e11482708/crates/starknet_api/src/block_hash/block_hash_calculator.rs#L68[sequencer repository].

```

[#state_diff_hash]
== State diff commitment

```

The state diff commitment is obtained by the chain-hash of the following:

- updates to contract addresses stem: $[\$c_1, \dots, c_n\$]$, with diffs stem: $[(k^{1_1}, v^{1_1}), \dots, (k^{1_{m_1}}, v^{1_{m_1}}), \dots, (k^{n_1}, v^{n_1}), \dots, (k^{n_{m_n}}, v^{n_{m_n}})]$
- deployed contracts stem: $[(\text{deployed_address}_1, \text{deployed_class_hash}_1), \dots, (\text{deployed_address}_\ell, \text{deployed_class_hash}_\ell)]$
- declared classes stem: $[(\text{declared_class_hash}_1, \text{declared_compiled_class_hash}_1), \dots, (\text{declared_class_hash}_d, \text{declared_compiled_class_hash}_d)]$
- replaced classes stem: $[(\text{replaced_contract_address}_1, \text{new_class_hash}_1), \dots, (\text{replaced_contract_address}_r, \text{new_class_hash}_r)]$
- updated nonces stem: $[(\text{account}_1, \text{new_nonce}_1), \dots, (\text{account}_k, \text{new_nonce}_k)]$

More formally, the state-diff hash is given by:

```

[stem]

```

```

+++++
\begin{align}
& \text{h}\big( \text{ \& \text{"STARKNET\_STATE\_DIFF0"}}, \backslash \\
& \text{\& \quad \text{\ell} + r}, \backslash \\
& \text{\& \quad \text{deployed\_address}\_1, \text{deployed\_class\_hash}\_1,...,} \\
& \text{\text{deployed\_address}\_ell, \text{deployed\_class\_hash}\_ell}, \backslash \\
& \text{\& \quad \text{replaced\_contract\_address}\_1, \text{new\_class\_hash}\_1,...,} \\
& \text{\text{replaced\_contract\_address}\_r, \text{new\_class\_hash}\_r} \backslash \\
& \text{\& \quad d}, \backslash \\
& \text{\& \quad \text{declared\_class\_hash}\_1, \text{declared\_compiled\_class\_hash}\_1, ...,} \\
& \text{\text{declared\_class\_hash}\_d, \text{declared\_compiled\_class\_hash}\_d}, \backslash \\
& \text{\& \quad 1}, \backslash \\
& \text{\& \quad 0}, \backslash \\
& \text{\& \quad n}, \backslash \\
& \text{\& \quad c\_1} \backslash \\
& \text{\& \quad k^1\_1, v^1\_1,...,k^1\_{m\_1}, v^1\_{m\_1}} \backslash \\
& \text{\& \quad \vdots} \backslash \\
& \text{\& \quad c\_n} \backslash \\
& \text{\& \quad k^n\_1, v^n\_1,...,k^n\_{m\_n}, v^n\_{m\_n}} \backslash \\
& \text{\& \quad k} \backslash \\
& \text{\& \quad \text{account}\_1, \text{new\_nonce}\_1,...,\text{account}\_k, \text{new\_nonce}\_k} \backslash \big) \\
\end{align}
+++++

```

Where:

- stem:[\$h\$] is the Poseidon hash function
- stem:[\$1, 0\$] in the hash computation are placeholders that may be used in the future

[#receipt_hash]
 == Receipt hash

A transaction receipt consists of the following fields:

```

[%autowidth]
|===
| Name | Type | Description
| `transaction_hash` | `felt252` | the hash of the transaction
| `actual_fee` | `u128` | the fee paid on-chain
| `events` | `List<Event>` | ordered list of the events emitted by the transaction
| `messages` | `List<L2toL1Message>` | ordered list of the l2->l1 messages sent by the transaction
| `revert_reason` | `String` | The revert reason, in case the transaction was reverted
| `l1_gas_consumed` | `u128` | The amount of l1 gas that was consumed

```

| `l1_data_gas_consumed` | `u128` | The amount of l1 data (blob) gas that was consumed
 | `l2_gas_consumed` | `u128` | The amount of l2 gas that was consumed

|===

The hash of the transaction receipt is given by:

[,subs="quotes"]

```
_h_(receipt) = _h_(
  transaction_hash,
  actual_fee,
  h(messages),
  sn_keccak(revert_reason),
  h(l2_gas_consumed, l1_gas_consumed, l1_data_gas_consumed)
)
```

Where:

- h is the Poseidon hash function
- given messages stem:[\$m_1=(\text{from}_1, \text{to}_1, \text{payload}_1)...\text{m}_n=(\text{from}_n, \text{to}_n, \text{payload}_n)\$], their hash is given by:

[stem]

++++

$h(n, \text{from}_1, \text{to}_1, h(\text{payload}_1), \dots, \text{from}_n, \text{to}_n, h(\text{payload}_n))$

++++

where each message's payload is length-prefixed.

- events are omitted from the receipt's hash since they are committed separately in the block.

[#event_hash]

== Event hash

The hash of an event stem:[\$(\text{keys}, \text{data})\$] emitted by a contract whose address is `emitter_address` and a transaction whose hash is `tx_hash` is given by:

[stem]

++++

$h(\text{big}(\text{emitter_address}, \text{tx_hash}, h(\text{keys}), h(\text{data})) \text{big})$

++++

Where $\text{stem}:[\$h\$]$ is the Poseidon hash function.

[NOTE]

====

Zeros inside the hash computation of an object are used as placeholders, to be replaced in the future by meaningful fields.

====[id="data_availability"]

= Data availability

[id="introduction"]

== Introduction

Starknet is a Validity Rollup, which means that after consolidating and proving a set of Layer 2 changes, it updates, on L1, the latest proven L2 state. Alongside the proof, it publishes the state diff on L1. The state diff is the difference between the previous and new states.

Anyone monitoring Ethereum can use this data to reconstruct the current state of Starknet.

[NOTE]

====

To update the Starknet state on L1, you only need to send a valid proof along with the state difference, and there is no need to include additional details, such as transactions and events.

Therefore, depending on the use case, you might need more information to track Starknet's state.

====

== Data availability: EIP-4844, Starknet 0.13.1 and forward

Starting with Starknet version 0.13.1, the sequencer determines whether to publish the state difference on Ethereum as calldata or blobdata. In extreme situations where blob prices significantly exceed those of calldata, the Starknet sequencer can switch to publish the state diff as calldata.

Under normal conditions, blobs are the default method for publishing Starknet's state differences.

The format for state diffs remains the same as in version 0.11.0, but the data sent to Ethereum is a Fast Fourier Transform (FFT) of the original data. To recover Starknet's state diff based on blobs published onchain, you must first perform an Inverse Fast Fourier Transform (IFFT) on the raw blob, and then proceed with decoding according to the format described below.

.Additional resources

* <https://community.starknet.io/t/data-availability-with-eip4844/>[Data availability with EIP-4844] on the Starknet Community Forum

* <https://etherscan.io/tx/0x8a227491bc78424c2cac1b203c95cdd99ede5112d41f0e7eab26f3c8aa9c658d/>[An

example blob published on Ethereum by the Starknet sequencer in a state update transaction]

== Data availability: v0.11.0 and forward

[id="v0.11.0_format"]

=== v0.11.0 format

The state diffs contain information on every contract whose storage was updated and additional information on contract deployments.

For each affected contract, the following information is sent as calldata on L1:

* The contract address

* A single 32-byte word that includes the nonce and the following meta information about the update:

+

** class information flag, whose value is one of the following:

+

[horizontal,labelwidth="2"]

`0`:: Storage updates only.

`1`:: The contract was deployed or replaced in this state update.

+

When this flag is set to 1, the new class hash occupies an additional word before the storage updates section.

** number of storage updates

+

The expected format of this 32-byte word is as follows:

+

[stem]

++++

$\underbrace{0 \cdots 0}_{\text{127 bits}}$

$\underbrace{\text{class information flag}}_{\text{1 bit}}$

$\underbrace{\text{new nonce}}_{\text{64 bits}}$

$\underbrace{\text{\# of storage updates}}_{\text{64 bits}}_{\text{LSB}}$

++++

Each storage update includes the following:

- * key - the address inside the contract's storage where the value is updated
- * value - the new value

Newly declared classes include the following:

- * The number of Cairo classes that were declared in the block
- * Each class includes the following:
 - ** The class hash
 - ** xref:smart-contracts/class-hash.adoc[The compiled class hash]

[id="v0.11.0_example"]
=== v0.11.0 example of onchain data

Consider the following onchain data that was extracted from L1:

```
[source,json]
----
[
  1, <1>
  201917239009505132386904748107510200373124613299705751896592797910141
3600827, <2>
  18446744073709551617, <3>
  100, <4>
  200, <4>
  1, <5>
  135114824264500554000416253155080507699574774608754203009518655753664
1755046, <6>
  558404273560404778508455254030458021013656352466216690688595011803280
448032 <7>
]
```

-
- <1> The number of contracts whose state was updated.
 - <2> The address of the first, and only, contract whose state changed.
 - <3> `18446744073709551617`, which is $2^{64}+1$, encodes the following:
 - * The class information flag is `0`, that is, the contract was not deployed or replaced just now, so you shouldn't treat the next word as the class hash.
 - * The new nonce is `1`.
 - * One storage cell was updated.
 - <4> These two elements, `100` and `200`, encode the storage update, where the value of key `100` is set to `200`.
 - <5> The new declare section: `1` includes a single xref:network-architecture/transactions.adoc#declare_v2[declare v2] transaction in this state update.
 - <6> Encoding of xref:../smart-contracts/class-hash.adoc[the class hash].
 - <7> Encoding of the compiled class hash of the declared class.

== Data availability: pre v0.11.0

[id="pre_v0.11.0_format"]

=== Pre v0.11.0 format

The state diffs contain information on every contract whose storage was updated and additional information on contract deployments. Those differences are sent as `uint256[]` array as part of the calldata, and are encoded as follows:

- * Number of cells that encode contract deployments
- * Each deployed contract has the following:

** `contract_address` - the xref:smart-contracts/contract-address[address] of the deployed contract. See also xref:network-architecture/data-availability.adoc[Data availability].

** `contract_hash` - the xref:../smart-contracts/class-hash.adoc[hash] of the class

- * Number of contracts whose storage is updated.

+

Each such contract has the following:

** `contract_address` - the xref:../network-architecture/data-availability.adoc[address] of the contract

** `num_of_storage_updates` - number of storage updates

** `nonce, num of storage updates` - a `uint256` value that encodes both the number of storage updates for that contract and the updated nonce:

+

[stem]

++++

$\underbrace{0\cdots0}_{\text{128 bits}} \mid \underbrace{\text{new nonce}}_{\text{64 bits}} \mid \underbrace{\text{\# of storage updates}}_{\text{64 bits}}_{\text{LSB}}$

++++

+

For each storage update:

*** `key` - the address inside the contract's storage where the value is updated

*** `value` - the new value

[id="pre_v0.11.0_example"]

=== Pre v0.11.0 example

The example below shows onchain data that was extracted from L1. An explanation follows, according to the above format.

[source,json]

[
2,
247293930732837103945597765099422640702460775406356299385622407725459
4995194,
133604347792591060217542962755536955126222971226621788748152964265090
7574765,
5,
201917239009505132386904748107510200373124613299705751896592797910141
3600827,
18446744073709551617,
5,
102,
211115821442973626010179745381534126565851611842138731485062553590511
5418634,
2,
619473939880410191267127038055308002651079521370507951329266275707625
062498,
147158405518488970147150712956737660766678552245547639413077443475441
1633091,
619473939880410191267127038055308002651079521370507951329266275707625
062499,
541081937647750334353499719661793404023294520617957763260656728924567
461866,
247293930732837103945597765099422640702460775406356299385622407725459
4995194,
1,
955723665991825982403667749532843665052270105995360175183368988948217
233556,
243927228903233004188542777391602139092690345091709731780746808295858
1062272,
342931971350305439924375172853234950048909644418186764022880923399399
2987070,
1,
5,
1110,
347613889183800112861470455373196471063423858754180349900182232260242
1164873,
6,
596640152862911255867271811870458495289302987417286399586140765893748
75456,
600,
221246409693049874911156614478125967098431447433028390043893900771521
609973,
400,
558404273560404778508455254030458021013656352466216690688595011803280
448030,

```

100,
558404273560404778508455254030458021013656352466216690688595011803280
448031,
200,
558404273560404778508455254030458021013656352466216690688595011803280
448032,
300,
135114824264500554000416253155080507699574774608754203009518655753664
1755046,
500
]
----
```

* The first element, `2`, is the number of cells that encode contracts deployment.
 * The next two elements describe a single contract deployment with the following parameters:

** `contract_address`:

```

+
----
2472939307328371039455977650994226407024607754063562993856224077254594
995194
----
```

** `contract_hash`:

```

+
----
1336043477925910602175429627555369551262229712266217887481529642650907
574765
----
```

* The next element, `5` (index 3 in the array), is the number of contracts whose storage was updated. We will take only the first contract as an example.

** `contract_address`:

```

+
----
2019172390095051323869047481075102003731246132997057518965927979101413
600827
----
```

** Following the above contract address, we have `18446744073709551617` (index 8 in the array), which is stem:[\$2^{64}+1\$], thus:

*** The new contract nonce is `1`

*** One storage key is updated

*** The value at key `5` was changed to `102`

The next 4 contract storage updates are interpreted in the same manner.

[id="extract_from_ethereum"]
== Extract from Ethereum

The data described above is sent across several Ethereum transactions, each holding a part of this array as calldata. Each new Starknet block has its associated state diff transactions.

You can find the code for extracting this data from Ethereum in https://github.com/eqlabs/pathfinder/blob/2fe6f549a0b8b9923ed7a21cd1a588bc571657d6/crates/pathfinder/src/ethereum/state_update/retrieve.rs [Pathfinder's repo]. Pathfinder is the first Starknet full node implementation. You may also take a look at the https://github.com/eqlabs/pathfinder/blob/2fe6f549a0b8b9923ed7a21cd1a588bc571657d6/crates/pathfinder/resources/fact_retrieval.py [Python script] which extracts the same information.

[id="gas-and-transaction-fees"]
= Gas and transaction fees
:--auto-ids:

This section describes fees that are paid on L2 starting in Starknet 0.13.0. For information about messaging fees that are paid on L1, see [xref:network-architecture/messaging-mechanism.adoc#l1-l2-message-fees](#) [L1 !' L2 message fees].

[#overall_fee]
== Overall transaction fee

Starting with Starknet v0.13.1, Starknet distinguishes between blocks whose state diffs are sent to L1 as calldata and blocks whose state diffs are sent to L1 as blobs. The `l1_da_mode` property in the Starknet block header contains this information. The cost of computation remains the same on both options, but the cost related to data availability differs.

[#overall_fee_blob]
=== Overall transaction fee with blobs

This section shows the formula for determining a transaction's fee. The following sections describe how this formula was derived.

The following formula describes the overall fee, F , for a transaction:

[stem]
++++
$$F = \text{gas_price} \cdot \left(\max_k v_k w_k + \text{message_calldata_cost} + \text{l1_log_data_cost} \cdot \sum_{i=1}^t q_i \right) + \dots$$

```

& \quad + \; \left(\text{l1\_storage\_write\_cost}+\text{log\_message\_to\_l1\_cost}\right)\cdot
t + \; \backslash\backslash
& \quad + \; \text{l2\_payload\_costs}\Bigg) + \backslash\backslash
& \text{data\_gas\_price}\cdot\text{felt\_size\_in\_bytes}\cdot\text{bigg}(2(n-1)+2(m-1) + \ell + 2D
\text{bigg})
\end{align}
++++

```

where:

* $\text{stem}:[\$v\$]$ is a vector that represents resource usage, where each of its entries, $\text{stem}:[\$v_k\$]$, corresponds to different resource types: Cairo steps and number of applications of each builtin.

+

For more information see [xref:#calculation_of_computation_costs](#)[Calculation of computation costs].

* $\text{stem}:[\$w\$]$ is the

[xref:#calculation_of_computation_costs](#)[`CairoResourceFeeWeights`] vector.

* $\text{stem}:[\$n\$]$ is [xref:#storage_updates](#)[the number of unique contracts updated], which also includes changes to classes of existing contracts and contract deployments, even if the storage of the newly deployed contract is untouched. In other words, $\text{stem}:[\$n\geq \ell\$]$. Notice that $\text{stem}:[\$n\geq 1\$]$ always holds, because the fee token contract is always updated, which does not incur any fee.

* $\text{stem}:[\$m\$]$ is the number of values updated, not counting multiple updates for the same key. Notice that $\text{stem}:[\$m\geq 1\$]$ always holds, because the sequencer's balance is always updated, which does not incur any fee.

* $\text{stem}:[\$t\$]$ is the number of L2->L1 messages sent, where the corresponding payload sizes are denoted by $\text{stem}:[\$q_1, \dots, q_t\$]$.

* $\text{stem}:[\$ell\$]$ is the number of contracts whose class was changed, which happens on contract deployment and when applying the `replace_class` syscall.

* $\text{stem}:[\$D\$]$ is 1 if the transaction is of type `DECLARE` and 0 otherwise. Declare transactions need to post on L1 the new class hash and compiled class hash which are added to the state.

* L2->L1 messaging constants:

+

--

** $\text{stem}:[\$ \text{message_calldata_cost} \$]$ is 1124 gas per 32-byte word.

** $\text{stem}:[\$ \text{l1_log_data_cost} \$]$ is 256 gas.

** $\text{stem}:[\$ \text{l1_storage_write_cost} \$]$ is the cost of writing to a new storage slot on Ethereum, which is 20,000 gas.

** $\text{stem}:[\$ \text{log_message_to_l1_cost} \$]$ is 1637 gas.

--

+

For more information, see [xref:#l2-l1_messages](#)].

* $\text{stem}:[\$ \text{l2_payload_costs} \$]$ is the gas cost of data sent over L2. This includes calldata, code, and event emission. For more details see [xref:#l2_calldata](#)].

* stem:[\$\text{felt_size_in_bytes}\\$] is 32, which is the number of bytes required to encode a single STARK field element.

[#overall_fee_calldata]

=== Overall transaction fee with calldata

This section shows the formula for determining a transaction's fee. The following sections describe how this formula was derived.

The following formula describes the overall fee, stem:[F], for a transaction:

[stem]

++++

\begin{align}

$F = \text{gas_price} \cdot \Big(\max_k v_k w_k + \backslash$

$\& + \backslash; \text{da_calldata_cost} \left(2(n-1) + 2(m-1) + \backslashell + 2D + 3t + \sum\limits_{i=1}^t q_i \right) \backslash$

$\& - \backslash; \text{contract_update_discount} \cdot (n-1) - 240 \backslash$

$\& + \backslash; \text{message_calldata_cost} \cdot 3t + (\text{message_calldata_cost} +$

$\text{l1_log_data_cost}) \cdot \sum\limits_{i=1}^t q_i \backslash$

$\& + \backslash; \left(\text{l1_storage_write_cost} + \text{log_message_to_l1_cost} \right) \cdot t \backslash$

$\& + \backslash; \text{l2_payload_costs} \Big)$

\end{align}

++++

where:

* The following constants are defined in the same manner as in the blob-based formula:

** stem:[\$v, w, n, m, t, \backslashell, D\$]

** stem:[\$\text{message_calldata_cost}, \backslash; \text{l1_log_data_cost}, \backslash;

$\text{log_message_to_l1_cost}, \backslash; \text{l1_storage_write_cost}\$]$

** stem:[\$\text{l2_payload_costs}\\$]

* stem:[\$\text{da_calldata_cost}\\$] is 551 gas per 32-byte word. This cost is derived as follows:

+

** 512 gas per 32-byte word for calldata.

** ~100 gas for onchain hashing that happens for every sent word.

** a 10% discount, because the sequencer does not incur additional costs for repeated updates to the same storage slot within a single block.

* stem:[\$240\$] is the gas discount for updating the sender's balance, for the derivation of this number see xref:#storage_updates[].

* stem:[\$\text{contract_update_discount}\\$] is 312 gas, for the derivation of this discount see xref:#storage_updates[].

== When is the fee charged?

The fee is charged atomically with the transaction execution on L2. The Starknet OS injects a transfer of the fee-related ERC-20, with an amount equal to the fee paid, the sender equal to the transaction submitter, and the sequencer as a receiver.

[#fee_limitations]

== Transaction Fee limits

[#v3_fee_limitations]

=== v3 transactions

With v3 transactions, users specify the max amount and max price for each resource. At the time of writing, the only available resource is L1 gas. In the future, we will introduce L2 gas which will be used to price L2 work (as opposed to only charging for the proof verification in L1 gas, which is what happens today).

[#deprecated_fee_limitations]

=== Deprecated transactions (version < 3)

With older transaction versions, users specify the maximum fee that they are willing to pay for a transaction.

The only limitation on the sequencer, which is enforced by the Starknet OS, is that the actual fee charged is bounded by `max_fee`. While not enforced in the proof, the Starknet sequencer usually charges less than `max_fee`, as it charges in accordance with the above fee formula.

[#what_do_we_pay_for]

== What do we price

At the time of writing, the following components are contributing to the transaction fee:

- * xref:#computation_without_builtins[Computational complexity]: The marginal cost of verifying the transaction on L1, measured in L1 gas.

- * xref:#onchain_data_components[Onchain data]: The cost of posting the state diffs induced by the transaction to L1 (for more details, see xref:network-architecture/data-availability.adoc[data availability]). This is measured in L1 gas or L1 data gas, depending on whether or not the L2 block in which the transaction was included uses calldata or blobs.

- * L2!L1 messages: Messages sent to L1 are eventually sent to the Starknet core contract as L1 calldata by the sequencer; therefore L2 transaction that send L2->L1 messages incur an additional L1 gas cost.

- * L2 calldata, events and code: From Starknet 0.13.1 onwards, there is a per-byte (or per felt) price for L2 payloads. For more details, see xref:#l2_calldata[].

== Fee units

Each transaction is associated with an estimate of the amount of gas used. Combining this estimate with the price of gas yields the estimated fee.

For transactions prior to v3, the fee is denominated in WEI. For transactions v3 and later, the fee is denominated in STRK.

```
[#fee_calculation]
== Fee calculation
```

```
[#calculation_of_gas_costs]
=== Calculation of gas prices
```

Every 60 seconds, Starknet samples the base price of gas and data gas on L1.

The price of gas on Starknet is set to the average of the last 60 gas price samples, plus 1 gwei.

The price of data gas on Starknet is set to the average of the last 60 data gas price samples. The data gas price on Ethereum is derived from the value of ``excess_blob_gas``. For more information, see link:<https://github.com/ethereum/EIPs/blob/master/EIPS/eip-4844.md>[EIP-4844].

```
[#computation_without_builtins]
=== Computation without builtins
```

Let's analyze the correct metric for measuring transaction complexity. For simplicity, we will ignore Cairo's builtins, and address them later.

A Cairo program execution yields an execution trace. When proving a Starknet block, we aggregate all the transactions appearing in that block to the execution trace.

Starknet's prover generates proofs for execution traces, up to some maximal length $\text{stem}:[L]$, derived from the specs of the proving machine and the desired proof latency.

Tracking the execution trace length associated with each transaction is simple. Each assertion over field elements, such as verifying addition/multiplication over the field, requires the same, constant number of trace cells, which is where our "no-builtins" assumption kicks in: Pedersen occupies more trace cells than addition. Therefore, in a world without builtins, the fee of the transaction $\text{stem}:[tx]$ is correlated with $\text{stem}:[\text{TraceCells}[tx]/L]$.

```
[#computation_with_builtins]
=== Computation with builtins
```

In the Cairo execution trace each builtin has its own slot, which is important to consider

when determining the fee.

For example, consider that the prover can process a trace with the following limits:

[%autowidth]

|===

| up to 500,000,000 Cairo Steps | up to 20,000,000 Pedersen hashes | up to 4,000,000 signature verifications | up to 10,000,000 range checks

|===

The proof is closed and sent to L1 when any of these slots is filled.

Suppose that a transaction uses 10,000 Cairo steps and 500 Pedersen hashes. At most 40,000 such transactions can fit into the hypothetical trace ($20,000,000/500$). Therefore, its gas price correlates with $1/40,000$ of the cost of submitting proof.

Notice that this estimate ignores the number of Cairo steps, as it is not the limiting factor, since $500,000,000/10,000 > 20,000,000/500$.

With this example in mind, it is possible to formulate the exact fee associated with L2 computation.

[IMPORTANT]

=====

The allocation of resources among builtin operations must be predetermined; it is not possible to decide, post-execution, to include only 20,000,001 Pedersen hashes without additional components.

This safeguards fairness and prevents manipulation, ensuring integrity in proof generation and fee determination.

=====

[#calculation_of_computation_costs]

=== Calculation of computation costs

For each transaction, the sequencer calculates a vector, `CairoResourceUsage`, that contains the following:

- * The number of Cairo steps.
- * The number of applications of each Cairo builtin. For example, five range checks and two Pedersen hashes.

The sequencer crosses this information with the `CairoResourceFeeWeights` vector. For each resource type, either a Cairo step or a specific builtin application, `CairoResourceFeeWeights` has an entry that specifies the relative gas cost of that

component in the proof.

Going back to the above example, if the cost of submitting a proof with 20,000,000 Pedersen hashes is roughly 5m gas, then the weight of the Pedersen builtin is 0.25 gas per application (5,000,000/20,000,000). The sequencer has a predefined weights vector, in accordance with the proof parameters.

The sequencer charges only according to the limiting factor. Therefore the fee is correlated with:

[stem]
++++
$$\max_k [\text{CairoResourceUsage}_k \cdot \text{CairoResourceFeeWeights}_k]$$

++++

where stem:[k] enumerates the Cairo resource components, that is the number of Cairo steps and builtins used.

The weights are listed in the table [xref:#gas_cost_per_cairo_step_or_builtin_step\[\]](#).

[#gas_cost_per_cairo_step_or_builtin_step]
.Amount of gas used per Cairo step or per each time a Cairo builtin is applied

[width=80%,cols="1,2",options="header",stripes=even]

|===

| Step or builtin | Gas cost

| Cairo step | 0.0025 gas/step
| Pedersen | 0.08 gas/application
| Poseidon | 0.08 gas/application
| Range check | 0.04 gas/application
| ECDSA | 5.12 gas/application
| Keccak | 5.12 gas/application
| Bitwise | 0.16 gas/application
| EC_OP | 2.56 gas/application
|===

[id="onchain_data_components"]

=== Onchain data components

The onchain data associated with a transaction is composed of three parts

- * Storage updates
- * L2!L1 messages
- * Deployed contracts
- * Declared classes (only relevant for `DECLARE` transactions, and adds two additional

words)

[#storage_updates]

=== Onchain data: Storage updates

Whenever a transaction updates some value in the storage of some contract, the following data is sent to L1:

- * two 32-byte words per contract
- * two 32-byte words for every updated storage value

For information on the exact data and its construction, see [xref:architecture-and-concepts:network-architecture/data-availability.adoc#v0.11.0_format\[Data availability\]](#).

[NOTE]

=====

Only the most recent value reaches L1. So the transaction's fee only depends on the number of `_unique_` storage updates. If the same storage cell is updated multiple times within the transaction, the fee remains that of a single update.

=====

The following formula describes the storage update fee for a transaction:

[stem]

++++

$$\underbrace{\text{gas_price} \cdot \left(\text{da_calldata_cost} \cdot 2^{(n-1)} - \text{contract_update_discount} \cdot (n-1) \right)}_{\text{contract addresses + new nonce and number of storage updates}}$$

+ \\

$$\underbrace{\text{gas_price} \cdot \left(\text{da_calldata_cost} \cdot (2^{(m-1)} - 240) \right)}_{\text{storage updates}}$$

++++

where:

* `stem:[n]` is [xref:#storage_updates](#)[the number of unique contracts updated], which also includes changes to classes of existing contracts and contract deployments, even if the storage of the newly deployed contract is untouched. In other words, `stem:[$n \geq 1$]`. Notice that `stem:[$n \geq 1$]` always holds, because the fee token contract is always updated at the end of each transaction, in order to update the sequencer's and the sender's balances. The fee token contract update is not taken into account when computing the fee.

* `stem:[m]` is the number of values updated, not counting multiple updates for the

same key. Notice that $\text{stem}:[m \geq 1]$ always holds, because the sequencer's balance is updated at the end of each transaction. The sequencer's balance update is not taken into account when computing the fee.

* $\text{stem}:[\text{contract_update_discount}]$ is 312 gas, which is discounted for every updated contract. This discount is a result of the fact that out of the $\text{stem}:[2n]$ words caused by updating contracts, $\text{stem}:[n]$ words are short, including at most 6 non-zero bytes:

+

--

** three bytes for the nonce

** two bytes for the number of storage updates

** one byte for the class information flag

--

+

Taking into account that zero bytes only cost 4 gas, the cost difference between a full 32-byte word, which does not contain zeros, and a word with only six non-zero bytes is $\text{stem}:[32 \cdot 16 - (6 \cdot 16 + 26 \cdot 4) = 312]$.

* $\text{stem}:[240]$ is the gas discount for updating the sender's balance, and is derived by assuming the balance requires at most 12 non-zero bytes, which is enough for 1.2B ETH or STRK, resulting in the following discount: $\text{stem}:[512 - (20 \cdot 4 + 12 \cdot 16) = 240]$.

[NOTE]

====

Improvements to the above pessimistic estimation might be gradually implemented in future versions of Starknet.

For example, if different transactions within the same block update the same storage cell, there is no need to charge for both transactions, because only the last value reaches L1. In the future, Starknet might include a refund mechanism for such cases.

====

[#l_2-l_1_messages]

=== Onchain data: L2->L1 messages

When a transaction that raises the `send_message_to_l1` syscall is included in a state update, the following data reaches L1:

* L2 sender address

* L1 destination address

* Payload size

* Payload (list of field elements)

Consequently, the gas cost associated with a single L2!L1 message is:

```

[stem]
++++
\begin{align}
\text{MESSAGE\_COST} = & \; \; \; \text{message\_calldata\_cost} \cdot \left(3 + \right. \\
& \; \; \; \text{payload\_size} \cdot \left. \right) \; + \; \\
& \; \; \; \text{l1\_log\_data\_cost} \cdot \text{payload\_size} \; + \; \\
& \; \; \; \text{log\_message\_to\_l1\_cost} \; + \; \\
& \; \; \; \text{l1\_storage\_write\_cost} \\
\end{align}
++++

```

Where:

* stem:[\$\text{message_calldata_cost}\$] is 1124 gas. This is the sum of the 512 gas paid to the core contract on submitting the state update, and 612 gas paid for the submitting of the same word to the verifier contract (which incurs ~100 additional gas for hashing). That is, messages are sent to Ethereum twice.

* stem:[\$\text{log_message_to_l1_cost}\$] is 1637 gas. This is the fixed cost involved in emitting a `LogMessageToL1` event. This event has two topics and a data array, which adds two data words to the event, resulting in a total of stem:[\$375+2 \cdot 375+2 \cdot 256\$] gas (log opcode cost, topic cost, and two data words cost).

* stem:[\$\text{l1_log_data_cost}\$] is 256 gas, which is paid for every payload element during the emission of the `LogMessageToL1` event.

* stem:[\$\text{l1_storage_write_cost}\$] is 20,000 gas per message which is paid in order to store the message hash on the Starknet core contract. This recording of the message is what later enables the intended L1 contract to consume the message.

[#deployed_contracts]

=== Onchain data: Deployed contracts

When a transaction that raises the `deploy` syscall is included in a state update, the following data reaches L1:

- * contract address
- * number of storage updates and the new nonce
- * class hash

The first two elements are counted in the number of unique modified contracts, denoted by stem:[\$n\$] throughout this page. So the only additional word comes from publishing the class hash, which adds 551 gas. For more information, see stem:[\$\text{da_calldata_cost}\$] in the xref:#overall_fee[final formula].

[#l2_calldata]

=== L2 payloads: calldata, events, and code

As of Starknet v0.13.1 onwards, L2 data is taken into account during pricing. This

includes:

- * calldata: this includes transaction calldata (in the case of `INVOKE` transactions or `L1_HANDLER`), constructor calldata (in the case of `DEPLOY_ACCOUNT` transactions), and signatures
- * events: data and keys of emitted events
- * ABI: classes abi in `DECLARE` transactions (relevant only for `DECLARE` transactions of version "e 2")
- * CASM bytecode (for all available `DECLARE` transactions, where in version "e 2" this refers to the compiled class)
- * Sierra bytecode (relevant only for `DECLARE` transactions of version "e 2")

The pricing of the above components in terms of L1 gas is given by the following table:

===	
Resource	Gas cost
Event key	0.256 gas/felt
Event data	0.128 gas/felt
Calldata	0.128 gas/felt
CASM bytecode	1 gas/felt
Sierra bytecode	1 gas/felt
ABI	0.032 gas/character
===	
[id="messaging_mechanism"]	
= L1-L2 messaging mechanism	

Starknet's ability to interact with L1 is crucial. `_Messaging_` is the mechanism that enables this interaction.

For example, you can perform computations on L2 and use the result on L1.

Bridges on Starknet use the L1-L2 messaging mechanism. Consider that you want to bridge tokens from Ethereum to Starknet. You deposit your tokens in the L1 bridge contract, which automatically triggers the minting of the same token on L2. Another good use case for L1-L2 messaging is Defi pooling. For more information, see [link:https://starkware.co/resource/defi-pooling/](https://starkware.co/resource/defi-pooling/) [DeFi pooling] on StarkWare's site and [link:https://www.starknet.io/en/ecosystem/dapps/](https://www.starknet.io/en/ecosystem/dapps/) [dApps] on <https://www.starknet.io>.

Be aware that the messaging mechanism is `_asynchronous_` and `_asymmetric_`.

- * `_Asynchronous_`: Your contract code, whether Cairo or Solidity, cannot await the result of the message being sent on the other layer within your contract code's execution.
- * `_Asymmetric_`: Sending a message from Ethereum to Starknet, L1->L2, is fully automated by the Starknet sequencer, so the message is automatically delivered to the

target contract on L2. However, when sending a message from Starknet to Ethereum, L2->L1, the sequencer only sends the hash of the message. You must then consume the message manually using a transaction on L1.

```
[id="l2-l1_messages"]  
== L2 -> L1 messages
```

Contracts on L2 can interact asynchronously with contracts on L1 using the L2->L1 messaging protocol.

The protocol consists of the following stages:

- . During the execution of a transaction, a contract on Starknet sends a message from L2 to L1 by calling the `send_message_to_L1` syscall.

- . The sequencer attaches the message parameters to the block that includes the syscall invocation. The message parameters include the address of the sender on L2, the address of the recipient contract on L1, and the message data.

+

For example:

+

```
[source,cairo]
```

```
----
```

```
let mut payload: Array<felt252> = ArrayTrait::new();
```

```
let to_address: EthAddress = 1_felt252.try_into().unwrap();
```

```
payload.append(1);
```

```
// potentially add more elements to payload (payload[1], payload[2], etc.)
```

```
send_message_to_l1_syscall(to_address: to_address.into(), payload: payload.span());
```

```
----
```

- . The prover proves the state update that includes this transaction.

- . The sequencer updates the L1 state.

- . The message is stored on L1 in the Starknet Core Contract and a counter on the Core Contract increases by one. +

- . The `processMessage` function, which is part of the Starknet Core Contract, emits the `LogMessageToL1` event, which contains the message parameters.

- . The message recipient on L1 can access and consume the message by calling the

link:[https://github.com/starkware-libs/cairo-lang/](https://github.com/starkware-libs/cairo-lang/blob/4e233516f52477ad158bc81a86ec2760471c1b65/src/starkware/starknet/eth/StarknetMessaging.sol#L119)

[blob/4e233516f52477ad158bc81a86ec2760471c1b65/src/starkware/starknet/eth/StarknetMessaging.sol#L119](https://github.com/starkware-libs/cairo-lang/blob/4e233516f52477ad158bc81a86ec2760471c1b65/src/starkware/starknet/eth/StarknetMessaging.sol#L119) `consumeMessageFromL2` function, which includes the message parameters within the transaction.

This function, which is part of the Starknet Core Contract, verifies the following:

- * The hashes of the L2 sent message parameters, now stored on the Core Contract, and the L1 received message parameters, are the same.

- * The entity calling the function is indeed the recipient on L1.

+

// We need to separate out these functions into a reference.
 In such a case, the counter corresponding to the message hash in the Starknet Core Contract decreases by one. For more information, see the link:[https://github.com/starkware-libs/cairo-lang/blob/4e233516f52477ad158bc81a86ec2760471c1b65/src/starkware/starknet/eth/StarknetMessaging.sol#L130C7-L130C7#\[`consumeMessageFromL2`\] function in `StarknetMessaging.sol`](https://github.com/starkware-libs/cairo-lang/blob/4e233516f52477ad158bc81a86ec2760471c1b65/src/starkware/starknet/eth/StarknetMessaging.sol#L130C7-L130C7#[`consumeMessageFromL2`] function in `StarknetMessaging.sol`).

xref:#diagram_l2-l1_messaging_mechanism[] illustrates this flow:

[#diagram_l2-l1_messaging_mechanism]
 .L2->L1 Messaging mechanism
 image::l2l1.png[L2->L1 message mechanism]

=== L2 -> L1 message structure

// xref:#structure_l2-l1[] illustrates the structure of an L2 -> L1 message.

The structure of an L2 -> L1 message is described as follows under `MSG_TO_L1` in the link:https://github.com/starkware-libs/starknet-specs/blob/master/api/starknet_api_openrpc.json [Starknet API JSON RPC] specification:

[horizontal,labelwidth="30",role="stripes-odd"]
 `from_address` (`felt252`):: The address of the L2 contract sending the message.
 `to_address` (`EthAddress`):: The target L1 address the message is sent to.
 `payload` (`Array<felt252>`):: The payload of the message.

[#hashing_l2-l1]
 === L2 -> L1 message hashing

The hash of an L2 -> L1 message is computed on L1 as follows:

```
[source,js]
----
keccak256(
  abi.encodePacked(
    FromAddress,
    uint256(ToAddress),
    Payload.length,
    Payload
  )
);
----
```

[NOTE]
 =====

Sending an L2 to L1 message always incurs a fixed cost of 20,000 gas, because the hash of the message being sent must be written to L1 storage in the Starknet Core Contract.

====

[id="l1-l2-messages"]
== L1 -> L2 messages

Contracts on L1 can interact asynchronously with contracts on L2 using the `_L1->L2_` messaging protocol.

The protocol consists of the following stages:

- . An L1 contract induces a message to an L2 contract on Starknet by calling the link:<https://github.com/starkware-libs/cairo-lang/blob/54d7e92a703b3b5a1e07e9389608178129946efc/src/starkware/starknet/solidity/IStarknetMessaging.sol#L13> [`sendMessageToL2``] function on the Starknet Core Contract with the message parameters.

+

The Starknet Core Contract hashes the message parameters and updates the L1->L2 message mapping to indicate that a message with this hash was indeed sent. The L1 contract records the fee that the sender paid. For more information, see `xref:#l1-l2-message-fees` [L1 -> L2 message fees].

- . The message is then decoded into a Starknet transaction that invokes a function annotated with the ``l1_handler`` decorator on the target contract. Transactions like this on L2 are called `xref:#l1_handler_transaction` [L1 handler transactions].

- .. The Starknet sequencer, upon receiving enough L1 confirmations for the transaction that sent the message, initiates the corresponding L2 transaction.

- .. The L2 transaction invokes the relevant ``l1_handler`` function.

- . The L1 Handler transaction that was created in the previous step is added to a proof.

- . The Core Contract receives the state update.

- . The message is cleared from the Core Contract's storage to consume the message.

Clearing the Core Contract's storage does the following:

+

- * incurs a fixed cost of 5,000 gas

- * emits an L1 event logging the message consumption

At this point, the message is handled.

// The above flow is illustrated in the following diagram:

// #THIS IMAGE IS WRONG & MISLEADING#

// image::l1l2.png[l1l2]

An L1->L2 message consists of the following:

- * L1 sender's address
- * L2 recipient's contract address
- * Function selector
- * Calldata array
- * Message nonce

+

[NOTE]

====

The message nonce is maintained on the Starknet Core Contract on L1, and is incremented whenever a message is sent to L2. The nonce is used to avoid a hash collision between different L1 handler transactions that is caused by the same message being sent on L1 multiple times.

For more information, see `xref:#l1_l2_message_structure[L1->L2 structure]`.

====

[id="l2-l1_message_cancellation"]

== L1 -> L2 message cancellation

[NOTE]

====

The flow described here should only be used in edge cases such as bugs on the Layer 2 contract preventing message consumption.

====

Consider that Alice sends an L1 asset to a Starknet bridge to transfer it to L2, which generates the corresponding L1->L2 message. Now, consider that the L2 message consumption doesn't function, which might happen due to a bug in the dApp's Cairo contract. This bug could result in Alice losing custody of their asset forever.

To mitigate this risk, the contract that initiated the L1->L2 message can cancel it by declaring the intent to cancel, waiting five days, and then completing the cancellation. This delay protects the sequencer from a DoS attack in the form of repeatedly sending and canceling a message before it is included in L1, rendering the L2 block which contains the activation of the corresponding L1 handler invalid.

The steps in this flow are as follows:

- . The user that initiated the L1->L2 message calls the `https://github.com/starkware-libs/cairo-lang/blob/4e233516f52477ad158bc81a86ec2760471c1b65/src/starkware/starknet/eth/StarknetMessaging.sol#L134[`startL1ToL2MessageCancellation`]` function in the Starknet Core Contract.`
- . The user waits five days until she can finalize the cancellation.
- . The user calls the `https://github.com/starkware-libs/cairo-lang/blob/4e233516f52477ad158bc81a86ec2760471c1b65/src/starkware/starknet/eth/StarknetMessaging.sol#L147[`cancelL1ToL2Message`]` function.`

```
[id="l1-l2-message-fees"]  
=== L1 -> L2 message fees
```

An L1 -> L2 message induces a transaction on L2, which, unlike regular transactions, is not sent by an account. This calls for a different mechanism for paying the transaction's fee, for otherwise the sequencer has no incentive of including L1 handler transactions inside a block.

To avoid having to interact with both L1 and L2 when sending a message, L1 -> L2 messages are payable on L1, by sending ETH with the call to the payable function ``sendMessageToL2`` on the Starknet Core Contract.

The sequencer takes this fee in exchange for handling the message. The sequencer charges the fee in full upon updating the L1 state with the consumption of this message.

The fee itself is calculated in the [xref:architecture-and-concepts:network-architecture/fee-mechanism.adoc#overall_fee](#) [same manner] as "regular" L2 transactions. You can use the [xref:cli:starkli.adoc#starknet-estimate_fee\[CLI\]](#) to get an estimate of an L1 -> L2 message fee.

```
[id="structure_and_hashing_l1-l2"]  
[#l1_l2_message_structure]  
=== L1 -> L2 structure
```

For completeness, [xref:#l1_l2_message_structure\[\]](#) describes the precise structure of both the message as it appears on L1 and the induced transaction as it appears on L2.

```
[#L1-L2_message_structure]  
.L1 -> L2 message structure  
[%autowidth.stretch]  
|===  
| FromAddress      | ToAddress      | Selector      | Payload      | Nonce      |  
| `EthereumAddress` | `FieldElement` | `FieldElement` | `List+++<FieldElement>+++` |  
| `FieldElement`   |  
|===
```

```
[#hashing_l1-l2]  
=== L1 -> L2 hashing
```

The hash of the message is computed on L1 as follows:

```
[source,js]
```

```

----
keccak256(
    abi.encodePacked(
        uint256(FromAddress),
        ToAddress,
        Nonce,
        Selector,
        Payload.length,
        Payload
    )
);
----

```

```

[id="l1_handler_transaction"]
=== L1 handler transaction

```

```

[%autowidth.stretch]
|===
| Version      | ContractAddress | Selector      | Calldata      | Nonce      |
| `FieldElement` | `FieldElement` | `FieldElement` | `List+++<FieldElement>+++` | `FieldElement` |
|===

```

The hash of the corresponding L1 handler transaction on L2 is computed as follows:

```

[source,cairo]
----
l1_handler_tx_hash = _h_(
    "l1_handler",
    version,
    contract_address,
    entry_point_selector,
    _h_(calldata),
    0,
    chain_id,
    nonce
)
----

```

Where:

- * `l1_handler` is a constant prefix, encoded in bytes (ASCII), as big-endian.
- * `version` is the transaction version. Only version 0 is currently supported.
- * `chain_id` is a constant value that specifies the network to which this transaction is sent.

* `_h_` is the `xref:architecture-and-concepts:cryptography/hash-functions.adoc#pedersen_hash[Pedersen]` hash (note that since we're hashing an array, the `#` of inputs needs to be appended to the hash).
* ``0`` indicates that L1 to L2 message fees are charged on L1.

[NOTE]

=====

In an L1 handler transaction, the first element of the `calldata` is always the sender's Ethereum address.

=====

[NOTE]

=====

Since L1 handler transactions are not initiated by an account, invoking `link:https://github.com/starkware-libs/cairo/blob/2203a47f8a098cd4718d03bd109ca014049419e7/corelib/src/starknet/info.cairo#L49[get_caller_address()]` or similar account-related functions returns the address ``0x0``.

=====

.Supported versions of the ``L1HandlerTransaction`` transaction type

[cols=",,,"]

|=====

|Current version |Deprecated versions | Unsupported versions

| v0 | N/A | N/A

|=====

== Additional resources

* `xref:architecture-and-concepts:smart-contracts/system-calls-cairo1.adoc#send_message_to_L1[send_message_to_L1]` syscall

* `link:https://github.com/starkware-libs/cairo-lang/blob/54d7e92a703b3b5a1e07e9389608178129946efc/src/starkware/starknet/solidity/IStarknetMessaging.sol#L13[sendMessageToL2]` function on the Starknet Core Contract

* For more information on how messaging works within the Starknet Core Contract, including details on coding, see `link:https://book.cairo-lang.org/ch16-04-L1-L2-messaging.html[L1-L2 Messaging]` in `_The Cairo Book: The Cairo Programming Language_`

[id="messaging_function_reference"]

= Messaging function and event reference

:description: Comprehensive function and event reference for the ``StarknetMessaging`` smart contracts.

The ``StarknetMessaging`` smart contracts include functions and events that are required when sending messages between L1 and L2.

For information on the messaging mechanism, see [xref:architecture-and-concepts:network-architecture/messaging-mechanism.adoc](#)[Messaging mechanism].

The L1 functions, where available, are defined in the following smart contract:

```
[cols=",,",]
```

```
|===
```

```
|Contract |Description |Functions
```

```
|https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessaging.sol | StarknetMessaging.sol |
```

```
|The contract that implements messaging functions.
```

Interfaces are available through <https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessaging.sol> [IStarknetMessaging.sol]. a|

```
* xref:#cancelL1ToL2Message[ cancelL1ToL2Message` ]
```

```
* xref:#consumeMessageFromL2[ consumeMessageFromL2` ]
```

```
* xref:#getMaxL1MsgFee[ getMaxL1MsgFee` ]
```

```
* xref:#l1ToL2MessageCancellations[ l1ToL2MessageCancellations` ]
```

```
* xref:#l1ToL2MessageNonce[ l1ToL2MessageNonce` ]
```

```
* xref:#l1ToL2Messages[ l1ToL2Messages` ]
```

```
* xref:#l2ToL1Messages[ l2ToL1Messages` ]
```

```
* xref:#messageCancellationDelay[ messageCancellationDelay` ]
```

```
* xref:#sendMessageToL2[ sendMessageToL2` ]
```

```
* xref:#startL1ToL2MessageCancellation[ startL1ToL2MessageCancellation` ]
```

```
|===
```

The L1 events, where available, are defined in the following smart contract:

```
[cols=",,",]
```

```
|===
```

```
|Contract |Description |Events
```

```
|https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol | StarknetMessagingEvents.sol |
```

```
|The contract that defines messaging events. a|
```

```
* xref:#ConsumedMessageToL1[ ConsumedMessageToL1` ]
```

```
* xref:#ConsumedMessageToL2[ ConsumedMessageToL2` ]
```

```
* xref:#LogMessageToL1[ LogMessageToL1` ]
```

```
* xref:#LogMessageToL2[ LogMessageToL2` ]
```

```
* xref:#MessageToL2Canceled[ MessageToL2Canceled` ]
```

```
* xref:#MessageToL2CancellationStarted[ MessageToL2CancellationStarted` ]
```

```
|===
```

== L1 function reference

Functions are listed in alphabetical order.

'''

[#cancelL1ToL2Message]
=== `cancelL1ToL2Message`

[discrete]
==== Description

Cancels an L1 to L2 message. Call this function after calling the xref:#startL1ToL2MessageCancellation[`startL1ToL2MessageCancellation`] function. The time between the calls to these two functions must be at least the number of seconds defined by the xref:#messageCancellationDelay[`messageCancellationDelay`] function.

Only a sender can cancel a message.

If the message is missing, the call reverts.

Be aware that the message fee is not refunded.

[discrete]
==== State Mutability

None.

[discrete]
==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]
`uint256 _toAddress_`:: The address of the L2 contract.
`uint256 _selector_`::
// tag::uint256_selector[]
The function, in the recipient L2 contract, that the message called.
// end::uint256_selector[]
`uint256[] calldata _payload_`:: The payload of the message.
`uint256 _nonce_`:: The nonce of the message.

[discrete]
==== Returns

[horizontal,labelwidth="30",role=stripes-odd]

``bytes32 _msgHash_``:: The hash of the canceled message.

[discrete]
==== Emitted event

xref:#MessageToL2Canceled[``MessageToL2Canceled``]

[discrete]
==== Function definition

Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol>[``StarknetMessaging.sol``]

* Function: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol#L161>[``cancelL1ToL2Message``]

'''

[#consumeMessageFromL2]
=== ``consumeMessageFromL2``

[discrete]
==== Description

Consumes a message that was sent from an L2 contract.

Returns the hash of the message.

[discrete]
==== State Mutability

None.

[discrete]
==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]
``uint256 _fromAddress_``:: The address of the L2 contract sending the message.
``uint256[] calldata _payload_``:: The payload of the message.

[discrete]
==== Returns

[horizontal,labelwidth="30",role=stripes-odd]
``bytes32 _msgHash_``:: The hash of the consumed message.

[discrete]
==== Emitted event

xref:#ConsumedMessageToL1[`ConsumedMessageToL1`]

[discrete]
==== Function definition

Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol>[`StarknetMessaging.sol`]

* Function: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol#L132>[`consumeMessageFromL2`]

'''

[#getMaxL1MsgFee]
=== `getMaxL1MsgFee`

[discrete]
==== Description

Returns the maximum fee, in Wei, that Starknet accepts for a single message. If the fee passed is higher than this value, the transaction is not accepted.

[discrete]
==== State Mutability

`pure`

[discrete]
==== Parameters

None.

[discrete]
==== Returns

[horizontal,labelwidth="30",role=stripes-odd]
`uint256 _MAX_L1_MSG_FEE_`:: The maximum fee, in Wei, that Starknet accepts for a single message.

[discrete]
==== Function definition

Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/>

starknet/solidity/StarknetMessaging.sol[StarknetMessaging.sol]

* Function: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol#L31>[getMaxL1MsgFee]

'''

[#1ToL2MessageCancellations]
=== `l1ToL2MessageCancellations`

[discrete]
==== Description

Returns the timestamp of the cancellation request.

[discrete]
==== State Mutability

`view`

[discrete]
==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]
`bytes32 _msgHash` :: The message hash.

[discrete]
==== Returns

[horizontal,labelwidth="30",role=stripes-odd]
`uint256 _result` :: The Ethereum block timestamp.

Returns `0` if `cancelL1ToL2Message` was not called with the message hash
`_msgHash`.

[discrete]
==== Function definition

Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol>[StarknetMessaging.sol]

* Function: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol#L73>[l1ToL2MessageCancellations]

'''

[#1ToL2MessageNonce]
=== `1ToL2MessageNonce`

[discrete]
==== Description

Returns the nonce of the next message sent to the L2 contract. So if If `_n_` messages have been sent to Starknet, this function returns `_n_ + 1`.

[discrete]
==== State Mutability

`view`

[discrete]
==== Parameters

None.

[discrete]
==== Returns

[horizontal,labelwidth="30",role=stripes-odd]
`uint256 _nonce_`:: The nonce of the next message sending to L2 contract.

[discrete]
==== Function definition

Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol>[`StarknetMessaging.sol`]

* Function: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol#L55>[`1ToL2MessageNonce`]

'''

[#1ToL2Messages]
=== `1ToL2Messages`

[discrete]
==== Description

Indicates if a pending message is associated with a given message hash.

[discrete]
==== State Mutability

`view`

[discrete]
==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]
`bytes32 _msgHash_`:: The message hash.

[discrete]
==== Returns

Returns `uint256 _result_`, where `_result_` is one of the following:
[horizontal,labelwidth="5",role=stripes-odd]
`message_fee + 1`:: A pending message is associated with the `_msgHash_` parameter.
`0`:: No pending message is associated with the `_msgHash_` parameter.

[discrete]
==== Function definition

Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol>[`StarknetMessaging.sol`]

* Function: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol#L39>[`I1ToL2Messages`]

'''

[#I2ToL1Messages]
=== `I2ToL1Messages`

[discrete]
==== Description

Indicates if a pending message is associated with a given message hash.

[discrete]
==== State Mutability

`view`

[discrete]
==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]

``bytes32 _msgHash_``:: The message hash.

[discrete]
==== Returns

Returns ``uint256 _result_``, where ``_result_`` is one of the following:
[horizontal,labelwidth="30",role=stripes-odd]
``1``:: A pending message is associated with the ``_msgHash_`` parameter.
``0``:: No pending message is associated with the ``_msgHash_`` parameter.

[discrete]
==== Function definition

Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol>[`StarknetMessaging.sol`]

* Function: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol#L43>[`l2ToL1Messages`]

'''

[#messageCancellationDelay]
=== ``messageCancellationDelay``

[discrete]
==== Description

Returns the time interval, in seconds, after which you can cancel a message starting from the moment of calling the `xref:#startL1ToL2MessageCancellation[`startL1ToL2MessageCancellation`]` function. You can get the real value by calling the link:<https://etherscan.io/address/0xc662c410c0ecf747543f5ba90660f6abebd9c8c4#readProxyContract#F11>[`messageCancellationDelay`] function on a block explorer, such as Etherscan.

[discrete]
==== State Mutability

``view``

[discrete]
==== Parameters

None.

[discrete]
==== Returns

[horizontal,labelwidth="30",role=stripes-odd]
`uint256 _result_`:: The time interval.

[discrete]
==== Function definition

Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol>[`StarknetMessaging.sol`]

* Function: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol#L59>[`messageCancellationDelay`]

'''

[#sendMessageToL2]
=== `sendMessageToL2`

[discrete]
==== Description

Sends a message to an L2 contract. The message fee is the cost of executing this function.

[discrete]
==== State Mutability

`payable`

[discrete]
==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]
`uint256 _toAddress_`:: The address of the L2 contract.
`uint256 _selector_`::
include::messaging-reference.adoc[tag=uint256_selector]
`uint256[] calldata _payload_`:: The payload of the message.

[discrete]
==== Returns

[horizontal,labelwidth="30",role=stripes-odd]
`bytes32 _msgHash_`:: The hash of the message.
`uint256 _nonce_`:: The nonce of the message.

[discrete]

==== Emitted event

xref:#LogMessageToL2[`LogMessageToL2`]

[discrete]

==== Function definition

Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol>[`StarknetMessaging.sol`]

* Function: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol#L110>[`sendMessageToL2`]

'''

[#startL1ToL2MessageCancellation]

=== `startL1ToL2MessageCancellation`

[discrete]

==== Description

Starts the cancellation of a message from L1 to L2.

You can cancel a message after a predefined amount of time from the moment this function is called. The amount of time is set by the xref:#messageCancellationDelay[`messageCancellationDelay`] function.

You can only call this function for a message that is currently pending, and the caller must be the sender of that message.

[discrete]

==== State Mutability

None.

[discrete]

==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]

`uint256 _toAddress_`:: The address of the L2 contract.

`uint256 _selector_`::

include::messaging-reference.adoc[tag=uint256_selector]

`uint256[] calldata _payload_`:: The payload of the message.

`uint256 _nonce_`:: The nonce of the message.

[discrete]

==== Returns

[horizontal,labelwidth="30",role=stripes-odd]
`bytes32 _msgHash_`:: The hash of the cancellation message.

[discrete]
==== Emitted event

xref:#MessageToL2CancellationStarted[`MessageToL2CancellationStarted`]

[discrete]
==== Function definition

Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol>[`StarknetMessaging.sol`]

* Function: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/StarknetMessaging.sol#L147>[`startL1ToL2MessageCancellation`]

== L1 event reference

Events are listed in alphabetical order.

// Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol>[`IStarknetMessagingEvents`]

'''

[#ConsumedMessageToL1]
=== `ConsumedMessageToL1`

[discrete]
==== Description

This event is emitted when a message from L2 to L1 is consumed by the
xref:#consumeMessageFromL2[`consumeMessageFromL2`] function.

[discrete]
==== Event attributes

[horizontal,role=stripes-odd]
`uint256 indexed _fromAddress_`:: The address of the sender on L2.
`address indexed _toAddress_`:: The address of the receiver on L1.
`uint256[] _payload_`:: The payload of the consumed message.

[discrete]

==== Event definition

* Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol>[`IStarknetMessagingEvents`]

* Event: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol#L19>[`ConsumedMessageToL1`]

```
// ""
//
// [#ConsumedMessageToL2]
// === `ConsumedMessageToL2`
//
// [discrete]
// ==== Description
//
// This event is emitted when a message from L1 to L2 is consumed by the link:https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/Output.sol#L76[`processMessages`] function.
//
// [discrete]
// ==== Event attributes
//
// [horizontal,role=stripes-odd]
// `address indexed _fromAddress_`:: The address of the sender on L1.
// `uint256 indexed _toAddress_`:: The address of the receiver on L2.
// `uint256 indexed _selector_`::
// include::messaging-reference.adoc[tag=uint256_selector]
// `uint256[] _payload_`:: The payload of the consumed message.
// `uint256 _nonce_`:: The nonce of the consumed message.
//
// [discrete]
// ==== Event definition
//
// * Contract: link:https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol[`IStarknetMessagingEvents`]
//
// * Event: link:https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol#L26[`ConsumedMessageToL2`]
//
// ""
//
// [#LogMessageToL1]
// === `LogMessageToL1`
//
// [discrete]
```



```
// ==== Description
//
// This event is emitted when a message is sent from L2 to L1 by the link:https://
github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/
Output.sol#L76[`processMessages`] function.
//
// [discrete]
// ==== Event attributes
//
// [horizontal,role=stripes-odd]
// `uint256 indexed _fromAddress_`:: The address of the sender on L2.
// `address indexed _toAddress_`:: The address of the receiver on L1.
// `uint256[] _payload_`:: The payload of the message.
//
// [discrete]
// ==== Event definition
//
// * Contract: link:https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/
starknet/solidity/IStarknetMessagingEvents.sol[`IStarknetMessagingEvents`]
//
// * Event: link:https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/
starknet/solidity/IStarknetMessagingEvents.sol#L6[`LogMessageToL1`]
//
'''
```

```
[#LogMessageToL2]
=== `LogMessageToL2`
```

```
[discrete]
==== Description
```

This event is emitted when a message is sent from L1 to L2 by the
xref:#sendMessageToL2[`sendMessageToL2`] function.

```
[discrete]
==== Event attributes
```

```
[horizontal,role=stripes-odd]
`address indexed _fromAddress_`:: The address of the sender on L1.
`uint256 indexed _toAddress_`:: The address of the receiver on L2.
`uint256 indexed _selector_`::
include::messaging-reference.adoc[tag=uint256_selector]
`uint256[] _payload_`:: The payload of the message.
`uint256 _nonce_`:: The nonce of the message.
`uint256 _fee_`:: The fee associated with the message.
```

[discrete]

==== Event definition

* Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol>[`IStarknetMessagingEvents`]

* Event: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol#L9>[`LogMessageToL2`]

'''

[#MessageToL2Canceled]

=== `MessageToL2Canceled`

[discrete]

==== Description

This event is emitted when an L1 to L2 message is canceled by the xref:#cancelL1ToL2Message[`cancelL1ToL2Message`] function.

[discrete]

==== Event attributes

[horizontal,role=stripes-odd]

`address indexed _fromAddress_`:: The address of the sender on L1.

`uint256 indexed _toAddress_`:: The address of the receiver on L2.

`uint256 indexed _selector_`::

include::messaging-reference.adoc[tag=uint256_selector]

`uint256[] _payload_`:: The payload of the canceled message.

`uint256 _nonce_`:: The nonce of the canceled message.

[discrete]

==== Event definition

* Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol>[`IStarknetMessagingEvents`]

* Event: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol#L44>[`MessageToL2Canceled`]

'''

[#MessageToL2CancellationStarted]

=== `MessageToL2CancellationStarted`

[discrete]

==== Description

This event is emitted when the cancellation of an L1 to L2 message is started by the `xref:#startL1ToL2MessageCancellation[`startL1ToL2MessageCancellation`]` function.

[discrete]

==== Event attributes

[horizontal,role=stripes-odd]

`address indexed _fromAddress_`:: The address of the sender on L1.

`uint256 indexed _toAddress_`:: The address of the receiver on L2.

`uint256 indexed _selector_`::

include::messaging-reference.adoc[tag=uint256_selector]

`uint256[] _payload_`:: The payload of the message to be canceled.

`uint256 _nonce_`:: The nonce of the message to be canceled.

[discrete]

==== Event definition

* Contract: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol>[`IStarknetMessagingEvents`]

* Event: link:<https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/solidity/IStarknetMessagingEvents.sol#L35>[`MessageToL2CancellationStarted`]

[id="starknet_architecture_overview"]

= Starknet architecture: Overview

:description: An overview of the different components of Starknet, including sequencers, provers, and nodes. Explains the roles of each component and how they interact to create a highly scalable, efficient, and secure network. Debunks common misconceptions about the relationship between sequencers and provers. Discusses the different methods for nodes to keep track of the network's state.

:keywords: Starknet, Starknet architecture, [.x .x-first .x-last]#sequencers#, Provers, Nodes, Starknet roles, Starknet components, Starknet architecture overview, Starknet architecture introduction, Starknet architecture details, Starknet architecture explanation, Starknet architecture guide, Starknet architecture tutorial, Starknet architecture documentation, Starknet architecture manual, Starknet architecture reference, Starknet architecture handbook, Starknet architecture how-to

Starknet is a coordinated system, with each component—sequencers, provers, and nodes—playing a specific yet interconnected role. Although Starknet is not yet fully decentralized, it's actively moving toward

that goal. This description of the system's roles and how they interact should help you better grasp the intricacies of the Starknet ecosystem.

== Transaction flow

Starknet's operation begins when a transaction is received by a gateway, which serves as the Mempool. This stage could also be managed by the sequencer. The transaction is initially marked as `RECEIVED`. The sequencer then incorporates the transaction into the network state and tags it as `ACCEPTED_ON_L2`. The final step involves the prover, which executes the operating system on the new block, calculates its proof, and submits it to Layer 1 (L1) for verification.

For more information on the transaction flow, see [xref:network-architecture/transaction-life-cycle.adoc\[\]](#).



In essence, Starknet's architecture involves multiple components:

- * The `_sequencer_` receives transactions, orders them, and produces blocks. It operates similarly to validators in Ethereum or Bitcoin.
- * The `_prover_` generates proofs for the created blocks and transactions. It uses the Cairo Virtual Machine (Cairo VM) to run provable programs, thereby creating execution traces necessary for generating STARK proofs.
- * L1, in this case, Ethereum, hosts a smart contract capable of verifying these STARK proofs. If the proofs are valid, Starknet's state root on L1 is updated.

== Starknet's state

Starknet's state is a comprehensive snapshot maintained through Merkle trees, much like in Ethereum. This establishes the architecture of the validity roll-up and the roles of each component.

// After exploring the introductory overview of the different components,
// delve deeper into their specific roles by referring to their dedicated
// subchapters.

== Sequencers

Sequencers are the backbone of the Starknet network, similar to Ethereum's

validators. They usher transactions into the system.

Validity Rollups, also known as ZK-Rollups, excel at offloading some network chores, like bundling and processing transactions, to specialized players. This setup is similar to how Ethereum and Bitcoin delegate security to miners. Sequencing, like mining, demands hefty resources.

Validity Rollups like Starknet outsource transaction processing to specialized entities and then verify their work. These specialized entities, in the context of Validity Rollups, are known as sequencers.

Instead of providing security, as miners do, sequencers provide transaction capacity. They order, that is, sequence, multiple transactions into a single batch, execute them, and produce a block that is later proved by the prover and submitted to the Layer 1 network as a single, compact proof, known as a rollup. Just as validators in Ethereum and miners in Bitcoin are specialized actors securing the network, sequencers in Validity Rollup-based networks are specialized actors that provide transaction capacity.

This mechanism enables Validity Rollups to handle a higher volume of transactions while maintaining the security of the underlying Ethereum network, enhancing scalability without compromising on security.

Sequencers follow a systematic method for processing transactions:

- . Sequencing: They collect transactions from users and order them.
- . Executing: Sequencers then process these transactions.
- . Batching: Transactions are grouped together in batches for efficiency.
- . Block Production: Sequencers produce blocks that contain batches of processed transactions.

Sequencers must be reliable and highly available, as their role is critical to the network's smooth functioning. They need powerful and well-connected machines to perform their role effectively, as they must process transactions rapidly and continuously.

The current roadmap for Starknet includes decentralizing the sequencer role. This shift towards decentralization will enable more participants to become sequencers, contributing to the robustness of the network.

// For more details on the sequencer role, refer to the dedicated subchapter.

== Provers

Provers serve as the second line of verification in the Starknet network. Their main task is to validate the work of the sequencers, when they receive the block produced by the sequencer, and to generate proofs that these processes were correctly performed.

A prover does the following:

- . *Receives blocks:* Provers receive blocks of processed transactions from sequencers.
- . *Processes blocks:* Provers process these blocks a second time, ensuring that all transactions within the block have been correctly handled.
- . *Generates a proof:* After processing, provers generate a proof of correct transaction processing.
- . *Sends the proof to Ethereum:* Finally, the proof is sent to the Ethereum network for validation. If the proof is correct, the Ethereum network accepts the block of transactions.

Provers need even more computational power than sequencers because they have to calculate and generate proofs, a process that is computationally heavy. However, the work of provers can be split into multiple parts, allowing for parallelism and efficient proof generation. The proof generation process is asynchronous, meaning it doesn't have to occur immediately or in real-time. This flexibility allows for the workload to be distributed among multiple provers. Each prover can work on a different block, allowing for parallelism and efficient proof generation.

The design of Starknet relies on these two types of actors—sequencers and provers—working in tandem to ensure efficient processing and secure verification of transactions.

// For more details on the role of the prover , refer to the dedicated subchapter.

== Optimizing sequencers and provers: Debunking common misconceptions

The relationship between sequencers and provers in blockchain technology often sparks debate. A common misunderstanding suggests that either the

prover or the sequencer is the main bottleneck. To set the record straight, let's discuss the optimization of both components.

Starknet, which uses the Cairo programming language, currently supports only sequential transactions. Plans are in place to introduce parallel transactions in the future. However, as of now, the sequencer operates one transaction at a time, making it the bottleneck in the system.

In contrast, provers operate asynchronously and can execute multiple tasks in parallel. The use of proof recursion allows for task distribution across multiple machines, making scalability less of an issue for provers.

// I'd prefer not to discuss time-bound info, like current development focus, unless we keep it within the bounds of specific release versions.

// Given the asynchronous and scalable nature of provers, Starknet development // is currently focused on enhancing the sequencer's efficiency.

== Nodes

When it comes to defining what nodes do in Bitcoin or Ethereum, people often misinterpret their role as keeping track of every transaction within the network. This, however, is not entirely accurate.

Nodes serve as auditors of the network, maintaining the state of the network, such as how much Bitcoin each participant owns or the current state of a specific smart contract. They maintain network state by processing transactions and preserving a record of all transactions, but that is a means to an end, not the end itself.

In Validity Rollups and specifically within Starknet, this concept is somewhat reversed. Nodes don't necessarily have to process transactions to get the state. In contrast to Ethereum or Bitcoin, Starknet nodes aren't required to process all transactions to maintain the state of the network.

You can access network state data by using the Starknet API, which uses the JSON RPC protocol, to communicate with a node. Previously, Starknet's Gateway and Feeder Gateway APIs enabled querying the sequencer, but as the network has matured, the JSON RPC has become the standard.

Operating your own node is typically faster than using a shared architecture, like the gateway.

It's worth noting that more people running nodes increases the resilience of the network and prevents server flooding, which has been an issue in other L2 networks.

Currently, there are primarily three methods for a node to keep track of the network's state and nodes can implement any of these methods:

. ***Replaying Old Transactions***: Like Ethereum or Bitcoin, a node can take all the transactions and re-execute them. Although this approach is accurate, it isn't scalable unless you have a powerful machine that's capable of handling the load. If you can replay all transactions, you can become a sequencer.

. ***Relying on L2 Consensus***: Nodes can trust the sequencer to execute the network correctly. When the sequencer updates the state and adds a new block, nodes accept the update as accurate.

. ***Checking Proof Validation on L1***: Nodes can monitor the state of the network by observing L1 and ensuring that every time a proof is sent, they receive the updated state. This way, they don't have to trust anyone and only need to keep track of the latest valid transaction for Starknet.

Each type of node setup comes with its own set of hardware requirements and trust assumptions.

=== Nodes that replay transactions

Nodes that replay transactions require powerful machines to track and execute all transactions. These nodes don't have trust assumptions; they rely solely on the transactions they execute, guaranteeing that the state at any given point is valid.

=== Nodes that rely on L2 consensus

Nodes that rely on L2 consensus require less computational power. They need sufficient storage to keep the state but don't need to process a lot of transactions. The tradeoff here is a trust assumption.

Currently, Starknet revolves around one sequencer, so these nodes trust StarkWare not to disrupt the network. However, once a consensus mechanism and leader election amongst sequencers are in place, these nodes will only need to trust that a sequencer who staked their stake to produce a block is not willing to lose it.

=== Nodes that check proof validation on L1

Nodes that only update their state based on proof validation on L1 require the least hardware. They have the same requirements as an

Ethereum node, and once Ethereum light nodes become a reality, maintaining such a node could be as simple as using a smartphone. The only tradeoff is higher latency. Proofs are sent to Ethereum intermittently, not for every block, resulting in delayed state updates. Plans are in place to produce proofs more frequently, even if they are not sent to Ethereum immediately, lowering node latency.

// == Conclusion

//

// Through this chapter, we delve into Starknet's structure, uncovering the
// importance of sequencers, provers, and nodes. Each plays a unique role,
// but together, they create a highly scalable, efficient, and secure
// network that marks a significant step forward in Layer 2 solutions. As
// Starknet progresses towards decentralization, understanding these roles
// will provide valuable insight into the inner workings of this network.

//

// As we venture further into the Starknet universe, our next stop will be
// an exploration of the transaction lifecycle before we dive into the
// heart of coding with Cairo.

[id="starknet_state"]

= Starknet state

:stem: latexmath

Starknet's state consists of:

[horizontal,labelwidth="25",role="stripes-odd"]

xref:architecture-and-concepts:smart-contracts/contract-classes.adoc[Contract
classes]:: a mapping

between the class hash and the class definition

xref:architecture-and-concepts:smart-contracts/contract-classes.adoc[Contract
instances]:: a mapping between addresses (251-bit field elements) and the contract's
state

A contract instance's state consists of:

[horizontal,labelwidth="25",role="stripes-odd"]

xref:smart-contracts/class-hash.adoc[Class hash]:: defines the functionality of the
contract

xref:smart-contracts/contract-storage.adoc[Contract storage]:: a key-value mapping
where the key/values are field elements

xref:accounts/approach.adoc#replay_protection[Contract nonce]:: the number of transactions sent from this contract

[#transitioning_to_a_new_state]
== Transitioning to a new state

A transaction stem:[\$tx\$] transitions the system from state stem:[\$S\$] to state stem:[\$S'\$] if:

* stem:[\$tx\$] is an xref:network-architecture/transactions.adoc#invoke_transaction[Invoke] transaction, and the storage of stem:[\$S'\$] is the result of executing the target contract code with respect to the previous state stem:[\$S\$]. The arguments, contract instance's address, and the specific function entry point are part of the transaction.

* stem:[\$tx\$] is a xref:network-architecture/transactions.adoc#deploy_account_transaction[Deploy account] transaction, and stem:[\$S'\$] contains the new contract instance's state at the contract instance's address. Additionally, the storage of stem:[\$S\$] is updated according to the execution of the contract instance's constructor.

* stem:[\$tx\$] is a xref:network-architecture/transactions.adoc#declare_transaction[Declare] transaction, and stem:[\$S'\$] contains the class hash and definition in the contract class's mapping

[id="state_commitment"]
== State commitment

The state commitment is a digest that represents the state.

In Starknet, the state commitment combines the roots of two binary xref:#merkle_patricia_trie[Merkle-Patricia tries] of height 251 in the following manner:

```
[,subs="quotes"]
----
state_commitment = _h_~Pos~(
    "STARKNET_STATE_V0",
    contract_trie_root,
    class_trie_root
)
----
```

Where:

* `_h_~Pos~`` is the xref:cryptography/hash-functions.adoc#poseidon_hash[Poseidon] hash function.

- * ``STARKNET_STATE_V0`` is a constant prefix string encoded in ASCII (and represented as a field element).
- * ``contract_trie_root`` is the root of the `xref:#contracts_trie[_contract_trie_]`, a Merkle-Patricia trie whose leaves are the contracts' states.
- * ``class_trie_root`` is the root of the `xref:#classes_trie[_class_trie_]`, a Merkle-Patricia trie whose leaves are the compiled class hashes.

```
[id="contracts_trie"]
=== The contract trie
```

As with Ethereum, this trie is a two-level structure, whose leaves correspond to distinct contracts. The address of each contract determines the path from the trie's root to its corresponding leaf, whose content encodes the contract's state.

The information stored in the leaf is as follows:

```
// [stem]
// ++++
// h(h(h(\text{class_hash}, \text{storage_root}), \text{nonce})),0)
// ++++
```

```
[,subs="quotes"]
----
_h_~Ped~(
  _h_~Ped~(
    _h_~Ped~(
      class_hash,
      storage_root
    ),
    nonce
  ),
  0
)
----
```

Where:

- * ``_h_~Ped~`` is the `xref:../cryptography/hash-functions.adoc#pedersen_hash[Pedersen]` hash function.
- * ``class_hash`` is the hash of `xref:../smart-contracts/class-hash.adoc` [the contract's definition].
- * ``storage_root`` is the root of another Merkle-Patricia trie of height 251 that is constructed from the contract's storage.
- * ``nonce`` is the current nonce of the contract.

```
[id="classes_trie"]  
=== The class trie
```

The class trie encodes the information about all existing
xref:../smart-contracts/contract-classes.adoc[classes] in Starknet's state. This trie maps
xref:smart-contracts/class-hash.adoc#cairo1_class[class hashes] to their
compiled class hashes. The information stored in a leaf at a path corresponding to
some class hash is as follows:

```
[source,subs="quotes"]  
----  
_h_~Pos~(  
  CONTRACT_CLASS_LEAF_V0,  
  compiled_class_hash  
)  
----
```

Where:

- * `_h_~Pos~` is the xref:../cryptography/hash-functions.adoc#poseidon_hash[Poseidon] hash function
- * `CONTRACT_CLASS_LEAF_V0` is a constant prefix string encoded in ASCII (and represented as a field element).
- * `compiled_class_hash` is the hash of the Cairo assembly resulting from compiling the given class via the Sierra-to-Casm compiler.

```
[#note_compiled_class_hash]  
[NOTE]  
====  
.Compiled class hash
```

The compiled class hash identifies the output of a specific Casm compilation as unique.

Cairo classes that are part of the state commitment are defined with Sierra, an intermediate representation between Cairo and Cairo assembly (Casm). However, the prover only works with Casm.

So in order to prevent needing to compile from Sierra to Casm in every block in which the class is used, the state commitment must have some information about the corresponding Cairo assembly. The compiled class hash provides this information.

For more information, see xref:architecture-and-concepts:smart-contracts/cairo-and-sierra.adoc[Cairo and Sierra].

The party that declares the contract signs the compiled class hash, which they obtain

using an SDK, as part of the xref:network-architecture/transactions.adoc#declare_v2[`DECLARE``] transaction. If the transaction is included in a block, then the compiled class hash becomes part of the state commitment.

In the future, when Sierra-to-Casm compilation becomes part of the Starknet OS, this value might be updated via a proof of the Sierra-to-Casm compiler execution, showing that compiling the same class with a newer compiler version results in some new compiled class hash.

====

[#merkle_patricia_trie]
== Merkle-Patricia trie

The state commitment scheme uses a binary Merkle-Patricia trie with the Pedersen hash function.

[#about_nodes]
=== About nodes

Each node in the trie is represented by a triplet stem:[\$(length, path, value)\$], where:

[horizontal,labelwidth=10,role="stripes-odd"]
stem:[\$length\$]: is the length of the path, measured in nodes.

stem:[\$path\$]: is the path from the current node to its unique non-empty subtrie.

+

stem:[\$path\$] is an integer in the set stem:[\$\{0,\ldots,2^{\text{length}}-1\}\$], and the binary expansion of stem:[\$path\$] indicates how to proceed along the trie, as follows:

+

. Expand stem:[\$path\$] to its binary representation.

. Starting with the most significant bit, representing the root of the trie, traverse the tree node by node, where the bit values stem:[\$0\$] and stem:[\$1\$] indicate left and right, respectively.

stem:[\$value\$]: is the value of the node, which can be either data, or the hash of two non-empty child nodes.

An empty node is one whose triplet values are stem:[\$(0,0,0)\$]. Leaf nodes and internal nodes can be empty. A subtrie rooted at a node stem:[\$(length, path, value)\$] has a single non-empty subtrie, rooted at the node obtained by following the path specified by stem:[\$path\$].

[NOTE]

====

Length is specified, and cannot be deduced from stem:[\$path\$], because the numbers in the triplet stem:[\$(length, path, value)\$] are field elements of fixed size, 251 bits each.

For a node where $\text{stem}:[\text{length} > 0]$, $\text{stem}:[\text{path}]$ leads to the highest node whose left and right children are not empty.

====

=== Trie construction

The following rules specify how the trie is constructed from a given set of leaves:

The hash of a node $\text{stem}:[N = (\text{length}, \text{path}, \text{value})]$, denoted by $\text{stem}:[H(N)]$, is:

```
[stem]
++++
H(N) = \begin{cases}
\text{value}, & \text{if } \text{length} = 0 \\
h_{\text{Ped}}(\text{value}, \text{path}) + \text{length}, & \text{otherwise}
\end{cases}
\end{cases}
++++
```

[NOTE]

====

All arithmetic operations in the above description of $\text{stem}:[H]$ are done in the STARK field, as described in [xref:cryptography/p-value.adoc](#) [The STARK field].

====

=== Mathematical definition of the nodes in the trie

The triplet representing the parent of the nodes $\text{stem}:[\text{left} = (\ell_L, p_L, v_L)]$, $\text{stem}:[\text{right} = (\ell_R, p_R, v_R)]$ is defined as follows:

```
[stem]
++++
parent =
\begin{cases}
(0, 0, 0), & \text{if } \text{left} = \text{right} = (0, 0, 0) \\
(\ell_L + 1, p_L, v_L), & \text{if } \text{right} = (0, 0, 0) \text{ and } \text{left} \neq (0, 0, 0) \\
(\ell_R + 1, p_R + 2^{\ell_R}, v_R), & \text{if } \text{right} \neq (0, 0, 0) \text{ and } \text{left} = (0, 0, 0) \\
(0, 0, h_{\text{Ped}}(H(\text{left}), H(\text{right}))), & \text{otherwise}
\end{cases}
\end{cases}
++++
```

[#example_trie]

=== Example trie

The diagram [xref:#3-level_trie\[\]](#) illustrates the construction of a three-level-high Merkle-Patricia trie from the leaves whose values are $\text{stem}:[(0, 0, 1, 0, 0, 1, 0, 0)]$:

[#3-level_trie]

.A three-level Merkle-Patricia trie

image::trie.png[3-level-high Merkle-Patricia trie]

Where $\text{stem}:[\$r=h_{\{\text{Ped}\}}(H(2,2,1),H((2,1,1)))\$]$. Notice that the example does not skip from the root, whose length is zero, so the final state commitment to the trie is $\text{stem}:[\$H((0,0,r))=r\$]$.

Suppose that you want to prove, with respect to the state commitment just computed, that the value of the leaf whose path is given by $\text{stem}:[\$101\$]$ is $\text{stem}:[\$1\$]$. In a standard Merkle trie, the proof would consist of data from three nodes, which are siblings along the path to the root.

In a Merkle-Patricia trie, because the trie is sparse, you only need to send the two children of the root, which are $\text{stem}:[\$ (2,2,1) \$]$ and $\text{stem}:[\$ (2,1,1) \$]$. These two children are enough to reproduce the state commitment $\text{stem}:[\$r\$]$, and because you know that the height of the trie is three, and that it is fixed, you know that the path $\text{stem}:[\$01\$]$ of length $\text{stem}:[\$2\$]$ specified by the right-hand child, $\text{stem}:[\$ (2,1,1) \$]$, leads to the desired leaf.

== Special addresses

Starknet uses special contract addresses to provide distinct capabilities beyond regular contract deployment.

Two such addresses are ``0x0`` and ``0x1``. These addresses are reserved for specific purposes and are characterized by their unique behavior in comparison to traditional contract addresses.

=== Address ``0x0``

Address ``0x0`` functions as the default ``caller_address`` for external calls, including interactions with the L1 handler or deprecated Deploy transactions. Unlike regular contracts, address ``0x0`` does not possess a storage structure and does not accommodate storage mapping.

=== Address ``0x1``

Address ``0x1`` is another special contract address within Starknet's architecture. It functions as a storage space for mapping block numbers to their corresponding block hashes. The storage structure at this address is organized as follows:

[horizontal,labelwidth="20",role="stripes-odd"]

Keys:: Block numbers between $\text{stem}:[\text{first_v0_12_0_block}]$ and $\text{stem}:[\text{current_block} - 10]$.

Values:: Corresponding block hashes for the specified blocks.

Default Values:: For all other block numbers, the values are set to `0`.

The storage organization of address `0x1` supports the efficient retrieval of block hashes based on block numbers within a defined range and is also used by the `xref:architecture-and-concepts:smart-contracts/system-calls-cairo1.adoc#get_block_hash[get_block_hash]` system call.

`[id="transaction_lifecycle"]`

= Transaction lifecycle

`[id="transaction_flow"]`

== Transaction flow

image::transaction-flow.png[]

The high-level steps in the Starknet transaction lifecycle are as follows:

. ***Transaction submission:** A transaction is submitted to one of the gateways, which functions as the mempool, and marks the transaction status as `RECEIVED`.

. ***Mempool validation:**

The mempool performs a preliminary validation on the transaction, such as ensuring that the current account balance exceeds the value of `max_fee` (prior to v3 transactions) or assuring the transaction's calldata length is within the legal limit. If the transaction is invalid, it does not proceed.

+

Mempool validation in this context is analogous to Ethereum's signature checking, including running the account's `+__validate__+` function on an `INVOKE` transaction, `+__validate_declare__+` on a `DECLARE` transaction, or `+__validate_deploy__+` on a `DEPLOY_ACCOUNT` transaction, ensuring that the current account balance exceeds the value of `max_fee` (prior to v3 transactions), and more.

. ***Sequencer validation:** The sequencer performs preliminary validation on the transaction before executing it to ensure that the transaction is still valid. If the transaction is invalid, it does not proceed.

+

This validation stage repeats the same validation run during the mempool validation.

. ***Execution:** The sequencer operation sequentially applies all transactions that passed the preliminary validation to the state. If a transaction fails during execution, it is included in the block with the status `REVERTED`.

. ***Proof generation and verification:** The Prover executes the operating system on the new block, computes the proof, and transmits it to the L1 verifier, which verifies the proof. At this point, the L1 state is updated to include the transaction.

[id="transaction_status"]
== Transaction status

The diagram below illustrates how each transaction status fits into the overall transaction flow:

image::txn-flow.png[]

The following are the possible statuses of a transaction from the moment a user sends it until the moment it passes the L1 verifier:

[cols="1,2,4",]

|===

|Status type |Status |Explanation

.5+|*Finality* |`NOT_RECEIVED` |The transaction is not yet known to the sequencer.
|`RECEIVED` a|The transaction was received by the mempool. The transaction now either executes successfully, is rejected, or reverted.

The transaction has no execution status.

|`REJECTED` a|The transaction was received by the mempool but failed validation in the sequencer. Such transactions are not included in a block.

The transaction has no execution status.

[NOTE]

=====

A rejected transaction is stored in the mempool. You cannot send another transaction with the same transaction hash.

=====

|`ACCEPTED_ON_L2` |The transaction was executed and entered an actual created block on L2.

|`ACCEPTED_ON_L1` |The transaction was accepted on Ethereum.

.2+|*Execution* |`REVERTED` a|The transaction passed validation but failed during execution in the sequencer. It is included in the block with the status `REVERTED`.

[NOTE]

=====

Since only `INVOKE` transactions have an execution phase, `DEPLOY_ACCOUNT` and `DECLARE` transactions cannot be reverted. If either the `+__VALIDATE_DEPLOY__+` or the `+__VALIDATE_DECLARE__+` function fails when run in the sequencer, then the transaction is rejected.

=====

|`SUCCEEDED`|The transaction was successfully executed by the sequencer. It is included in the block.

|===

[id="transaction-state-implications"]

== State implications of a reverted transaction

When a transaction is marked as `REVERTED`, the following state implications occur:

[horizontal,labelwidth="20",role="stripes-odd"]

Nonce increases:: The nonce value for the account of the failed transaction iterates despite the failure.

Fee charge:: The sequencer charges a fee for the execution of the transaction up to the point of failure. A `Transfer` event is emitted.

Partial reversion:: All changes that occurred during the validation stage are not reverted. However, all changes that occurred during the execution stage are reverted, including all messages to L1 or any events that were emitted during this stage.

+

Events might still be emitted from the validation stage or the fee charge stage.

Fee calculation:: The fee charged for `REVERTED` transactions is the smaller of the following two values:

- * The maximum fee that the user is willing to pay, either `max_fee` (pre-v3 transactions) or $\text{stem}[\text{max_amount}] \cdot \text{max_price_per_unit}$ (v3 transactions).

- * The total consumed resources.

// calculated as follows:

// +

// stem:[$\text{actual_fee} = \text{Min}(\text{max_fee}, \text{consumed_resources})$]. For v3 transactions, max_fee is `max_amount` x `max_price_per_unit`.

Consumed Resources:: The resources used for the execution of the transaction up to the point of failure. This includes Cairo steps, builtins, syscalls, L1 messages, events, and state diffs during the validation and execution stages.

[id="transaction_receipt"]

== Transaction receipt

To get a receipt, use the JSON RPC method `starknet_getTransactionReceipt`.

// For example, using a node with the API Key `1234`, and the transaction hash `0xdeadbeef`, you can enter the following `curl` command to get a receipt for a transaction on Sepolia testnet:

//

```
// [source,bash]
// ----
// curl https://starknet-sepolia.rpc_provider.io/v3/1234 \
// -X POST \
// -H "Content-Type: application/json" \
// -d '{"jsonrpc":"2.0","method":"starknet_getTransactionReceipt","params":
["0xdeadbeef"],"id":1}'
// ----
```

The transaction receipt contains the following fields:

```
[horizontal,labelwidth="25",role="stripes-odd"]
`transaction_hash`:: The hash of the transaction.
`actual_fee`:: The actual fee paid for the transaction.
`finality_status`:: The finality status of the transaction.
`execution_status`:: The execution status of the transaction.
`block_hash`:: The hash of the block that includes the transaction
`block_number`:: The sequential number of the block that includes the transaction
`messages_sent`:: A list of messages sent to L1.
`events`:: The events emitted.
`execution_resource`:: A summary of the execution resources used by the transaction.
`type`:: The type of the transaction.
```

The following is an example of a receipt:

```
[source,json]
----
{
  "jsonrpc": "2.0",
  "result": {
    "actual_fee": "0x221db5dbf6db",
    "block_hash":
"0x301fc0d09c5810600af7bb9610be10596ad6f4e6d28a60d397dd148f0962a88",
    "block_number": 906096,
    "events": [
      {
        "data": [
          "0x181de8b0cd32999a5cc962c5f724bc0f6a322f02957b80e1d5fef49a87588b7",
          "0x0",
          "0x9184e72a000",
          "0x0"
        ],
        "from_address":
"0x49d36570d4e46f48e99674bd3fcc84644ddd6b96f7c741b1562b82f9e004dc7",
        "keys": [
          "0x99cd8bde557814842a3121e8ddfd433a539b8c9f14bf31ebf108d12e6196e9"
        ]
      }
    ]
  }
}
```

```

    ]
  },
  {
    "data": [
      "0x764da020183e28a48ee38a9474f84e7e5ff13194",
      "0x9184e72a000",
      "0x0",
      "0x181de8b0cd32999a5cc962c5f724bc0f6a322f02957b80e1d5fef49a87588b7"
    ],
    "from_address":
      "0x73314940630fd6dcda0d772d4c972c4e0a9946bef9dabf4ef84eda8ef542b82",
    "keys": [
      "0x194fc63c49b0f07c8e7a78476844837255213824bd6cb81e0ccfb949921aad1"
    ]
  },
  {
    "data": [
      "0x181de8b0cd32999a5cc962c5f724bc0f6a322f02957b80e1d5fef49a87588b7",
      "0x1176a1bd84444c89232ec27754698e5d2e7e1a7f1539f12027f28b23ec9f3d8",
      "0x221db5dbf6db",
      "0x0"
    ],
    "from_address":
      "0x49d36570d4e46f48e99674bd3fcc84644ddd6b96f7c741b1562b82f9e004dc7",
    "keys": [
      "0x99cd8bde557814842a3121e8ddfd433a539b8c9f14bf31ebf108d12e6196e9"
    ]
  }
],
"execution_status": "SUCCEEDED",
"finality_status": "ACCEPTED_ON_L2",
"messages_sent": [
  {
    "from_address":
      "0x73314940630fd6dcda0d772d4c972c4e0a9946bef9dabf4ef84eda8ef542b82",
    "payload": [
      "0x0",
      "0x764da020183e28a48ee38a9474f84e7e5ff13194",
      "0x9184e72a000",
      "0x0"
    ],
    "to_address": "0xc3511006c04ef1d78af4c8e0e74ec18a6e64ff9e"
  }
],
"transaction_hash": "0xdeadbeef",
"type": "INVOKE"

```

```

    },
    "id": 1
  }
}
-----

```

```

[id="transaction_structure"]
= Transaction types

```

Starknet supports the following types of transactions, as defined in the Starknet API:

```

[horizontal,labelwidth="20"]
`DECLARE`:: Declares new contract classes, enabling new contract instances.
`INVOKE`:: Invokes an existing function in a contract.
`DEPLOY_ACCOUNT`:: Deploys new account contracts in smart wallets.

```

To see how these transaction types appear in the Starknet API, see link:https://github.com/starkware-libs/starknet-specs/blob/b5c43955b1868b8e19af6d1736178e02ec84e678/api/starknet_api_openrpc.json[starknet_api_openrpc.json].

This topic describes the available fields for these transaction types and how each transaction's hash is calculated.

```

[id="transaction_versioning"]
== Transaction versions

```

When the fields that comprise a transaction change, either with the addition of a new field or the removal of an existing field, then the transaction version increases.

Deprecated transaction versions are still supported, but support will be removed in a future release of Starknet.

.Supported versions of Starknet transaction types

```

[cols=",,",]
|===
|Transaction name |Current version |Deprecated versions | Unsupported versions

```

```

|`INVOKE` | v3 |v1, v0 | N/A
|`DECLARE` |v3 |v2, v1 | v0
|`DEPLOY_ACCOUNT` | v3 | v0 | N/A
|`DEPLOY` | N/A | N/A | v0
|===

```

Additionally, see information on the [xref:network-architecture/messaging-mechanism.adoc#l1_handler_transaction](#)[L1 handler transaction type].

[IMPORTANT]

====

Do not submit a transaction that uses an unsupported transaction type, because it cannot be included in a proof, and so cannot become part of a Starknet block.

====

[NOTE]

====

While the ``L1HandlerTransaction`` type is a valid transaction type within Starknet, be aware that it cannot be broadcast through the Starknet API like the other transaction types listed in the table above. This transaction type is specifically designed for internal Starknet operations, particularly for handling messages from L1 to L2.

For more details, refer to the xref:network-architecture/messaging-mechanism.adoc#l1_handler_transaction [L1 handler transaction page].

====

[id=invoke_transaction]
== ``INVOKE`` transaction

:invoke:

// [IMPORTANT]

// =====

// ``INVOKE`` transaction versions 0 and 1 are deprecated and will be removed in a future release of Starknet.

// =====

The ``INVOKE`` transaction type invokes a function in an existing contract instance. The contract code of the account that sends the ``INVOKE`` transaction determines how to process the transaction.

[NOTE]

=====

Because an account's ``+__validate__+`` and ``+__execute__+`` functions can contain any logic, the account ultimately determines how to handle the ``INVOKE`` transaction.

=====

Every ``INVOKE`` transaction in Starknet undergoes the validation and execution stages, initiated by the ``+__validate__+`` and ``+__execute__+`` functions. The validation stage verifies that the account that sent the transaction approves it.

=== v3 transaction fields

include::partial\$snippet_transaction_fields_master_table.adoc[]

=== v3 hash calculation

The `INVOKE` v3 transaction hash is calculated as a Poseidon hash over the given transaction elements, specifically:

```
[,subs="quotes"]
----
invoke_v3_tx_hash = _h(
    "invoke",
    version,
    nonce,
    sender_address,
    _h(tip, l1_gas_bounds, l2_gas_bounds),
    _h(paymaster_data),
    chain_id,
    data_availability_modes,
    _h(account_deployment_data),
    _h(calldata)
)
----
```

Where:

* `invoke` is a constant prefix string, encoded in ASCII.
include::partial\$snippet_transaction_params.adoc[]

=== v1 (deprecated) transaction fields

. `INVOKE` v1 transaction fields

[%autowidth.stretch]

|===

| Name | Type | Description

| `sender_address` | `FieldElement` | The address of the sender of this transaction.

| `calldata` | `List<FieldElement>` | The arguments that are passed to the `validate` and `execute` functions.

| `signature` | `List<FieldElement>` | Additional information given by the sender, used to validate the transaction.

| `max_fee` | `FieldElement` | The maximum fee that the sender is willing to pay for the transaction

| `nonce` | `FieldElement` | The transaction nonce.

| `version` | `FieldElement` | The transaction's version. The value is 1. +

When the fields that comprise a transaction change, either with the addition of a new field or the removal of an existing field, then the transaction version increases.

|===

=== v1 (deprecated) hash calculation

The `INVOKE` v1 transaction hash is calculated as a hash over the given transaction elements, specifically:

```
[,subs="quotes"]
```

```
----
```

```
invoke_v1_tx_hash = _h_(  
    "invoke",  
    version,  
    sender_address,  
    0,  
    _h_(calldata),  
    max_fee,  
    chain_id,  
    nonce  
)  
----
```

Where:

- * `invoke` is a constant prefix string, encoded in ASCII.
- * The placeholder zero is used to align the hash computation for the different types of transactions.
- * `chain_id` is a constant value that specifies the network to which this transaction is sent. See [xref:chain-id\[Chain-Id\]](#).
- * `_h_` is the [xref:../cryptography/hash-functions.adoc#pedersen_hash\[Pedersen\]](#) hash

=== v0 (deprecated) hash calculation

The hash of a v0 `INVOKE` transaction is computed as follows:

```
[,subs="quotes"]
```

```
----
```

```
invoke_v0_tx_hash = _h_(  
    "invoke",  
    version,  
    contract_address,  
    entry_point_selector,  
    _h_(calldata),  
    max_fee,  
    chain_id  
)
```

Where:

- * ``invoke`` is a constant prefix string, encoded in (ASCII).
- * ``chain_id`` is a constant value that specifies the network to which this transaction is sent. See [xref:chain-id\[Chain-Id\]](#). v2 and v3
- * ``_h`` is the [xref:../cryptography/hash-functions.adoc#pedersen_hash\[Pedersen\]](#) hash

`!invoke:`

```
[id="declare-transaction"]  
== `DECLARE` transaction  
:declare:
```

The ``DECLARE`` transaction introduces new contract classes into the state of Starknet, enabling other contracts to deploy instances of those classes or use them in a library call. For more information, see [xref:architecture-and-concepts:smart-contracts/contract-classes.adoc\[contract classes\]](#).

```
// [IMPORTANT]  
// =====  
// You can only declare Cairo classes with v2 and v3 of the `DECLARE` transaction  
// type.  
//  
// v1, which declares Cairo 0 classes, is deprecated and will be removed in a future  
// Starknet version.  
//  
// v0 is no longer supported.  
// =====
```

=== v3 transaction fields

```
include::partial$snippet_transaction_fields_master_table.adoc[]
```

=== v3 hash calculation

The hash of a v3 ``DECLARE`` transaction is computed as follows:

```
[,subs="quotes"]  
----  
declare_v3_tx_hash = _h(  
    "declare",  
    version,  
    sender_address,  
    _h(tip, l1_gas_bounds, l2_gas_bounds),
```

```

    _h_(paymaster_data),
    chain_id,
    nonce,
    data_availability_modes,
    _h_(account_deployment_data),
    class_hash,
    compiled_class_hash
)
----

```

Where:

- * ``declare`` is a constant prefix string, encoded in ASCII.
include::partial\$snippet_transaction_params.adoc[]
- * ``class_hash`` is the hash of the contract class. See xref:smart-contracts/class-hash.adoc#computing_the_cairo_1_class_hash[Class Hash] for details about how the hash is computed
- * ``compiled_class_hash`` is the hash of the xref:starknet-versions:upcoming-versions.adoc#what_to_expect[compiled class] generated by the Sierra->Casm compiler that is used in Starknet

[id="declare_v2"]
 === v2 (deprecated) transaction fields

.`DECLARE` v2 transaction fields

[%autowidth.stretch]
 |===
 | Name | Type | Description

| ``chain_id`` | ``FieldElement`` | The id of the chain to which the transaction is sent.

| ``contract_class`` | ``ContractClass`` | The (Cairo 1.0) xref:smart-contracts/class-hash.adoc#definition_of_a_cairo_1_class[class].

| ``compiled_class_hash`` | ``FieldElement`` | The hash of the compiled class (see xref:starknet-versions:upcoming-versions.adoc#what_to_expect[here] for more information)

| ``sender_address`` | ``FieldElement`` | The address of the account initiating the transaction.

| ``signature`` | ``List<FieldElement>`` | Additional information given by the sender, used to validate the transaction.

| ``max_fee`` | ``FieldElement`` | The maximum fee that the sender is willing to pay for the transaction.

| ``nonce`` | ``FieldElement`` | The transaction nonce.

| ``version`` | ``FieldElement`` | The transaction's version. The value is 2. +

When the fields that comprise a transaction change, either with the addition of a new

field or the removal of an existing field, then the transaction version increases.

|===

=== v2 (deprecated) hash calculation

The hash of a v2 `DECLARE` transaction is computed as follows:

[,subs="quotes"]

```
declare_v2_tx_hash = __h__(
    "declare",
    version,
    sender_address,
    0,
    _h_(class_hash),
    max_fee,
    chain_id,
    nonce,
    compiled_class_hash
)
```

Where:

* `_h_` is the [xref:cryptography/hash-functions.adoc#poseidon_hash](#)[Poseidon hash function]

* `class_hash` is the hash of the contract class. See [xref:smart-contracts/class-hash.adoc#computing_the_cairo_1_class_hash](#)[Class Hash] for details about how the hash is computed

* `compiled_class_hash` is the hash of the [xref:starknet-versions:upcoming-versions.adoc#what_to_expect](#)[compiled class] generated by the Sierra->Casm compiler that is used in Starknet

=== v1 (deprecated) transaction fields

This transaction version was used to declare Cairo 0 classes.

.`DECLARE` v1 transaction fields

[%autowidth.stretch]

|===

Name	Type	Description
------	------	-------------

`contract_class`	`ContractClass`	The class object.
------------------	-----------------	-------------------

`sender_address`	`FieldElement`	The address of the account initiating the transaction.
------------------	----------------	--

| `max_fee` | `FieldElement` | The maximum fee that the sender is willing to pay for the transaction.

| `signature` | `List<FieldElement>` | Additional information given by the sender, used to validate the transaction.

| `nonce` | `FieldElement` | The transaction nonce.

| `version` | `FieldElement` | The transaction's version. Possible values are 1 or 0. +
When the fields that comprise a transaction change, either with the addition of a new field or the removal of an existing field, then the transaction version increases.

|===

=== v1 (deprecated) hash calculation

The hash of a v1 `DECLARE` transaction is computed as follows:

```
[,subs="quotes"]
----
declare_v1_tx_hash = _h_(
    "declare",
    version,
    sender_address,
    0,
    _h_(class_hash),
    max_fee,
    chain_id,
    nonce
)
----
```

Where:

- * `declare` is a constant prefix string, encoded in ASCII.
- * `class_hash` is the hash of the `xref:architecture-and-concepts:smart-contracts/contract-classes.adoc[contract class]`. See `xref:architecture-and-concepts:smart-contracts/class-hash.adoc[Class Hash]` for details about how the hash is computed.
- * The placeholder zero is used to align the hash computation for the different types of transactions.
- * `chain_id` is a constant value that specifies the network to which this transaction is sent. See `xref:#chain-id[Chain-Id]`.
- * `_h_` is the `xref:../cryptography/hash-functions.adoc#pedersen_hash[Pedersen]` hash

=== v0 (unsupported) hash calculation

This transaction version was used to declare Cairo 0 classes.

The hash of a v0 `DECLARE` transaction is computed as follows:

```
[,subs="quotes"]
----
declare_v0_tx_hash = _h_(
    "declare",
    version,
    sender_address,
    0,
    _h_(),
    max_fee,
    chain_id,
    class_hash
)
----
```

Where:

- * `declare` is a constant prefix string, encoded in ASCII.
- * The placeholder zeros are used to align the hash computation for the different types of transactions.
- * `_h_` is the [xref:../cryptography/hash-functions.adoc#pedersen_hash](#)[Pedersen] hash
- * `chain_id` is a constant value that specifies the network to which this transaction is sent. See [xref:chain-id](#)[Chain-Id].
- * `class_hash` is the hash of the [xref:architecture-and-concepts:smart-contracts/contract-classes.adoc](#)[contract class]. See [xref:architecture-and-concepts:smart-contracts/class-hash.adoc](#)[Class Hash] for details about how the hash is computed.

:!declare:

```
[id=deploy_account_transaction]
== `DEPLOY_ACCOUNT` transaction
:deploy_account:
```

Since [xref:starknet-versions:version-notes.adoc#version0.10.1](#)[StarkNet v0.10.1] the `DEPLOY_ACCOUNT` transaction replaces the `DEPLOY` transaction for deploying account contracts.

To use it, you should first pre-fund your new account address so that you can pay the transaction fee. You can then send the `DEPLOY_ACCOUNT` transaction.

For more information, see [xref:accounts/deploying-new-accounts.adoc](#)[].

=== v3 transaction fields

include::partial\$snippet_transaction_fields_master_table.adoc[]

=== v3 hash calculation

The hash of a `DEPLOY_ACCOUNT` transaction is computed as follows:

[,subs="quotes"]

```
deploy_account_v3_tx_hash = _h_(
    "deploy_account",
    version,
    contract_address,
    _h_(tip, l1_gas_bounds, l2_gas_bounds),
    _h_(paymaster_data),
    chain_id,
    nonce,
    data_availability_modes,
    _h_(constructor_calldata),
    class_hash,
    contract_address_salt
)
```

Where:

* `deploy_account` is a constant prefix string, encoded in ASCII.

include::partial\$snippet_transaction_params.adoc[]

* `class_hash` is the hash of the xref:architecture-and-concepts:smart-contracts/contract-classes.adoc[contract class]. See xref:architecture-and-concepts:smart-contracts/class-hash.adoc[Class Hash] for details about how the hash is computed.

* `contract_address` is the address of the newly deployed account. For information on how this address is calculated, see xref:architecture-and-concepts:smart-contracts/contract-address.adoc[Contract address].

=== v1 (deprecated) transaction fields

. `DEPLOY_ACCOUNT` transaction fields

[%autowidth]

|===

| Name | Type | Description

| `class_hash` | `FieldElement` | The hash of the desired account class.

| `constructor_calldata` | `List<FieldElement>` | The arguments to the account constructor.

| ``contract_address_salt`` | ``FieldElement`` | A random salt that determines the
xref:smart-contracts/contract-address.adoc[account address].
| ``signature`` | ``List<FieldElement>`` | Additional information given by the sender, used to
validate the transaction.
| ``max_fee`` | ``FieldElement`` | The maximum fee that the sender is willing to pay for the
transaction
| ``nonce`` | ``FieldElement`` | The transaction nonce.
| ``version`` | ``FieldElement`` | The transaction's version. The value is 1. +

|===

=== v1 (deprecated) hash calculation

The hash of a ``DEPLOY_ACCOUNT`` transaction is computed as follows:

```
[,subs="quotes"]
----
deploy_account_v1_tx_hash = _h_(
    "deploy_account",
    version,
    contract_address,
    0,
    _h_(class_hash, contract_address_salt, constructor_calldata),
    max_fee,
    chain_id,
    nonce
)
----
```

Where:

- * ``deploy_account`` is a constant prefix string, encoded in ASCII.
- * The placeholder zero is used to align the hash computation for the different types of transactions.
- * ``_h_`` is the xref:../cryptography/hash-functions.adoc#pedersen_hash[Pedersen] hash
- * ``chain_id`` is a constant value that specifies the network to which this transaction is sent. See xref:chain-id[Chain-Id].
- * ``class_hash`` is the hash of the xref:architecture-and-concepts:smart-contracts/contract-classes.adoc[contract class]. See xref:architecture-and-concepts:smart-contracts/class-hash.adoc[Class Hash] for details about how the hash is computed.

:!deploy_account:

```
[id="deploy_transaction"]
== `DEPLOY` (unsupported) transaction hash calculation
```

If you need to retrieve the hash of an existing `DEPLOY` transaction, you can use this information to calculate the hash of the transaction.

Before you can calculate the transaction hash, get the deployed contract address. The `DEPLOY` transaction's hash is calculated as shown in the following pseudo code:

```
[,subs="quotes"]
----
deploy_tx_hash = _h_(
    "deploy",
    version,
    contract_address,
    sn_keccak("constructor"),
    _h_constructor_calldata),
    0,
    chain_id
)
----
```

Where:

- * The placeholder zero is used to align the hash computation for the different types of transactions.
- * `deploy` and `constructor` are constant strings encoded in ASCII.
- * `_h_` is the [xref:../cryptography/hash-functions.adoc#pedersen_hash](#)[Pedersen] hash and
- * `sn_keccak` is [xref:../cryptography/hash-functions.adoc#starknet_keccak](#)[Starknet Keccak].
- * `chain_id` is a constant value that specifies the network to which this transaction is sent. See [xref:#chain-id](#)[Chain-Id].
- * `contract_address` is calculated as described [xref:architecture-and-concepts:smart-contracts/contract-address.adoc](#)[here].

```
[id="signature"]
== Signature
```

While Starknet does not have a specific signature scheme built into the protocol, the Cairo language, in which smart contracts are written, does have an efficient implementation for ECDSA signature with respect to a [xref:../cryptography/stark-curve.adoc](#)[STARK-friendly curve].

The generator used in the ECDSA algorithm is $G = (g_x, g_y)$ where:

stem:
[g_x=874739451078007766457464989774322083649278607533249481151382481072868806602] stem:
[g_y=1526667920715188308685755578129483530414204007

80739481342941381225525861407]

[id="v3-fee-estimation"]
== v3 transaction fee estimation

Estimate the fees of transactions with the `starknet_estimateFee` API call, which is part of Starknet's API v0.7.0 and above. For more information, see the link:https://github.com/starkware-libs/starknet-specs/blob/v0.7.1/api/starknet_api_openrpc.json#L612[Starknet JSON RPC specification]. For more information on how to construct the appropriate `resource_bounds` based on the response of `starknet_estimateFee`, see link:<https://community.starknet.io/t/starknet-v0-13-1-pre-release-notes/113664#sdkswallets-how-to-use-the-new-fee-estimates-7>[How to use the new fee estimates?] on the Starknet community forum.

[id="chain-id"]
== Chain ID

Chain IDs are given as numbers, representing the ASCII encoding of specific constant strings, as illustrated by the following Python snippet:

```
[source,python]
----
chain_id = int.from_bytes(value, byteorder="big", signed=False)
----
```

The following constants are currently used. They correspond to the chain IDs that Starknet currently supports:

- * `SN_MAIN` for Starknet's main network.
- * `SN_SEPOLIA` for Starknet's public testnet on Sepolia.

include::ROOT:partial\$snippet_important_goerli_removed.adoc[]
[#nodes]
= Nodes

include::ROOT:partial\$snippet-note-content-from-sn-book.adoc[]

This topic explores the role and functionality of nodes in the Starknet ecosystem, their interactions with sequencers, and their overall importance.

[#overview-of-nodes-in-the-starknet-ecosystem]
== Overview of nodes in the Starknet ecosystem

A node in the Starknet ecosystem is a computer equipped with Starknet software, contributing significantly to the network's operation. Nodes are vital for the Starknet ecosystem's functionality, security, and overall health. Without nodes, the Starknet

network would not be able to function effectively.

Nodes in Starknet are categorized into two types:

- * **Full nodes***: Store the entire Starknet state and validate all transactions, crucial for the network's integrity.

- * **Light nodes***: Do not store the entire Starknet state but rely on full nodes for information. Light nodes are faster and more efficient but offer less security than full nodes.

[#core-functions-of-nodes]

=== Core functions of nodes

Nodes are fundamental to the Starknet network, performing a variety of critical functions:

- * **Transaction validation***: Nodes ensure transactions comply with Starknet's rules, helping prevent fraud and malicious activities.

- * **Block Creation and Propagation***: Nodes create and circulate blocks to maintain a consistent blockchain view across the network.

- * **State maintenance***: Nodes track the Starknet network's current state, including user balances and smart contract code, essential for transaction processing and smart contract execution.

- * **API endpoint provision***: Nodes provide API endpoints, aiding developers in creating applications, wallets, and tools for network interaction.

- * **Transaction relay***: Nodes relay user transactions to other nodes, improving network performance and reducing congestion.

[#interplay-of-nodes-sequencers-clients-and-mempool-in-the-starknet-network]

== Interplay of nodes, sequencers, clients, and the mempool in Starknet

[#nodes-and-sequencers]

=== Nodes and sequencers

Nodes and sequencers are interdependent:

- * **Nodes and block production***: Nodes depend on sequencers to create blocks and update the network state. Sequencers integrate the transactions validated by nodes into blocks, maintaining a consistent and current Starknet state.

- * **Sequencers and transaction validation***: Sequencers rely on nodes for transaction validation and network consensus. Prior to executing transactions, sequencers work with nodes to confirm transaction legitimacy, deterring fraudulent activities. Nodes also contribute to the consensus mechanism, ensuring uniformity in the blockchain state.

[#nodes-and-clients]

=== Nodes and clients

The relationship between nodes and clients in the Starknet ecosystem is characterized by a client-server model:

* *Client requests and node responses*: Clients send requests, like transaction submissions or state queries. Nodes process these requests, validating transactions, updating the network state, and providing clients with the requested data.

* *Client experience*: Clients receive node responses, updating their local view with the latest network information. This loop enables user interaction with Starknet DApps, with nodes maintaining network integrity, while clients provide a user-friendly interface.

[#nodes-and-the-mempool]

=== Nodes and the mempool

The mempool acts as a holding area for unprocessed transactions:

* *Transaction validation and mempool storage*: Upon receiving a transaction, a node validates it. Valid transactions are added to the mempool and broadcast to other network nodes.

* *Transaction selection and block inclusion*: Nodes select transactions from the mempool for processing, incorporating them into blocks that are added to the blockchain.

[#node-implementations-in-starknet]

== Node implementations in Starknet

Each Starknet node implementation has its own strengths:

* <https://github.com/eqlabs/pathfinder> [*Pathfinder*], by Equilibrium: Pathfinder is a full node written in Rust. Pathfinder excels in high performance, scalability, and aligns with the Starknet Cairo specification.

* <https://github.com/NethermindEth/juno> [*Juno*], by Nethermind: Juno, is a full node written in Golang. Juno is known for user-friendliness, ease of deployment, and compatibility with Ethereum tools.

* <https://github.com/starkware-libs/papyrus> [*Papyrus*], by StarkWare: Papyrus is also a full node written in Rust. Papyrus focuses on security and robustness. It's integral to the upcoming Starknet Sequencer, expected to boost network throughput.

These implementations are continuously being improved, with new features and enhancements. The choice of implementation depends on user or developer preferences and requirements.

Key characteristics of each node implementation are summarized below:

|===

|Node Implementation |Language |Strengths

```
|Pathfinder |Rust |High performance, scalability, Cairo specification adherence
|Papyrus |Rust |Security, robustness, Starknet Sequencer foundation
|Juno |Golang |User-friendliness, ease of deployment, Ethereum compatibility
|===
[id="provers"]
= Provers
:down_arrow: &#65516;
```

```
include::ROOT:partial$snippet-note-content-from-sn-book.adoc[]
```

The only Prover in use on Starknet as of this writing is SHARP.

SHARP is like public transportation for proofs on Starknet, aggregating multiple Cairo programs to save costs and boost efficiency. It uses recursive proofs, enabling parallelization and optimization, making it more affordable for all users. Critical services like the gateway, validator, and Prover work together with a stateless design for flexibility. SHARP's adoption by StarkEx and Starknet highlights its significance and potential for future optimization.

This topic discusses SHARP, how it has evolved to incorporate recursive proofs, and its role in reducing costs and improving efficiency within the Starknet network.

```
[#what-is-sharp]
== What is SHARP?
```

SHARP, which stands for `_Shared Prover_`, is a mechanism that aggregates multiple Cairo programs from different users, each containing different logic. These Cairo programs are then executed together, generating a single proof common to all the programs. Rather than sending the proof directly to the Solidity Verifier in Ethereum, it is initially sent to a STARK Verifier program written in Cairo. The STARK Verifier generates a new proof to confirm that the initial proofs were verified, which can be sent back into SHARP and the STARK Verifier. Details for this recursive proof process appear below. Ultimately, the last proof in the series is sent to the Solidity Verifier on Ethereum. In other words, there are many proofs generated until we reach Ethereum and the Solidity Verifier.

The primary benefit of SHARP lies in its ability to decrease costs and enhance efficiency within the Starknet network. It achieves this by aggregating multiple Cairo jobs, which are individual sets of computation. This aggregation enables the protocol to leverage the exponential amortization offered by STARK proofs.

Exponential amortization means that as the computational load of the proofs increases, the cost of verifying those proofs rises at a slower logarithmic rate than the computation increase. As a result, the cost of each transaction within the aggregated set is significantly reduced, making the overall process more cost-effective and accessible for

users.

[NOTE]

====

In the context of SHARP and Cairo context, a `_jobs_` refer to the individual Cairo programs or tasks submitted by different users. These jobs contain specific logic or computations that must be executed on the Starknet network.

====

Additionally, SHARP enables smaller users with limited computation to benefit from joining other jobs and share the cost of generating the proofs. This collaborative approach is similar to using public transportation instead of a private car, where the cost is distributed among all participants, making it more affordable for everyone.

[#recursive-proofs-in-sharp]

== Recursive proofs in SHARP

One of the most powerful features of SHARP is its use of recursive proofs. Rather than directly sending the generated proofs to the Solidity Verifier, they are first sent to a STARK Verifier program written in Cairo. This Verifier, which is also a Cairo program, receives the proof and creates a new Cairo job that is sent to the Prover. The Prover then generates a new proof to confirm that the initial proofs were verified. These new proofs can be sent back into SHARP and the STARK Verifier, restarting the process.

This process continues recursively, with each new proof being sent to the Cairo Verifier until a trigger is reached. At this point, the last proof in the series is sent to the Solidity Verifier on Ethereum. This approach enables greater parallelization of the computation and reduces the time and cost associated with generating and verifying proofs.

// [.text-center]

|===

^|*Generated Proofs*

{down_arrow}

STARK Verifier program (in Cairo)

{down_arrow}

Cairo Job

{down_arrow}

Prover

{down_arrow}

New Proof Generated

{down_arrow}

Repeat Process

{down_arrow}

Trigger Reached (last proof)

{down_arrow}

Solidity Verifier

|===

At first glance, recursive proofs may seem more complex and time-consuming. However, there are several benefits to this approach:

- . ***Parallelization***: Recursive proofs enable work parallelization, reducing user latency and improving SHARP efficiency.
- . ***Cheaper onchain costs***: Parallelization enables SHARP to create larger proofs, which were previously dependent on the limited availability of large cloud machines. As a result, onchain costs are reduced.
- . ***Lower cloud costs***: Since each job is shorter, the required memory for processing is reduced, resulting in lower cloud costs.
- . ***Optimization***: Recursive proofs enable SHARP to optimize for various factors, including latency, onchain costs, and time to proof.
- . ***Cairo support***: Recursive proofs only require support in Cairo, without the need to add support in the Solidity Verifier.

[NOTE]

====

Latency in Starknet encompasses the time taken for processing, confirming, and including transactions in a block. Latency is affected by factors like network congestion, transaction fees, and system efficiency. Minimizing latency ensures faster transaction processing and user feedback.

Time to proof, however, specifically pertains to the amount of time required to generate and verify cryptographic proofs for transactions or operations.

====

[#sharp-backend-architecture-and-data-pipeline]

== SHARP backend architecture and data pipeline

SHARP's backend architecture consists of several services that work together to process Cairo jobs and generate proofs. These services include:

- . ***Gateway***: Cairo jobs enter SHARP through the gateway.
- . ***Job Creator***: This service prevents job duplication and ensures that the system operates consistently, regardless of multiple identical requests.
- . ***Validator***: The validator service runs validation checks on each job, ensuring they meet the requirements and can fit within the Prover machines. Invalid jobs are tagged as such and do not proceed to the Prover.
- . ***Scheduler***: The scheduler service creates `_trains_` that aggregate jobs and send them to the Prover. Recursive jobs are paired and sent to the Prover together.
- . ***Cairo Runner***: This service runs Cairo for the Prover's needs. The Cairo Runner service runs Cairo programs, executing the necessary computations and generating the execution trace as an intermediate result. The Prover then uses this execution trace.
- . ***Prover***: The Prover computes the proofs for each train.
- . ***Dispatcher***: The Dispatcher serves two functions in the SHARP system.
 - * In the case of a recursive proof, the Dispatcher runs the Cairo Verifier program on the proof it has received from the Prover, resulting in a new Cairo job that goes back to the Validator.
 - * In the case of a proof that is to be published onchain, the Dispatcher creates `_packages_` from the proof, which can then be sent to the `_Blockchain Writer_`.
- . ***Blockchain Writer***: Once the packages have been created by the Dispatcher, they are sent to the Blockchain Writer. The Blockchain Writer is responsible for sending the packages to the appropriate blockchain, such as Ethereum, for verification. This step in the SHARP system ensures that the proofs are properly verified and that the transactions are securely recorded on the blockchain.
- . ***Catcher***: The Catcher monitors onchain transactions to ensure that they have been accepted. While the Catcher is relevant for internal monitoring purposes, be aware that if a transaction fails, the fact won't be registered onchain in the Fact Registry. As a result, the soundness of the system is still preserved even without the Catcher.

SHARP is designed to be stateless. That is, each Cairo job is executed in its own context and has no dependency on other jobs, enabling greater flexibility in processing jobs.

[#current-sharp-users]
== Current SHARP users

Currently, the primary users of SHARP include:

- * StarkEx
- * Starknet

[#challenges-and-optimization]
== Challenges and optimization

Optimizing the Prover involves the numerous challenges and potential projects on which the Starkware team and the community are currently working, including:

- * Exploring more efficient hash functions for Cairo, the Prover, and Solidity.
- * Investigating smaller fields for recursive proof steps could lead to more efficient computations.
- * Adjusting various parameters of the STARK protocol, such as FRI parameters and block factors.
- * Optimizing the Cairo code to make it faster, resulting in a faster recursive Prover.
- * Developing dynamic layouts, which should enable Cairo programs to scale resources as needed.
- * Improving scheduling algorithm. This optimization path is external to the Prover.

Dynamic layouts enable SHARP to determine and scale the required resources for a specific job and adjust the layout accordingly, instead of relying on predefined layouts with fixed resources. Scaling resources can lead to more efficient computation and better resource utilization. This approach can provide tailored solutions for each job, improving overall efficiency.

[#conclusion]
== Conclusion

In conclusion, SHARP is a critical component of Starknet's architecture, providing a more efficient and cost-effective solution for processing Cairo programs and verifying their proofs. By leveraging the power of STARK technology and incorporating recursive proofs, SHARP plays a vital role in improving the overall performance and scalability of the Starknet network. The stateless nature of SHARP and the reliance on the cryptographic soundness of the STARK proving system make it an innovative and valuable addition to the blockchain ecosystem.

[id="sierra"]
= Cairo and Sierra

Before `xref:starknet-versions:upcoming-versions.adoc` [Starknet Alpha v0.11.0] a developer would write contracts in Cairo 0 and compile them locally to Cairo assembly (or Casm for short).

Next, the developer would submit the compilation output, the `xref:smart-contracts/contract-classes.adoc` [contract class], to the Starknet sequencer via a `DECLARE` transaction.

Starting with Cairo 1.0, the contract class resulting from `xref:smart-contracts/class-hash.adoc#cairo1_class` [compiling Cairo 1.0] does not include Casm. Instead of Casm, it includes instructions in an intermediate representation called `**S**afe
Int**er**mediate **R**epresent**a**tion, _Sierra_` for short.

This new contract class is then compiled by the sequencer, via the Sierra & Casm compiler, to generate the Cairo assembly associated with this class. The Casm code is then executed by the Starknet OS.

== Why do we need Casm?

Starknet is a validity rollup, which means that the execution inside every block needs to be proven, and this is where STARKs come in handy.

However, STARK proofs can address statements that are formulated in the language of polynomial

constraints, and have no knowledge of smart contract execution.

To overcome this gap, we developed Cairo.

Cairo instructions, previously referred to as Casm, are translated to polynomial constraints that enforce the correct execution of a program according to the Cairo semantics defined in [link:https://github.com/starknet-io/starknet-stack-resources/blob/main/Cairo/Cairo%20%E2%80%93%20a%20Turing-complete%20STARK-friendly%20CPU%20architecture.pdf](https://github.com/starknet-io/starknet-stack-resources/blob/main/Cairo/Cairo%20%E2%80%93%20a%20Turing-complete%20STARK-friendly%20CPU%20architecture.pdf)[_Cairo-a Turing-complete STARK-friendly CPU architecture_].

Thanks to Cairo, we can formulate the statement "This Starknet block is valid" in a way that we can prove.

Be aware that we can only prove things about Casm. That is, regardless of what the user sends to the Starknet sequencer, what's proven is the correct Casm execution.

This means that we need a way to translate Sierra into Casm, and this is achieved with the Sierra -> Casm compiler.

== Why do we need Sierra?

To understand why we chose to add an additional layer between the code that the user writes (Cairo 1.0) and the code that is being proven (Casm), we need to consider more components in the system, and the limitations of Cairo.

=== Reverted transactions, unsatisfiable AIRs, and DoS attacks

A crucial property of every decentralized L2 is that the sequencers are guaranteed to be compensated for the work they do.

The notion of reverted transactions is a good example: even if the user's transaction failed mid execution, the sequencer should be able to include it in a block and charge execution fees up to the point of failure.

If the sequencer cannot charge for such transactions, then sending transactions that will eventually fail (after a lot of computation steps) is an obvious DoS attack on the sequencer.

The sequencer cannot look at a transaction and conclude that it would fail without actually doing the work (this is equivalent to solving the halting problem).

The obvious solution to the above predicament is to include such transactions in the block, similar to Ethereum. However, this may not be as simple to do in a validity rollup. With Cairo 0, there is no separating layer between user code and what is being proven.

This means that users can write code which is unprovable in some cases. In fact, such code is very easy to write, e.g. ``assert 0=1`` is a valid Cairo instruction that cannot be proven, as it translates to polynomial constraints that are not satisfiable. Any Casm execution that contains this instruction cannot be proven. Sierra is the layer between user code and the provable statement, that allows us to make sure all transactions are eventually provable.

=== Safe Casm

The method by which Sierra guarantees that user code is always provable is by compiling Sierra instructions to a subset of Casm, which we call "safe Casm". The important property that we require from safe Casm is being provable for all inputs. A canonical example for safe Casm is using ``if/else`` instructions instead of ``assert``, that is, making sure all failures are graceful.

To better understand the considerations that go into designing the Sierra \leftrightarrow Casm compiler, consider the ``find_element`` function from the common library of Cairo 0:

[source,cairo]

```
func find_element{range_check_ptr}(array_ptr: felt*, elm_size, n_elms, key) -> (elm_ptr:
felt*) {
    alloc_locals;
    local index;
    %{
        ...
    %}
    assert_nn_le(a=index, b=n_elms - 1);
    tempvar elm_ptr = array_ptr + elm_size * index;
    assert [elm_ptr] = key;
    return (elm_ptr=elm_ptr);
}
```

[NOTE]

=====

Below we abuse the "Casm" notation by not distinguishing Cairo 0 from Casm and referring to the above as Casm (while we actually refer to the compilation result of the above).

=====

For brevity, we have omitted the hint in the above snippet, but it's clear that this function can only execute correctly if the requested element exists in the array (otherwise it would fail for every possible hint - there is nothing we can substitute `index` for, that makes the following lines run successfully).

Such Casm cannot be generated by the Sierra->Casm compiler. Furthermore, simply replacing the assertion with an if/else statement doesn't do, as this results in non-deterministic execution. That is, for the same input, different hint values can yield different results. A malicious prover can use this freedom to harm the user - in this example, they are able to make it seem as if an element isn't part of the array, even though it actually is.

The safe Casm for finding an element in an array behaves like the above snippet in the happy flow (element is there): an index is given in a hint, and we verify that the array at the hinted index contains the requested element. However, in the unhappy flow (element isn't there), we *must* go over the entire array to verify this.

This was not the case in Cairo 0, as we were fine with certain paths not being provable (in the above snippet, the unhappy flow in which the element isn't in the array is never provable).

[NOTE]

====

Sierra's gas metering adds further complications to the above example. Even looking through the array to verify that the element isn't there may leave some flexibility to the prover.

If we take gas limitations into consideration, the user may have enough gas for the happy flow, but not for the unhappy one, making the execution stop mid-search, and allowing the prover to get away with lying about the element not being present.

The way we plan to handle this is by requiring the user to have enough gas for the unhappy flow before actually calling `find_element`.

====

=== Hints in Cairo 1.0

Smart contracts written with Cairo 1.0 cannot contain user defined hints. This is already true with Cairo 0 contracts (only whitelisted hints are accepted), but with Cairo 1.0 the hints in use are determined by the Sierra & Casm compiler. Since this compilation is there to ensure that only "safe" Casm is generated, there is no room for hints that are not generated by the

compiler.

In the future, native Cairo 1.0 may contain hint syntax similar to Cairo 0, but it will not be available in Starknet smart contracts (link:<https://medium.com/starkware/fractal-scaling-from-l2-to-l3-7fe238ecfb4f>[L3s] on top of Starknet may make use of such functionality).

Note that this is currently not part of Starknet's roadmap.

[id="cairo-builtins"]

= Cairo builtins

`_Builtins_` in Cairo are predefined optimized low-level execution units that the Cairo VM refers to in order to perform predefined computations that are expensive to perform in standard Cairo. Builtins enhance the functionality of the Cairo VM, enabling you to perform certain tasks, such as using the Poseidon hash, range-checks, or ECDSA signature verifications, more efficiently, using fewer trace cells.

In contrast to CairoZero, where you needed to consciously write code to take advantage of builtin optimizations, in Cairo, you simply write code without doing anything special, and when the Cairo VM executes the code, certain operations use builtins internally to optimize your program.

[#list-of-cairo-builtins]

.List of Cairo builtins

[cols="1,2",]

|===

|Name of builtin | Description

|Pedersen | Computes the Pedersen hash over two elements. Used internally in ``pedersen.cairo``. For more information see xref:architecture-and-concepts:cryptography/hash-functions.adoc[].

|Poseidon | Computes the Hades permutation on three field elements. Used internally in ``poseidon.cairo``. For more information, see xref:architecture-and-concepts:cryptography/hash-functions.adoc[]. The Cairo corelib functions use this builtin internally. The Cairo corelib functions are defined in link:<https://github.com/starkware-libs/cairo/blob/v2.6.0/corelib/src/starknet/info.cairo>[`info.cairo`] in the Cairo GitHub repository.

|Range check a|

Checks whether a field element is in the range $[0, 2^{128}-1]$.

Used when instantiating and comparing the various integer types.

All arithmetic comparisons use the range check builtin.

|ECDSA | Verifies the validity of an ECDSA signature over the STARK curve.

This is used in CairoZero, but is not used in Cairo because it fails on invalid signatures. In Cairo ECDSA verification is performed with high-level code, applying the `EC_OP` builtin twice.

|Keccak | Computes the keccak-f[1600] permutation. For more information see [link:https://keccak.team/keccak.html\[_Keccak_\]](https://keccak.team/keccak.html[_Keccak_]) page on the _Keccak Team_ site.

For high level Cairo keccak functions that use this builtin internally, see [link:https://github.com/starkware-libs/cairo/blob/main/corelib/src/keccak.cairo#L62\[_keccak.cairo_\]](https://github.com/starkware-libs/cairo/blob/main/corelib/src/keccak.cairo#L62[_keccak.cairo_]) in the Cairo corelib.

|Bitwise | Computes the bitwise operations `OR`, `AND`, and `XOR` of two felts.

Used internally when performing bitwise operations using the `|`, `&` and `^` operators.

|EC_OP |Multiplies a point on the STARK curve by a scalar.

|===

[id="contract_hash"]

= Class hash

:description: A Cairo class hash is a hash of the components that define a Cairo class: `contract_class_version`, `external_entry_points`, `l1_handler_entry_points`, `constructor_entry_points`, `abi_hash`, and `sierra_program_hash`

:keywords: class hash, Cairo class hash, Starknet class hash, Starknet contract class hash

The class hash is a hash chain of the components that define the class.

Classes written in Cairo are compiled into Sierra code. The Sierra code generated is an intermediate representation of the class. This new contract class is then compiled by the sequencer, via the Sierra -> Casm compiler, to generate the Cairo assembly associated with this class. The resulting Casm code is then executed by the Starknet OS.

For information on how the compiler converts code from Cairo to Sierra, see [xref:smart-contracts/cairo-and-sierra.adoc\[Cairo and Sierra\]](#).

[id="cairo1_class"]

== Components of a Cairo class definition

The components that define a class are:

[horizontal,labelwidth=35]

contract_class_version:: The version of the contract class object. Currently, the Starknet OS

supports version 0.1.0

Array of external functions entry points:: An entry point is a pair `(_selector_,

`_function_idx_``, where ``_function_idx_`` is the index of the function inside the Sierra program.

+

[NOTE]

====

The selector is an identifier through which the function is callable in transactions or in other classes. The selector is the `xref:../cryptography/hash-functions.adoc#starknet_keccak[starknet_keccak]` hash of the function name, encoded in ASCII.

====

Array of `xref:architecture-and-concepts/network-architecture/messaging-mechanism.adoc#l1-l2-message-fees[l1 handlers]` entry points :: -

Array of constructors entry points:: Currently, the compiler allows only one constructor.

ABI:: A string representing the ABI of the class. The ABI hash (which affects the class hash) is given by:

+

[source,python]

```
starknet_keccak(bytes(ABI, "UTF-8"))
```

+

[NOTE]

====

This string is supplied by the user declaring the class (and is signed on as part of the ``DECLARE`` transaction), and is not enforced to be the true ABI of the associated class. Without seeing the underlying source code (i.e. the Cairo code generating the class's Sierra), this ABI should be treated as the "intended" ABI by the declaring party, which may be incorrect (intentionally or otherwise).

The "honest" string would be the json serialization of the contract's ABI as produced by the Cairo compiler.

====

Sierra program:: An array of field elements representing the Sierra instructions.

[id="computing_the_cairo_1_class_hash"]

== Computing the Cairo class hash

The hash of the class is the chain hash of its components, computed as follows:

[source,cairo]

```
class_hash = ! (
    contract_class_version,
    external_entry_points,
    l1_handler_entry_points,
```

```

    constructor_entry_points,
    abi_hash,
    sierra_program_hash
)
----
```

Where

- * stem:[\$h\$] is the xref:../cryptography/hash-functions.adoc#poseidon_hash[Poseidon] hash function
- * The hash of an entry point array stem:[\$(selector,index)_{i=1}^n\$] is given by stem: [\$h(\text{selector}_1,\text{index}_1,...,\text{selector}_n,\text{index}_n)\$]
- * The `sierra_program_hash` is the xref:../cryptography/hash-functions.adoc#poseidon_hash[Poseidon] hash of the bytecode array

[NOTE]

====

The Starknet OS currently supports contract class version 0.1.0, which is represented in the above hash computation as the ASCII encoding of the string `CONTRACT_CLASS_V0.1.0` (hashing the version in this manner gives us domain separation between the hashes of classes and other objects).

====

For more details, see the <https://github.com/starkware-libs/cairo-lang/blob/7712b21fc3b1cb02321a58d0c0579f5370147a8b/src/starkware/starknet/core/os/contracts.cairo#L47>[Cairo implementation].
[id="compiled_class_hash"]
= Compiled class hash

The compiled class hash is a cryptographic hash that results from the compilation process of a Cairo class

from its intermediate representation (Sierra) to Cairo assembly (Casm). This process is managed by the Sierra!Casm compiler.

The compiled class hash is crucial for ensuring the uniqueness and integrity of compiled classes within Starknet. Whether you are a developer deploying contracts or a party interested in the inner workings of Starknet's state commitment, understanding the compiled class hash is essential.

For developers, the hash is an important part of the contract declaration process, ensuring that each compiled class is uniquely identifiable and verifiable. For those involved in maintaining the network, it contributes to the efficiency and performance of Starknet by optimizing the state commitment process.

The state commitment uses the Sierra code that results when compiling Cairo classes. Sierra acts as an intermediate representation between Cairo and Casm. Provers, however, operate solely with Casm.

In order to avoid recompiling, from Sierra to Casm, each block in which the class is deployed, the state commitment gets the information it needs about the corresponding Casm from the the information contained in the compiled class hash.

When declaring a contract, the party administering the contract endorses the compiled class hash, procured using an SDK, as an integral component of the `xref:network-architecture/transactions.adoc#declare_v2[`DECLARE`]` transaction. Following the inclusion of the transaction in a block, the compiled class hash integrates into the state commitment.

== Purpose and Significance

- * **Uniqueness:** The compiled class hash ensures the uniqueness of each compiled class. It is essentially a fingerprint for the compiled output, allowing the network to verify the integrity and uniqueness of the class.
- * **State Commitment:** In Starknet, state commitment includes various components, including the Cairo classes. These classes are initially defined using Sierra. However, for the prover to function efficiently, it requires Casm.
- * **Efficiency:** By including the compiled class hash in the state commitment, Starknet avoids the need to recompile from Sierra to Casm in every block where the class is used. This optimization significantly enhances the network's efficiency and performance.

== Usage

When a new contract is declared on Starknet, the compiled class hash plays a pivotal role. Here's how:

- * **Declaration Process:** The party declaring the contract computes the compiled class hash using an SDK provided by Starknet.
- * **DECLARE Transaction:** This hash is then included as part of the `xref:network-architecture/transactions.adoc#declare_v2[`DECLARE`]` transaction is a specific type of transaction in Starknet used to register new contracts.
- * **Inclusion in State Commitment:** Once the `xref:network-architecture/transactions.adoc#declare_v2[`DECLARE`]` transaction is included in a block, the compiled class hash becomes part of the state commitment. This inclusion ensures that the network recognizes and stores the unique compiled output of the contract.

Prospectively, as Sierra-to-Casm compilation integrates into the Starknet OS, the value might undergo updates via proof of the Sierra-to-Casm compiler execution. Such verification demonstrates that compiling the same class with an updated compiler version yields a fresh compiled class hash.

The compiled class hash is a basic element in Starknet's architecture, enabling efficient state commitment and ensuring the integrity and uniqueness of compiled classes.

```
[id="contract_abi"]
```

```
= Contract ABI
```

== Introduction

A contract ABI is a representation of a Starknet contract interface. It is formatted as JSON and describes the functions, structs and events which are defined in the contract.

You can get the contract's ABI by using `starknet-compile`:

```
[source,bash]
```

```
----
```

```
cargo run --bin starknet-compile -- --single-file </path/to/input.cairo> </path/to/output.json>
```

```
----
```

== An example contract ABI

The following is an example contract ABI:

```
[tabs]
```

```
=====
```

```
Cairo v2::
```

```
+
```

```
[source,json]
```

```
----
```

```
[
```

```
{
```

```
  "type": "impl",
```

```
  "name": "CounterContract",
```

```
  "interface_name":
```

```
"new_syntax_test_contract::new_syntax_test_contract::ICounterContract"
```

```
},
```

```
{
```

```
  "type": "interface",
```

```
  "name": "new_syntax_test_contract::new_syntax_test_contract::ICounterContract",
```

```
  "items": [
```

```

{
  "type": "function",
  "name": "increase_counter",
  "inputs": [
    {
      "name": "amount",
      "type": "core::integer::u128"
    }
  ],
  "outputs": [],
  "state_mutability": "external"
},
{
  "type": "function",
  "name": "decrease_counter",
  "inputs": [
    {
      "name": "amount",
      "type": "core::integer::u128"
    }
  ],
  "outputs": [],
  "state_mutability": "external"
},
{
  "type": "function",
  "name": "get_counter",
  "inputs": [],
  "outputs": [
    {
      "type": "core::integer::u128"
    }
  ],
  "state_mutability": "view"
}
]
},
{
  "type": "constructor",
  "name": "constructor",
  "inputs": [
    {
      "name": "initial_counter",
      "type": "core::integer::u128"
    }
  ],
  {

```

```

        "name": "other_contract_addr",
        "type": "core::starknet::contract_address::ContractAddress"
    }
]
},
{
    "type": "event",
    "name": "new_syntax_test_contract::new_syntax_test_contract::counter_contract::CounterIncreased",
    "kind": "struct",
    "members": [
        {
            "name": "amount",
            "type": "core::integer::u128",
            "kind": "data"
        }
    ]
},
{
    "type": "event",
    "name": "new_syntax_test_contract::new_syntax_test_contract::counter_contract::CounterDecreased",
    "kind": "struct",
    "members": [
        {
            "name": "amount",
            "type": "core::integer::u128",
            "kind": "data"
        }
    ]
},
{
    "type": "event",
    "name": "new_syntax_test_contract::new_syntax_test_contract::counter_contract::Event",
    "kind": "enum",
    "variants": [
        {
            "name": "CounterIncreased",
            "type": "new_syntax_test_contract::new_syntax_test_contract::counter_contract::CounterIncreased",
            "kind": "nested"
        },
        {
            "name": "CounterDecreased",
            "type": "new_syntax_test_contract::new_syntax_test_contract::counter_contract::CounterDecreased"
        }
    ]
}

```

```
    counterDecreased",
      "kind": "nested"
    }
  ]
}
]
```

Cairo v1::

+

[source,json]

```
[
  {
    "type": "function",
    "name": "constructor",
    "inputs": [
      {
        "name": "name_",
        "type": "core::felt252"
      },
      {
        "name": "symbol_",
        "type": "core::felt252"
      },
      {
        "name": "decimals_",
        "type": "core::integer::u8"
      },
      {
        "name": "initial_supply",
        "type": "core::integer::u256"
      },
      {
        "name": "recipient",
        "type": "core::starknet::contract_address::ContractAddress"
      }
    ],
    "outputs": [],
    "state_mutability": "external"
  },
  {
    "type": "function",
    "name": "get_name",
    "inputs": [],
    "outputs": [
      {

```

```

        "type": "core::felt252"
    }
],
"state_mutability": "view"
},
{
    "type": "function",
    "name": "get_symbol",
    "inputs": [],
    "outputs": [
        {
            "type": "core::felt252"
        }
    ],
    "state_mutability": "view"
},
{
    "type": "function",
    "name": "get_decimals",
    "inputs": [],
    "outputs": [
        {
            "type": "core::integer::u8"
        }
    ],
    "state_mutability": "view"
},
{
    "type": "function",
    "name": "get_total_supply",
    "inputs": [],
    "outputs": [
        {
            "type": "core::integer::u256"
        }
    ],
    "state_mutability": "view"
},
{
    "type": "function",
    "name": "balance_of",
    "inputs": [
        {
            "name": "account",
            "type": "core::starknet::contract_address::ContractAddress"
        }
    ]
}

```

```

],
"outputs": [
  {
    "type": "core::integer::u256"
  }
],
"state_mutability": "view"
},
{
  "type": "function",
  "name": "allowance",
  "inputs": [
    {
      "name": "owner",
      "type": "core::starknet::contract_address::ContractAddress"
    },
    {
      "name": "spender",
      "type": "core::starknet::contract_address::ContractAddress"
    }
  ],
  "outputs": [
    {
      "type": "core::integer::u256"
    }
  ],
  "state_mutability": "view"
},
{
  "type": "function",
  "name": "transfer",
  "inputs": [
    {
      "name": "recipient",
      "type": "core::starknet::contract_address::ContractAddress"
    },
    {
      "name": "amount",
      "type": "core::integer::u256"
    }
  ],
  "outputs": [],
  "state_mutability": "external"
},
{
  "type": "function",

```

```
"name": "transfer_from",
"inputs": [
  {
    "name": "sender",
    "type": "core::starknet::contract_address::ContractAddress"
  },
  {
    "name": "recipient",
    "type": "core::starknet::contract_address::ContractAddress"
  },
  {
    "name": "amount",
    "type": "core::integer::u256"
  }
],
"outputs": [],
"state_mutability": "external"
},
{
  "type": "function",
  "name": "approve",
  "inputs": [
    {
      "name": "spender",
      "type": "core::starknet::contract_address::ContractAddress"
    },
    {
      "name": "amount",
      "type": "core::integer::u256"
    }
  ],
  "outputs": [],
  "state_mutability": "external"
},
{
  "type": "function",
  "name": "increase_allowance",
  "inputs": [
    {
      "name": "spender",
      "type": "core::starknet::contract_address::ContractAddress"
    },
    {
      "name": "added_value",
      "type": "core::integer::u256"
    }
  ]
}
```

```

    ],
    "outputs": [],
    "state_mutability": "external"
  },
  {
    "type": "function",
    "name": "decrease_allowance",
    "inputs": [
      {
        "name": "spender",
        "type": "core::starknet::contract_address::ContractAddress"
      },
      {
        "name": "subtracted_value",
        "type": "core::integer::u256"
      }
    ],
    "outputs": [],
    "state_mutability": "external"
  },
  {
    "type": "event",
    "name": "Transfer",
    "inputs": [
      {
        "name": "from",
        "type": "core::starknet::contract_address::ContractAddress"
      },
      {
        "name": "to",
        "type": "core::starknet::contract_address::ContractAddress"
      },
      {
        "name": "value",
        "type": "core::integer::u256"
      }
    ]
  },
  {
    "type": "event",
    "name": "Approval",
    "inputs": [
      {
        "name": "owner",
        "type": "core::starknet::contract_address::ContractAddress"
      }
    ],
  },

```



```

    {
      "name": "spender",
      "type": "core::starknet::contract_address::ContractAddress"
    },
    {
      "name": "value",
      "type": "core::integer::u256"
    }
  ]
}
]

```

=====

== Cairo v2.3.0 changes

=== Nested events

With Cairo `v2.3.0` the limitations on the `Event` enum have been relaxed, allowing more flexibility on the events that can be emitted from a given contract.

For example:

- * It is no longer enforced that the `Event` enum variants are structs of the same name as the variant, they can now be a struct or an enum of any name.
- * Enum variants inside event ABI entries (entries in the abi with `"type": "event"` and `"kind": "enum"`) now have two possible kinds. Before `v2.3.0` it was always `"kind": "nested"`, now `"kind": "flat"` is also possible.
- * `v2.3.0` is backward compatible with version "e `2.0.0` ABI, so the same structure of the ABI is kept, while allowing flexibility.

[NOTE]

=====

Between versions `v2.0.0` and `v2.2.0`, to identify all potential serializations of events (what raw `keys`, `data` arrays can be emitted given the ABI), it was sufficient to iterate over the abi entries with `"type": "event"` and `"kind": "struct"`, skipping the encapsulating `Event` type which has `"kind": "enum"`.

With `v2.3.0` onwards, doing so may result in losing information.

=====

To illustrate this, consider the following example:

[source,cairo]

//high-level code defining the events

```
#[event]
#[derive(Drop, starknet::Event)]
enum Event {
    ComponentEvent: test_component::Event,
    TestCounterIncreased: CounterIncreased,
    TestCounterDecreased: CounterDecreased,
    TestEnum: MyEnum
}
```

```
#[derive(Drop, starknet::Event)]
struct CounterIncreased {
    amount: u128
}
```

```
#[derive(Drop, starknet::Event)]
struct CounterDecreased {
    amount: u128
}
```

```
#[derive(Copy, Drop, starknet::Event)]
enum MyEnum {
    Var1: MyStruct
}
```

```
#[derive(Copy, Drop, Serde, starknet::Event)]
struct MyStruct {
    member: u128
}
```

=== Variant names different from types

In `v2.3.0` enum variant types can now have any name.

As an example the `TestCounterIncreased` variant and the `CounterIncreased` type, as they appear in the ABI:

```
[source,json]
----
{
  "type": "event",
  "name": "<namespace>::Event",
  "kind": "enum",
  "variants": [
    {
```

```

    "name": "ComponentEvent",
    "type": "<namespace>::test_component::Event",
    "kind": "nested"
  },
  {
    "name": "TestCounterIncreased",
    "type": "<namespace>::CounterIncreased",
    "kind": "nested"
  },
  {
    "name": "TestCounterDecreased",
    "type": "<namespace>::CounterDecreased",
    "kind": "nested"
  },
  {
    "name": "TestEnum",
    "type": "<namespace>::MyEnum",
    "kind": "nested"
  }
]
},
{
  "type": "event",
  "name": "<namespace>::CounterIncreased",
  "kind": "struct",
  "members": [
    {
      "name": "amount",
      "type": "core::integer::u128",
      "kind": "data"
    }
  ]
}
]
}
----

```

When the contract emits the ``TestCounterIncreased`` event, for example by writing ``self.emit(CounterIncreased { amount })``, the event that is emitted has the following keys and data:

- * One key based on the variant name: ``sn_keccak(TestCounterIncreased)``. This information only appears in the ``<namespace>::Event`` type entry in the ABI, as the name ``TestCounterIncreased`` does not appear in the ``"kind": "struct"`` ABI entry. This did not matter in previous versions when the variant name and type had to be equal.

- * One data element based on the struct ``CounterIncreased`` which is associated with ``TestCounterIncreased`` via one of the ``Event`` type variants.

=== Enum variants inside Event

The introduction of components allows variants of `Event` to be enums. In the following example, we have two such variants: `TestEnum` (unrelated to components) and `ComponentEvent`.

The serialization to keys and data is the same in both cases, so this example will focus on `TestEnum`:

This example shows the `TestEnum` variant entry inside Event:

```
[source,json]
----
{
  "name": "TestEnum",
  "type": "<namespace>::MyEnum",
  "kind": "nested"
}
----
```

This example shows the `MyEnum` event entry:

```
[source,json]
----
{
  "type": "event",
  "name": "<namespace>::MyEnum",
  "kind": "enum",
  "variants": [
    {
      "name": "Var1",
      "type": "<namespace>::MyStruct",
      "kind": "nested"
    }
  ]
}
----
```

This example shows the `MyStruct` event entry:

```
[source,json]
----
{
  "type": "event",
  "name": "<namespace>::MyStruct",

```

```

    "kind": "struct",
    "members": [
      {
        "name": "member",
        "type": "core::integer::u128",
        "kind": "data"
      }
    ]
  }
}
----
```

[NOTE]

=====

If a ``TestEnum`` event is being emitted via ``self.emit(Event::TestEnum(MyEnum::Var1(MyStruct {member: 5})))``, you can implement the trait ``Into<MyStruct, Event>`` to avoid having to write it out in full.

=====

When the event is emitted, the serialization to keys and data happens as follows:

- * Since the ``TestEnum`` variant has ``kind`` nested, add the first key: ``sn_keccak(TestEnum)``, and the rest of the serialization to keys and data is done recursively via the ``starknet::event`` trait implementation of ``MyEnum``.
- * Next, you can handle a ``"kind": "nested"`` variant (previously it was ``TestEnum``, now it's ``Var1``), which means you can add another key depending on the sub-variant: ``sn_keccak(Var1)``, and proceed to serialize according to the ``starknet::event`` implementation of ``MyStruct``.
- * Finally, proceed to serialize ``MyStruct``, which gives us a single data member.

This results in ``keys = [sn_keccak(TestEnum), sn_keccak(Var1)]`` and ``data=[5]``

[NOTE]

=====

Allowing variants that are themselves enums (``TestEnum`` is an enum variant here) means further nesting is possible.

=====

For example, if the high level code is changed to:

```

[source,cairo]
----
#[event]
#[derive(Drop, starknet::Event)]
enum Event {
  ComponentEvent: test_component::Event,
```

```

    TestCounterIncreased: CounterIncreased,
    TestCounterDecreased: CounterDecreased,
    TestEnum: MyEnum
}

```

```

#[derive(Copy, Drop, starknet::Event)]
enum MyEnum {
    Var1: AnotherEnum
}

```

```

#[derive(Copy, Drop, Serde, starknet::Event)]
enum AnotherEnum {
    Var2: MyStruct
}

```

```

#[derive(Copy, Drop, Serde, starknet::Event)]
struct MyStruct {
    member: u128,
}
----

```

then ``self.emit(Event::TestEnum(MyEnum::Var1(AnotherEnum::Var2(MyStruct { member: 5 }))))``
(as before, ``Into`` implementations can shorten this) will emit an event with ``keys = [sn_keccak(TestEnum), sn_keccak(Var1), sn_keccak(Var2)]`` and ``data=[5]``.

This will look as follows in the ABI (only the relevant parts are shown):

```

[source,json]
----
{
  "type": "event",
  "name": "<namespace>::Event",
  "kind": "enum",
  "variants": [
    // ignoring all the other variants for brevity
    {
      "name": "TestEnum",
      "type": "<namespace>::MyEnum",
      "kind": "nested"
    }
  ]
},
{
  "type": "event",
  "name": "<namespace>::MyEnum",

```

```

"kind": "enum",
"variants": [
  {
    "name": "Var1",
    "type": "<namespace>::AnotherEnum",
    "kind": "nested"
  }
]
},
{
  "type": "event",
  "name": "<namespace>::AnotherEnum",
  "kind": "enum",
  "variants": [
    {
      "name": "Var2",
      "type": "<namespace>::MyStruct",
      "kind": "nested"
    }
  ]
}
}
----

```

As `TestEnum`, `Var1` and `Var2` are of kind `nested`, a selector should be added to the list of keys, before continuing to recursively serialize.

=== Flattened enum variants

You might not want to nest enums when serializing the event. For example, if you write an ERC-20 as a component, not a contract, that is pluggable anywhere, you might not want the contract to modify the keys of known events such as `Transfer`.

To avoid nesting, write the following high level code:

```

[source,cairo]
----
#[event]
#[derive(Drop, starknet::Event)]
enum Event {
  ComponentEvent: test_component::Event,
  TestCounterIncreased: CounterIncreased,
  TestCounterDecreased: CounterDecreased,
  #[flat]
  TestEnum: MyEnum
}
----

```

By writing the above, the `TestEnum` variant entry in the ABI will change to:

```
[source,json]
----
{
  "name": "TestEnum",
  "type": "<namespace>::MyEnum",
  "kind": "flat"
}
----
```

This means that `self.emit(Event::TestEnum(MyEnum::Var1(MyStruct {member: 5})))` will emit an event with `keys=[sn_keccak(Var1)]` and `data=[5]`.

== Cairo v2.0.0 changes

With the transition to `v2.0.0`, the contract ABI underwent some changes.

Consider the following high level code that generates the ABI in the following example:

```
[source, cairo]
----
#[starknet::interface]
trait IOtherContract<TContractState> {
  fn decrease_allowed(self: @TContractState) -> bool;
}

#[starknet::interface]
trait ICounterContract<TContractState> {
  fn increase_counter(ref self: TContractState, amount: u128);
  fn decrease_counter(ref self: TContractState, amount: u128);
  fn get_counter(self: @TContractState) -> u128;
}

#[starknet::contract]
mod counter_contract {
  use starknet::ContractAddress;
  use super::{
    IOtherContractDispatcher, IOtherContractDispatcherTrait,
    IOtherContractLibraryDispatcher
  };

  #[storage]
  struct Storage {
```



```

    counter: u128,
    other_contract: IOtherContractDispatcher
}

#[event]
#[derive(Drop, starknet::Event)]
enum Event {
    CounterIncreased: CounterIncreased,
    CounterDecreased: CounterDecreased
}

#[derive(Drop, starknet::Event)]
struct CounterIncreased {
    amount: u128
}

#[derive(Drop, starknet::Event)]
struct CounterDecreased {
    amount: u128
}

#[constructor]
fn constructor(
    ref self: ContractState, initial_counter: u128, other_contract_addr: ContractAddress
) {
    self.counter.write(initial_counter);
    self
        .other_contract
        .write(IOtherContractDispatcher { contract_address: other_contract_addr });
}

#[external(v0)]
impl CounterContract of super::ICounterContract<ContractState> {
    fn get_counter(self: @ContractState) -> u128 {
        self.counter.read()
    }

    fn increase_counter(ref self: ContractState, amount: u128) {
        let current = self.counter.read();
        self.counter.write(current + amount);
        self.emit(CounterIncreased { amount });
    }

    fn decrease_counter(ref self: ContractState, amount: u128) {
        let allowed = self.other_contract.read().decrease_allowed();
        if allowed {

```

```

        let current = self.counter.read();
        self.counter.write(current - amount);
        self.emit(CounterDecreased { amount });
    }
}
}
}
}
----

```

=== Interface and Impl ABI entries

Since the `CounterContract`impl`` is annotated with the ``#[external(v0)]`` attribute, you'll find the following ``impl`` entry in the ABI:

```

[source,json]
{
  "type": "impl",
  "name": "CounterContract",
  "interface_name":
    "new_syntax_test_contract::new_syntax_test_contract::ICounterContract"
}
----

```

This means that every function appearing in the ``ICounterContract`` interface is a possible entry point of the contract.

[NOTE]

=====

Standalone functions in the contract outside an external ``impl`` can also be annotated with ``#[external(v0)]`` (currently, this is the only way to add L1 handlers). In such cases, a corresponding ``function`` (or ``l1_handler``) entry will be found in the ABI in the same hierarchy as ``impls`` and interfaces.

=====

=== Events

In Cairo v2, a dedicated type for the contract's events was introduced. Currently, the contract event type must be an enum named ``Event``, whose variants are structs of the same name as the variant. Types that can be emitted via ``self.emit(_)`` must implement the ``Event`` [link:https://github.com/starkware-libs/cairo/blob/7144f2f383961cbca4804a7d056d48973721446c/corelib/src/starknet/event.cairo#L4\[trait\]](https://github.com/starkware-libs/cairo/blob/7144f2f383961cbca4804a7d056d48973721446c/corelib/src/starknet/event.cairo#L4[trait]), which defines how this type should be serialized into two ``felt252`` arrays, ``keys`` and ``data``.

The ``Event`` enum variants appear in the ABI under ``"type" = "event"`` rather than regular structs.

For such entries, each member has an additional ``kind`` field that specifies how the serialization into keys and data takes place:

- * If the kind is ``key``, then this member or variant are serialized into the event's keys.
- * If the kind is ``data``, then this member or variant are serialized into the event's data.
- * If the kind is ``nested``, then the member or variant are serialized according to the ``Event`` attribute, potentially adding to both keys and data.

[NOTE]

====

This feature is not yet supported, so no high level code written in Cairo ``v2.0.0`` can generate such an ABI.

====

=== Specification

You can find a link:https://github.com/starkware-libs/starknet-specs/blob/master/api/starknet_metadata.json#L20[JSON schema specification] of the ABI in the ``starknet-specs`` repository.

For a UI-friendly version, you can use the link:https://playground.open-rpc.org/?schemaUrl=https://raw.githubusercontent.com/starkware-libs/starknet-specs/master/api/starknet_metadata.json[OPEN-RPC playground].

[id="contract_address"]

= Contract address

The contract address is a unique identifier of the contract on Starknet. It is a chain hash of the following information:

[horizontal,labelwidth="26",role=stripes-odd]

``prefix``:: The ASCII encoding of the constant string

``STARKNET_CONTRACT_ADDRESS``.

``deployer_address``:: One of the following:

* When the contract is deployed via a ``DEPLOY_ACCOUNT`` transaction: ``0``

* When the contract is deployed via a ``deploy`` system call from another contract, the value of the ``deploy_from_zero`` parameter determines this value.

+

For information on the ``deploy_from_zero`` parameter, see the xref:smart-contracts/system-calls-cairo1.adoc#deploy[``deploy`` system call]

``salt``:: The salt passed by the contract calling the syscall, provided by the transaction sender.

``class_hash``:: See xref:./class-hash.adoc#computing_the_cairo_1_class_hash[the class hash documentation].

``constructor_calldata_hash``:: xref:cryptography/hash-

functions.adoc#pedersen_array_hash[Array hash] of the inputs to the constructor.

The address is computed as follows:

```
[source,]
----
contract_address = pedersen(
    "STARKNET_CONTRACT_ADDRESS",
    deployer_address,
    salt,
    class_hash,
    constructor_calldata_hash)
----
```

[NOTE]

====

A random `salt` ensures unique addresses for smart contract deployments, preventing conflicts when deploying identical contract classes.

It also thwarts replay attacks by influencing the transaction hash with a unique sender address.

====

.Additional resources

* For more information on the address computation, see https://github.com/starkware-libs/cairo/blob/2c96b181a6debe9a564b51dbeaaf48fa75808d53/corelib/src/starknet/contract_address.cairo in the Cairo code repository.

* xref:smart-contracts/system-calls-cairo1.adoc#deploy[`deploy` system call]
[id="contract_classes"]

= Contract classes and instances

As in object-oriented programming, Starknet distinguishes between a contract and its implementation by separating contracts into classes and instances.

== Contract classes

A `_contract class_` is the definition of a contract. It includes Cairo byte code, hint information, entry point names, and everything that defines its semantics.

Each class is uniquely identified by its `_class hash_`, comparable to a class name in traditional object-oriented programming languages.

== Contract instances

A `_contract instance_` is a deployed contract that corresponds to a class. Only contract

instances act as true contracts, in that they have their own storage and can be called by transactions or other contracts.

A contract class does not necessarily have a deployed instance in Starknet.

[IMPORTANT]

====

A contract class does not necessarily require a deployed instance in Starknet.

====

A contract instance has a nonce, the value of which is the number of transactions originating from this address plus 1. For example, when you deploy an account with a ``DEPLOY_ACCOUNT`` transaction, the nonce of the account contract in the transaction is ``0``. After the ``DEPLOY_ACCOUNT`` transaction, until the account contract sends its next transaction, the nonce is ``1``.

== Working with classes

[horizontal,labelwidth=20,role="stripes-odd"]

Adding new classes:: To introduce new classes to Starknet's state, use the ``DECLARE`` transaction.

Deploying instances:: To deploy a new instance of a previously declared class, use the ``deploy`` system call.

Using class functionality:: To use the functionality of a declared class without deploying an instance, use the ``library_call`` system call. Analogous to Ethereum's ``delegatecall``, it enables you to use code in an existing class without deploying a contract instance.

== Additional resources

- * [xref:architecture-and-concepts:smart-contracts/class-hash.adoc](#)[Class hash]
- * [xref:architecture-and-concepts:network-architecture/transactions.adoc#declare-transaction\[`DECLARE` transaction\]](#)
- * [xref:architecture-and-concepts:smart-contracts/system-calls-cairo1.adoc#deploy\[`deploy` system call\]](#)
- * [xref:architecture-and-concepts:smart-contracts/system-calls-cairo1.adoc#library_call\[`library_call` system call\]](#)

[id="contract_storage"]

= Contract storage

[id="storage_layout"]

== Storage layout

The contract's storage is a persistent storage space where you can read, write, modify,

and persist data. The storage is a map with stem:[\$2^{251}\$] slots, where each slot is a felt which is initialized to 0.

```
[id="storage_low_level_functions"]  
== Storage low level functions
```

The basic function for writing to storage writes, value to key is:

```
[source,js]  
----  
storage_write_syscall(address_domain, address, value)  
----
```

// todo add description explaining what address_domain is

`storage_read` is a basic function that is used for getting the storage address, this function is created by the compiler when defining a storage variable, as explained below. This function returns the address of the storage variable. Below we discuss how this address is determined from the variable's name and keys.

Both `storage_read` and `storage_write` are system calls that can be imported by adding the line:

```
[source,javascript]  
----  
use starknet::syscalls::storage_read_syscall;  
use starknet::syscalls::storage_write_syscall;  
----
```

```
[id="storage_variables"]  
== Storage variables
```

The most common way to interact with a contract's storage is through storage variables.

The `#[storage]` attribute above the `Storage` struct declares that the contents of this struct are part of the contract storage. The storage variables stored inside this struct can consist of a single felt, or it can be a mapping from multiple arguments to a tuple of felts or structs.

To use this variable, the `var.read(args)`, `var.write(args, value)` and `var.address(args)` functions are automatically created by the `#[storage]` attribute, for reading the storage value, writing the storage value and getting the storage address, respectively.

The Starknet contract compiler generates the Cairo code that maps the storage variable's name and argument values to an address -- so that it can be part of the

generated proof.

The address of a storage variable is computed as follows:

- * If it is a single value, then the address is ``sn_keccak(variable_name)``, where `variable_name` is the ASCII encoding of the variable's name.
- * If it is a (nested) mapping, then the address of the value at key ``+k_1,...,k_n+`` is ``+h(...h(h(sn_keccak(variable_name),k_1),k_2),...,k_n)+`` where `stem:[h$]` is the Pedersen hash and the final value is taken `stem:[bmod 2251]-256$]`
- * If it is a mapping to complex values (e.g., tuples or structs), then this complex value lies in a continuous segment starting from the address calculated in the previous point. Note that 256 field elements is the current limitation on the maximal size of a complex storage value.
- * Note that when calling ``var.address(args)`` for a storage variable with complex values, the returned value is the address of the first element in the storage.

We can summarize the above as follows:

``storage variable address := pedersen(keccak(variable name), keys)``

The following example defines storage variables with complex values.

```
[source,cairo]
----
#[storage]
struct Storage {
    name: felt252,
    symbol: felt252,
    decimals: u8,
    total_supply: u256,
    balances: LegacyMap::<ContractAddress, u256>,
    allowances: LegacyMap::<(ContractAddress, ContractAddress), u256>,
}
----
```

= Migrating a contract from Cairo v1 to Cairo v2

With the link:<https://github.com/starkware-libs/cairo/releases/tag/v2.0.0-rc0>[v2.0.0 release] of the Cairo compiler, the Starknet contract syntax has evolved, affecting the organization of functions, storage, and events.

For more information on the latest syntax changes, see the Community Forum post link:<https://community.starknet.io/t/cairo-1-contract-syntax-is-evolving/94794>[_Cairo 1: Contract Syntax is Evolving_].

.Prerequisites

- * A contract written with the Cairo compiler v1
- * The most recent version of the Cairo compiler

.Procedure

. Change the contract annotation from ``#[contract]`` to ``#[starknet::contract]``. For example::

```
+
[tab]
====
Cairo v1::
+
[source,cairo]
----
#[contract]
mod CounterContract {
    ...
}
----
```

```
Cairo v2::
+
[source,cairo]
----
#[starknet::contract]
mod CounterContract {
    ...
}
----
```

====
. Annotate the ``Storage`` struct with the ``#[storage]`` attribute. For example:

```
+
[tab]
====
Cairo v1::
+
[source,cairo]
----
struct Storage {
    counter: u128,
    other_contract: IOtherContractDispatcher
}
----
```

```
Cairo v2::
+
```



```

[source,cairo]
----
#[storage]
struct Storage {
    counter: u128,
    other_contract: IOtherContractDispatcher
}
----
=====

```

. Gather your contract's `external` and `view` function signatures under a trait annotated with

`#[starknet::interface]` as follows:

- +
 - * Add a generic parameter to the trait. In the following example, the name `TContractState` represents the state of your contract.
 - * For view functions, add the `self: @TContractState` argument.
 - * For external functions, add the `ref self: TContractState` argument.
 - * Static functions that do not touch storage or emit events do not require an additional argument.

+
For example:

```

+
[tabs]
=====
Cairo v1::
+
[source,cairo]
----
#[contract]
mod CounterContract {
    #[external]
    fn increase_counter(amount: u128) { ... }
    #[external]
    fn decrease_counter(amount: u128) { ... }
    #[view]
    fn get_counter() -> u128 { ... }
}
----
Cairo v2::
+
[source,cairo]
----
#[starknet::interface]
trait ICounterContract<TContractState> {

```

```

    fn increase_counter(ref self: TContractState, amount: u128);
    fn decrease_counter(ref self: TContractState, amount: u128);
    fn get_counter(self: @TContractState) -> u128;
}
#[starknet::contract]
mod CounterContract {
    ...
}
----
=====

```

. Add the external and view function bodies under an ``impl`` of the interface trait, and mark the ``impl`` with the ``external(v0)`` attribute, which generates the type of dispatcher that is used to call the contract.

+

For example:

+

[tabs]

=====

Cairo v1::

+

[source,cairo]

```

#[contract]
mod CounterContract {
    #[external]
    fn increase_counter(amount: u128) { ... }
    #[external]
    fn decrease_counter(amount: u128) { ... }
    #[view]
    fn get_counter() -> u128 { ... }
}
----

```

Cairo v2::

+

[source,cairo]

```

#[starknet::interface]
trait ICounterContract<TContractState> {
    fn increase_counter(ref self: TContractState, amount: u128);
    fn decrease_counter(ref self: TContractState, amount: u128);
    fn get_counter(self: @TContractState) -> u128;
}
#[starknet::contract]
mod CounterContract {

```

```

#[external(v0)]
impl CounterContract of super::ICounterContract<ContractState> {
    fn increase_counter(ref self: ContractState, amount: u128) { ... }
    fn decrease_counter(ref self: ContractState, amount: u128) { ... }
    fn get_counter(self: @ContractState) -> u128 { ... }
}
}
----
=====

```

. Replace the `#[abi]` attribute with `#[starknet::interface]`.

+
[TIP]
=====

While it doesn't affect the generated code, adding to the trait a generic parameter `T` representing the contract's state, and adding the `ref self: T` argument to external functions and `self: @T` argument for view functions makes the implementation more complete.

=====

+
For example:

+
[tabs]
=====

Cairo v1::

+
[source,cairo]

```

#[abi]
trait IOtherContract {
    fn decrease_allowed() -> bool;
}

```

Cairo v2::

+
[source,cairo]

```

#[starknet::interface]
trait IOtherContract<TContractState> {
    fn decrease_allowed(self: @TContractState) -> bool;
}

```

=====

. Modify storage accesses to happen through `ContractState` or `@ContractState`.

```
+
[NOTE]
====
No external functions in the contract that access storage also need to get it as an
argument.
```

```
// Get what as an argument? Storage?
```

```
====
```

```
+
For example:
```

```
+
[tabs]
```

```
====
```

```
Cairo v1::
```

```
+
```

```
[source,cairo]
```

```
----
```

```
let current = counter::read();
```

```
----
```

```
Cairo v2::
```

```
+
```

```
[source,cairo]
```

```
----
```

```
let current = self.counter.read();
```

```
----
```

```
====
```

. Unify all the contract's events under the `Event` enum, and add a corresponding struct for every variant.

```
+
```

```
[NOTE]
```

```
====
```

All the structs must derive the `Event` trait,
and each member type must implement the `Serde` trait.

```
====
```

```
+
```

```
For example:
```

```
+
```

```
[tabs]
```

```
====
```

```
Cairo v1::
```

```
+
```

```
[source,cairo]
```

```
----
```

```
#[event]
```

```
fn counter_increased(amount: u128) {}
```

```

#[event]
fn counter_decreased(amount: u128) {}
----
Cairo v2::
+
[source,cairo]
----
#[event]
#[derive(Drop, starknet::Event)]
enum Event {
    CounterIncreased: CounterIncreased,
    CounterDecreased: CounterDecreased
}
#[derive(Drop, starknet::Event)]
struct CounterIncreased {
    amount: u128
}
#[derive(Drop, starknet::Event)]
struct CounterDecreased {
    amount: u128
}
----
=====

```

. Emit events via the `ContractState` type. For example:

```

+
[tags]
=====
Cairo v1::
+
[source,cairo]
----
fn increase_counter(amount: u128) {
    ...
    counter_increased(amount);
}
----
Cairo v2::
+
[source,cairo]
----
fn increase_counter(ref self: ContractState, amount: u128) {
    ...
    self.emit(Event::CounterIncreased(CounterIncreased { amount }));
}
----

```

====

[id="block_execution_info"]

= Execution information

For the most up-to-date information, see the link:<https://github.com/starkware-libs/cairo/blob/main/corelib/src/starknet/info.cairo>[`info.cairo`] contract.

The struct `ExecutionInfo` contains the following information about the currently executing block and the transactions in the block.

== The `ExecutionInfo` struct

[horizontal,labelwidth="25",role="stripes-odd"]

`block_info`: Box<BlockInfo>:: Contains information about a block. For details, see xref:#block_info[]

`tx_info`: Box<TxInfo>:: Contains information about a transaction. For details, see xref:#tx_info[]

`caller_address`: ContractAddress:: The address of the contract that invokes the `get_execution_info` syscall.

`contract_address`: ContractAddress:: The address of the contract in which the `get_execution_info` syscall appears.

`entry_point_selector`: felt252:: The function that includes the `get_execution_info` syscall.

[#block_info]

== The `BlockInfo` struct

[horizontal,labelwidth="25",role="stripes-odd"]

`_block_number_`: u64:: The number of the block that is currently being executed.

When called from an account contract's +`__validate__`, +`__validate_deploy__`, or +`__validate_declare__`+ function, this value is rounded down to the nearest multiple of 100.

`_block_timestamp_`: u64:: The timestamp showing the creation time of the block, in seconds since the Unix epoch, based on UTC time, rounded down to the nearest second. When called from an account contract's +`__validate__`, +`__validate_deploy__`, or +`__validate_declare__`+ function, this value is rounded down to the nearest hour.

`_sequencer_address_`: ContractAddress:: The address of the Starknet sequencer contract.

[#tx_info]

== Transaction information: The `TxInfo` struct

[horizontal,labelwidth="25",role="stripes-odd"]

`_version_`: felt252:: The version of the transaction. It is fixed (currently, 3) in the OS,

and should be signed by the account contract. This field allows invalidating old transactions, whenever the meaning of the other transaction fields is changed (in the OS).

`_account_contract_address_`: `ContractAddress``:: The account contract from which this transaction originates.

`_max_fee_`: `u128``:: The `max_fee` field of the transaction.

`_signature_`: `Span<felt252>``:: The signature of the transaction.

`_transaction_hash_`: `felt252``:: The hash of the transaction.

`_chain_id_`: `felt252``:: The identifier of the chain.

This field can be used to prevent replay of testnet transactions on mainnet.

`_nonce_`: `felt252``:: The transaction's nonce.

`_resource_bounds_`: `Span<ResourceBounds>``:: A span of ``ResourceBounds`` structs.

For details, see [xref:#resource_bounds\[\]](#).

`_tip_`: `u128``:: The tip.

`_paymaster_data_`: `Span<felt252>``:: If specified, the paymaster should pay for the execution of the tx.

The data includes the address of the paymaster sponsoring the transaction, followed by extra data to send to the paymaster.

`_nonce_data_availability_mode_`: `u32``:: The data availability mode for the nonce.

`_fee_data_availability_mode_`: `u32``:: The data availability mode for the account balance from which fee will be taken.

`_account_deployment_data_`: `Span<felt252>``:: If nonempty, will contain the required data for deploying and initializing an account contract: its class hash, address salt and constructor calldata.

[[#resource_bounds](#)]

== The ``ResourceBounds`` struct

[[horizontal](#),[labelwidth="25"](#),[role="stripes-odd"](#)]

`_resource_`: `felt252``:: The name of the resource.

`_max_amount_`: `u64``:: The maximum amount of the resource allowed for usage during the execution.

`_max_price_per_unit_`: `u128``:: The maximum price the user is willing to pay for the resource unit.

[[id="serialization_of_types_in_Cairo"](#)]

= Serialization of Cairo types

When you interact with contracts, especially if you are a library or SDK developer that wants to construct transactions, you need to understand how Cairo handles types that are larger than 252 bits so you can correctly formulate the calldata in a transaction.

The field element (``felt252``), which contains 252 bits, is the only actual type in the Cairo VM. So all high-level Cairo types that are larger than 252 bits, such as ``u256`` or arrays, are ultimately represented by a list of felts. In order to interact with a contract, you need to know how to encode its arguments as a list of felts so you can correctly formulate the

calldata in the transaction.

SDKs, such as starknet.js, encode the calldata for you, so you can simply specify any type and the SDK properly formulates the calldata. For example, you don't need to know that a ``u256`` value is represented by two ``felt252`` values. You can simply specify a single integer in your code, and the SDK takes care of the serialization and encoding.

[#data_types_of_252_bits_or_less]
== Data types of 252 bits or less

The following types are smaller than 252 bits. For these types, each value is serialized as a single-member list that contains one ``felt252`` value.

- * ``ContractAddress``
- * ``EthAddress``
- * ``StorageAddress``
- * ``ClassHash``
- * Unsigned integers smaller than 252 bits: ``u8``, ``u16``, ``u32``, ``u64``, ``u128``, and ``usize``
- * ``bytes31``
- * ``felt252``
- * Signed integers smaller than 252 bits: ``i8``, ``i16``, ``i32``, ``i64``, and ``i128``.

+

A negative value, `stem:[-x]`, is serialized as `stem:[P-x]`, where:

+

[stem]

++++

$P = 2^{251} + 17 \cdot 2^{192} + 1$

++++

+

For example, ``-5`` is serialized as `stem:[P-5]`. For more information on the value of `stem:[P]`, see [xref:architecture-and-concepts:cryptography/p-value.adoc](#)[The STARK field].

[#data_types_greater_than_252_bits]
== Data types greater than 252 bits

The following Cairo data types have non-trivial serialization:

- * ``u256`` and ``u512``
- * arrays
- * enums
- * structs
- * ``ByteArray``, which represents strings

[#serialization_of_unsigned_integers]

== Serialization of unsigned integers

Among unsigned integers, only `u256` and `u512` have non-trivial serialization.

[#serialization_in_u256_values]

=== Serialization of `u256` values

A `u256` value in Cairo is represented by two `felt252` values, as follows:

- * The first `felt252` value contains the 128 least significant bits, usually referred to as the low part of the original `u256` value.

- * The second `felt252` value contains the 128 most significant bits, usually referred to as the high part of the original `u256` value.

For example:

- * A `u256` variable whose decimal value is `2` is serialized as `[2,0]`. To understand why, examine the binary representation of `2` and split it into two 128-bit parts, as follows:

+

[stem]

++++

$\underbrace{0\cdots0}_{\text{128 high bits}} \mid$

$\underbrace{0\cdots10}_{\text{128 low bits}}$

++++

//

// [#binary_representation_of_u256]

// .Binary representation of `2` in a serialized `u256`

// [%autowidth,cols="2"]

// |===

// |`felt252`~1~ = `0`~binary~ = `0`~decimal~|`felt252`~2~ = `10`~binary~ =
`2`~decimal~`

//

// a|` 0b000...000`

// [stem]

// +++++

// $\underbrace{0\cdots0}_{\text{128 bits}}$

// +++++

// a|` 0b000...000`

// [stem]

// +++++

// $\underbrace{0\cdots0}_{\text{128 bits}}$

// $\underbrace{0\cdots10}_{\text{128 bits}}$

// +++++

// |===

* A `u256` variable whose decimal value is 2^{128} is serialized as `[0,1]`. To understand why, examine the binary representation of 2^{128} and split it into two 128-bit parts, as follows:

```
+
[stem]
++++
\underbrace{0\cdots 01}_{\text{128 high bits}} |
\underbrace{0\cdots 0}_{\text{128 low bits}}
++++
```

* A `u256` variable whose decimal value is $2^{129} + 2^{128} + 20$, is serialized as `[20,3]`. To understand why, examine the binary representation of the $2^{129} + 2^{128} + 20$ and split it into two 128-bit parts, as follows:

```
+
[stem]
++++
\underbrace{0\cdots 011}_{\text{128 high bits}} |
\underbrace{0\cdots 10100}_{\text{128 low bits}}
++++
```

[#serialization_in_u512_values]
 === Serialization of `u512` values

The `u512` type in Cairo is a struct containing four `felt252` members, each representing a 128-bit limb of the original integer, similar to the `u256` type.

[#serialization_of_arrays]
 == Serialization of arrays

Arrays are serialized as follows:

`<__array_length__>, <__serialized_member_0__>, ..., <__serialized_member_n__>`

For example, consider the following array of `u256` values:

```
[source,cairo]
----
let POW_2_128: u256 = 0x100000000000000000000000000000000
let array: Array<u256> = array![10, 20, POW_2_128]
----
```

Each `u256` value in the array is represented by two `felt252` values. So the array above is serialized as follows:

[stem]

```
+++++
\underbrace{3}_{\textit{number_of_array_members}} ,
\underbrace{10,0}_{\textit{serialized_member_0}} ,
\underbrace{20,0}_{\textit{serialized_member_1}} ,
\underbrace{0,1}_{\textit{serialized_member_2}}
+++++
```

Combining the above, the array is serialized as follows: `[3,10,0,20,0,0,1]`

```
[#serialization_of_enums]
== Serialization of enums
```

An enum is serialized as follows:

```
`<__index_of_enum_variant__>,<__serialized_variant__>`
```

Note that enum variants indices are 0-based, not to confuse with their storage layout, which is 1-based, to distinguish the first variant from an uninitialized storage slot.

.Enum serialization example 1

Consider the following definition of an enum named `Week`:

```
[source,cairo]
----
enum Week {
    Sunday: (), // Index=0. The variant type is the unit type (0-tuple).
    Monday: u256, // Index=1. The variant type is u256.
}
----
```

Now consider instantiations of the `Week` enum's variants as shown in the table below:

```
[#serialization_of_Week]
.Serialization of `Week` variants
```

```
[cols=",,",]
|===
|Instance |Description |Serialization
```

```
|`Week::Sunday` | Index=`0`. The variant's type is the unit type. | `[0]`
|`Week::Monday(5)` a| Index=`1`. The variant's type is `u256`, hence serialized to
|[5,0], as shown in xref:#serialization_in_u256_values[] .| `[1,5,0]`
|===
```

.Enum serialization example 2

Consider the following definition of an enum named `MessageType`:

```
[source,cairo]
----
enum MessageType {
    A,
    #[default]
    B: u128,
    C
}
----
```

Now consider instantiations of the `MessageType` enum's variants as shown in the table below:

```
[#serialization_of_MessageType]
.Serialization of `MessageType` variants
[cols=",",","]
|===
|Instance |Description |Serialization

|`MessageType::A` | Index=`1`. The variant's type is the unit type. | `[0]`
|`MessageType::B(6)` a| Index=`0`. The variant's type is `u128`. | `[1,6]`
|`MessageType::C` | Index=`2`. The variant's type is the unit type. | `[2]`
|===
```

As you can see about, the `#[default]` attribute does not affect serialization. It only affects the storage layout of `MessageType`, where the default variant `B` will be stored as `0`.

```
[#serialization_of_structs]
== Serialization of structs
```

You serialize a struct by serializing its members one at a time.

Its members are serialized in the order in which they appear in the definition of the struct.

For example, consider the following definition of the struct `MyStruct`:

```
[source,cairo]
----
struct MyStruct {
    a: u256,
```

```

    b: felt252,
    c: Array<felt252>
}

```

The serialization is the same for both of the following instantiations of the struct's members:

```

[cols="2"]
|===
a|[source,cairo]
----
let my_struct = MyStruct {
    a: 2, b: 5, c: [1,2,3]
};
----

```

```

a|[source,cairo]
----
let my_struct = MyStruct {
    b: 5, c: [1,2,3], a: 2
};
----
|===

```

The serialization of ``MyStruct`` is determined as shown in the table [xref:#serialization_for_a_struct_in_cairo\[\]](#).

```

[#serialization_for_a_struct_in_cairo]
.Serialization for a struct in Cairo
[cols="3"]
|===
|Member|Description|Serialization
|`a: 2`
|For information on serializing `u256` values, see xref:#serialization\_in\_u256\_values\[\]
|`[2,0]`
|`b: 5`
|One `felt252` value
|`5`
|`c: [1,2,3]`
|An array of three `felt252` values
|`[3,1,2,3]`
|===

```

Combining the above, the struct is serialized as follows: `[0,2,5,3,1,2,3]`

```
[#serialization_of_byte_arrays]
== Serialization of byte arrays
```

A string is represented in Cairo as a `ByteArray` type. A byte array is actually a struct with the following members:

```
. *`data: Array<felt252>* +
```

Contains 31-byte chunks of the byte array. Each `felt252` value has exactly 31 bytes. If the number of bytes in the byte array is less than 31, then this array is empty.

```
. *`pending_word: felt252* +
```

The bytes that remain after filling the `data` array with full 31-byte chunks. The pending word consists of at most 30 bytes.

```
. *`pending_word_len: usize* +
```

The number of bytes in `pending_word`.

.Example 1: A string shorter than 31 characters

Consider the string `hello`, whose ASCII encoding is the 5-byte hex value `0x68656c6c6f`. The resulting byte array is serialized as follows:

```
[source,cairo]
```

```
----
```

```
0, // Number of 31-byte words in the data array.
0x68656c6c6f, // Pending word
5 // Length of the pending word, in bytes
```

```
----
```

.Example 2: A string longer than 31 bytes

Consider the string `Long string, more than 31 characters.`, which is represented by the following hex values:

```
* `0x4c6f6e6720737472696e672c206d6f7265207468616e203331206368617261` (31-  
byte word)
```

```
* `0x63746572732e` (6-byte pending word)
```

The resulting byte array is serialized as follows:

```
[source,cairo]
```

```

----
    1, // Number of 31-byte words in the array construct.
    0x4c6f6e6720737472696e672c206d6f7265207468616e203331206368617261, // 31-
byte word.
    0x63746572732e, // Pending word
    6 // Length of the pending word, in bytes
----

```

== Additional resources

* [link:https://book.cairo-lang.org/ch02-02-data-types.html#integer-types](https://book.cairo-lang.org/ch02-02-data-types.html#integer-types)[Integer types] in `_The Cairo Programming Language_`.

```

[id="events"]
= Events
:stem: latex

```

A contract may emit events throughout its execution. Each event contains the following fields:

- * ``from_address``: address of the contract emitting the events
- * ``keys``: a list of field elements
- * ``data``: a list of field elements

The keys can be used for indexing the events, while the data may contain any information that we wish to log (note that we are dealing with two separate lists of possibly varying size, rather than a list of key-value pairs).

```

[id="emitting_events"]
== Emitting events

```

Events can be defined in a contract using the ``@event`` decorator. Once an event ``E`` has been defined, the compiler automatically adds the function ``E.emit()``. The following example illustrates how an event is defined and emitted:

```

[source,cairo]
----
#[event]
fn Transfer(from: ContractAddress, to: ContractAddress, value: u256) {}
----

```

```

[source,cairo]
----
Transfer(12345, 12345, 1)
----

```

The emit function emits an event with a single key, which is an identifier of the event, given by `stem:[$\text{sn_keccak}(event_name)$]`, where `stem:[$event_name$]` is the ASCII encoding of the event's name and `stem:[sn_keccak]` is defined xref:../cryptography/hash-functions.adoc#starknet_keccak[here].

To emit custom keys, one should use the low level ``emit_event`` system call:

```
[source,cairo]
----
use starknet::syscalls::emit_event_syscall;

let keys = array!['key', 'deposit'];
let values = array![1, 2, 3];
emit_event_syscall(keys, values).unwrap_syscall();
----
```

The above code emits an event with two keys, the https://www.cairo-lang.org/docs/how_cairo_works/consts.html#short-string-literals ``key`` and ``deposit`` (think of those as identifiers of the event that can be used for indexing) and three data elements 1, 2, 3.

[TIP]

=====

When using the higher level ``emit`` syntax, the event's data may be of complex types, for example:

```
[source,cairo]
----
struct Point:
    member x : felt
    member y : felt
end

@event
func message_received(arr_len : felt, arr: felt*, p: Point):
end

# ...

let (data : felt*) = alloc()
assert data[0] = 1
assert data[1] = 2
let p = Point(3,4)
message_received.emit(2, data, p)
```

=====

The emitted events are part of the transaction receipt. For more information, see [xref:architecture-and-concepts:network-architecture/transaction-life-cycle.adoc#transaction-receipt](#)[Transaction receipt].

[id="event_abi"]
== Event definition in the ABI

The event definition appears in the contract's ABI. It contains the list of data fields, with the name and type for each, and the list of the custom keys, that is, all keys except the event identifier discussed above. Below is an example of an event inside the ABI:

[#example_of_an_event_in_the_abi]
.Example of an event in the ABI
[source,json]

{
 "type": "event",
 "name": "Transfer",
 "inputs": [
 {
 "name": "from",
 "type": "core::starknet::contract_address::ContractAddress"
 },
 {
 "name": "to",
 "type": "core::starknet::contract_address::ContractAddress"
 },
 {
 "name": "value",
 "type": "core::integer::u256"
 }
]
}

[id="event_hash"]
== Event hash

The event hash is given by:

```
[stem]
++++
h(h(h(h(0,from\_address),keys\_hash),data\_hash),3)
++++
```

Where:

- * stem:[\$keys_hash\$] and stem:[\$data_hash\$] are the hashes of the keys list and data list respectively. For more information, see xref:../cryptography/hash-functions.adoc#array_hashing[Array hashing].
- * stem:[\$h\$] is the Pedersen hash function.

The event hashes are included in the xref:network-architecture/block-structure.adoc[`event_commitment``] field of a block.

== Additional resources

- * xref:architecture-and-concepts/cryptography/hash-functions.adoc#array_hashing[]
 - * xref:architecture-and-concepts/cryptography/hash-functions.adoc#pedersen_hash[Pedersen hash function]
 - * The ``event_commitment`` field in xref:network-architecture/block-structure.adoc[]=
- System Calls

Writing smart contracts requires various associated operations, such as calling another contract or accessing the contract's storage, that standalone programs do not require. The Starknet contract language supports these operations by using system calls. System calls enable a contract to require services from the Starknet OS. You can use system calls in a function to get information that depends on the broader state of Starknet, which would otherwise be inaccessible, rather than local variables that appear in the function's scope.

```
[id="get_block_hash"]
== `get_block_hash`
```

```
[discrete]
=== Function signature
```

```
[source,cairo,subs="+quotes,+macros"]
----
extern fn get_block_hash_syscall(
    block_number: u64
) -> SyscallResult<felt252> implicits(GasBuiltin, System) nopanic;
----
```

```
[discrete]
```

=== Description

Gets the hash of a specific Starknet block within the range of
`[__first_v0_12_0_block__, __current_block__ - 10]`.

[discrete]

=== Arguments

[horizontal,labelwidth="25",role="stripes-odd"]

`_block_number_`: u64`:: The number of the block whose hash you want to get.

[discrete]

=== Return values

The hash of the specified block.

[discrete]

=== Common library

link:<https://github.com/starkware-libs/cairo/blob/0c882679fdb24a818cad19f2c18dec6f6ef66153/corelib/src/starknet/syscalls.cairo#L37> [`syscalls.cairo`^]

[discrete]

=== Error messages

[horizontal,labelwidth="25",role="stripes-odd"]

`Block number out of range`:: `_block_number` is greater than `_current_block__ - 10`.
`0`:: `_block_number` is less than the first block number of v0.12.0.

[id="get_execution_info"]

== `get_execution_info`

[discrete]

=== Function signature

[source,cairo,subs="+quotes,+macros"]

```
extern fn get_execution_info_syscall() ->
SyscallResult<Box<starknet::info::ExecutionInfo>> implicants(
    GasBuiltin, System
) nopanic;
----
```

[discrete]

=== Description

Gets information about the currently executing block and the transactions in the block. For a complete description of this information, see [xref:smart-contracts/execution-info.adoc\[\]](#)

This single system call contains all information for a block, transaction, and execution context.

When an account's `__validate__`, `__validate_deploy__`, or `__validate_declare__` function calls `get_execution_info`, the return values for `block_timestamp` and `block_number` are modified as follows:

* `block_timestamp` returns the hour, rounded down to the nearest hour.

* `block_number` returns the block number, rounded down to the nearest multiple of 100.

[discrete]

=== Arguments

None.

[discrete]

=== Return values

[horizontal,labelwidth="25",role="stripes-odd"]

link:<https://github.com/starkware-libs/cairo/blob/main/corelib/src/starknet/info.cairo#L8>`[`ExecutionInfo`]`:: A struct that contains information about the currently executing function, transaction, and block.

[discrete]

=== Common library

link:<https://github.com/starkware-libs/cairo/blob/cc08c898f0eb3e58797674f20994df0ba641983/corelib/src/starknet/syscalls.cairo#L35>`[`syscalls.cairo`^]`

[discrete]

=== Example

This example shows how to pull the block number from the `ExecutionInfo` struct.

[source,cairo]

```
let execution_info = get_execution_info().unbox();
let block_info = execution_info.block_info.unbox();
let block number = block_info.block_number;
```

[id="call_contract"]
== `call_contract`

[discrete]
=== Function signature

[source,cairo,subs="+quotes,+macros"]

```
extern fn call_contract_syscall(
    address: ContractAddress, entry_point_selector: felt252, calldata: Span<felt252>
) -> SyscallResult<Span<felt252>> implicits(GasBuiltin, System) nopanic;
```

[discrete]
=== Description

Calls a given contract. This system call expects the address of the called contract, a selector for a function within that contract, and call arguments.

[NOTE]

=====

An internal call can't return `Err(_)` as this is not handled by the sequencer and the Starknet OS.

If `call_contract_syscall` fails, this can't be caught and will therefore result in the entire transaction being reverted.

=====

[discrete]
=== Arguments

[horizontal,labelwidth=35]

`_address_`: ContractAddress`:: The address of the contract you want to call.
`_entry_point_selector_`: felt252`:: A selector for a function within that contract.
`_calldata_`: Span<felt252>`:: The calldata array.

[discrete]
=== Return values

* The call response, of type `SyscallResult<Span<felt252>>`.

[discrete]

=== Common library

link:<https://github.com/starkware-libs/cairo/blob/cca08c898f0eb3e58797674f20994df0ba641983/corelib/src/starknet/syscalls.cairo#L10>``syscalls.cairo`^`

[NOTE]

=====

This is considered a lower-level syntax for calling contracts.

If the interface of the called contract is available, then you can use a more straightforward syntax.

=====

[id="deploy"]

== ``deploy``

[discrete]

=== Function signature

[source,cairo,subs="+quotes,+macros"]

```
extern fn deploy_syscall(
    class_hash: ClassHash,
    contract_address_salt: felt252,
    calldata: Span<felt252>,
    deploy_from_zero: bool,
) -> SyscallResult<(ContractAddress, Span::<felt252>)> implicits(GasBuiltin, System)
nopanic;
```

[discrete]

=== Description

Deploys a new instance of a previously declared class.

[discrete]

=== Arguments

[horizontal,labelwidth=35]

``_class_hash_``: ClassHash :: The class hash of the contract to be deployed.

``_contract_address_salt_``: felt252 :: The salt, an arbitrary value provided by the sender, used in the computation of the xref:[smart-contracts/contract-address.adoc](#)[contract's address].

``_calldata_``: Span<felt252> :: The constructor's calldata. An array of felts.

``_deploy_from_zero_``: bool :: A flag that determines whether the deployer's address

affects the computation of the contract address. When not set, or when set to `FALSE`, the caller address is used as the new contract's deployer address. When set to `TRUE`, 0 is used.

[discrete]
=== Return values

* A tuple wrapped with `SyscallResult` where:
** The first element is the address of the deployed contract, of type `ContractAddress`.
** The second element is the response array from the contract's constructor, of type `Span::<felt252>`.

[discrete]
=== Common library

link:<https://github.com/starkware-libs/cairo/blob/main/corelib/src/starknet/syscalls.cairo#L20>[`syscalls.cairo`^]

[id="emit_event"]
== `emit_event`

[discrete]
=== Function signature

```
[source,cairo,subs="+quotes,+macros"]
----
extern fn emit_event_syscall(
    keys: Span<felt252>, data: Span<felt252>
) -> SyscallResult<()> implicits(GasBuiltin, System) nopanic;
----
```

[discrete]
=== Description

Emits an event with a given set of keys and data.

For more information, and for a higher-level syntax for emitting events, see [xref:architecture-and-concepts:smart-contracts/starknet-events.adoc](#)[Starknet events].

[discrete]
=== Arguments

[horizontal,labelwidth=35]
`_keys_`: Span<felt252>`:: The event's keys. These are analogous to Ethereum's event topics, you can use the link:<https://github.com/starkware-libs/starknet-specs/blob/>

c270b8170684bb09741672a7a4ae5003670c3f43/api/
starknet_api_openrpc.json#L569RPC[starknet_getEvents] method to filter by these
keys.

`_data_`: Span<felt252>`:: The event's data.

[discrete]
=== Return values

None.

[discrete]
=== Common library

link:<https://github.com/starkware-libs/cairo/blob/cca08c898f0eb3e58797674f20994df0ba641983/corelib/src/starknet/syscalls.cairo#L30>[`syscalls.cairo`^]

[discrete]
=== Example

The following example emits an event with two keys, the strings `key` and `deposit` and three data elements: `1`, `2`, and `3`.

```
[source,cairo]
----
let keys = array!['key', 'deposit'];
let values = array![1, 2, 3];
emit_event_syscall(keys, values).unwrap_syscall();
----
```

[id="library_call"]
== `library_call`

[discrete]
=== Function signature

```
[source,cairo,subs="+quotes,+macros"]
----
extern fn library_call_syscall(
    class_hash: ClassHash, function_selector: felt252, calldata: Span<felt252>
) -> SyscallResult<Span<felt252>> implicits(GasBuiltin, System) nopanic;
----
```

[discrete]
=== Description

Calls the requested function in any previously declared class. The class is only used for its logic.

This system call replaces the known delegate call functionality from Ethereum, with the important difference that there is only one contract involved.

[discrete]
=== Arguments

[horizontal,labelwidth=35]
`_class_hash_`: ClassHash`:: The hash of the class you want to use.
`_function_selector_`: felt252`:: A selector for a function within that class.
`_calldata_`: Span<felt252>`:: The calldata.

[discrete]
=== Return values

* The call response, of type `SyscallResult<Span<felt252>>`.

[discrete]
=== Common library

link:<https://github.com/starkware-libs/cairo/blob/cca08c898f0eb3e58797674f20994df0ba641983/corelib/src/starknet/syscalls.cairo#L43>[`syscalls.cairo`^]

[id="send_message_to_L1"]
== `send_message_to_L1`

[discrete]
=== Function signature

```
[source,cairo,subs="+quotes,+macros"]
----
extern fn send_message_to_l1_syscall(
    to_address: felt252, payload: Span<felt252>
) -> SyscallResult<()> implicits(GasBuiltin, System) nopanic;
----
```

[discrete]
=== Description

Sends a message to L1.

This system call includes the message parameters as part of the proof's output and exposes these parameters to the Starknet Core Contract on L1 once the state update,

including the transaction, is received.

For more information, see Starknet's [xref:network-architecture/messaging-mechanism.adoc\[messaging mechanism\]](#).

[discrete]
=== Arguments

[horizontal,labelwidth=35]
`_to_address_`: felt252`:: The recipient's L1 address.
`_payload_`: Span<felt252>`:: The array containing the message payload

[discrete]
=== Return values

None.

[discrete]
=== Common library

link:<https://github.com/starkware-libs/cairo/blob/cca08c898f0eb3e58797674f20994df0ba641983/corelib/src/starknet/syscalls.cairo#L51>[`syscalls.cairo`^]

[discrete]
=== Example

The following example sends a message whose content is `(1,2)` to the L1 contract whose address is `3423542542364363`.

```
[source,cairo,subs="+quotes,+macros"]
----
let payload = ArrayTrait::new();
payload.append(1);
payload.append(2);
send_message_to_l1_syscall(payload).unwrap_syscall();
----
```

[id="replace_class"]
== `replace_class`

[discrete]
=== Function signature

```
[source,cairo,subs="+quotes,+macros"]
----
```

```
extern fn replace_class_syscall(  
    class_hash: ClassHash  
) -> SyscallResult<()> implicits(GasBuiltin, System) nopanic;  
----
```

[discrete]

=== Description

Once `replace_class` is called, the class of the calling contract (i.e. the contract whose address is returned by `get_contract_address` at the time the syscall is called) will be replaced

by the class whose hash is given by the class_hash argument.

[NOTE]

=====

After calling `replace_class`, the code currently executing from the old class will finish running.

The new class will be used from the next transaction onwards or if the contract is called via

the `call_contract` syscall in the same transaction (after the replacement).

=====

[discrete]

=== Arguments

[horizontal,labelwidth=35]

`class_hash`: ClassHash:: The hash of the class you want to use as a replacement.

[discrete]

=== Return values

None.

[discrete]

=== Common library

link:[https://github.com/starkware-libs/cairo/blob/cca08c898f0eb3e58797674f20994df0ba641983/corelib/src/starknet/](https://github.com/starkware-libs/cairo/blob/cca08c898f0eb3e58797674f20994df0ba641983/corelib/src/starknet/syscalls.cairo#L77)

[syscalls.cairo#L77](https://github.com/starkware-libs/cairo/blob/cca08c898f0eb3e58797674f20994df0ba641983/corelib/src/starknet/syscalls.cairo#L77) [`syscalls.cairo`]

[id="storage_read"]

== `storage_read`

[discrete]

=== Function signature

[source,cairo,subs="+quotes,+macros"]

```
extern fn storage_read_syscall(  
    address_domain: u32, address: StorageAddress  
) -> SyscallResult<felt252> implicits(GasBuiltin, System) nopanic;
```

[discrete]

=== Description

Gets the value of a key in the storage of the calling contract.

This system call provides direct access to any possible key in storage, in contrast with ``var.read()``, which enables you to read storage variables that are defined explicitly in the contract.

For information on accessing storage by using the storage variables, see [xref:./contract-storage.adoc#storage_variables\[storage variables\]](#).

[discrete]

=== Arguments

[horizontal,labelwidth=35]

``_address_domain_``: `u32``:: The domain of the key, used to separate between different data availability modes. This separation is used in Starknet to offer different data availability modes. Currently, only the onchain mode (where all updates go to L1), indicated by domain ``0``, is supported. Other address domains which will be introduced in the future will behave differently in terms of publication (in particular, they will not be posted on L1, creating a tradeoff between cost and security).

``_address_``: `StorageAddress``:: The requested storage address.

[discrete]

=== Return values

* The value of the key, of type ``SyscallResult<felt252>``.

[discrete]

=== Common library

link:<https://github.com/starkware-libs/cairo/blob/cca08c898f0eb3e58797674f20994df0ba641983/corelib/src/starknet/syscalls.cairo#L60>[`syscalls.cairo`^]

[discrete]

=== Example

```
[source,cairo,subs="+quotes,+macros"]
```

```
----
```

```
use starknet::storage_access::storage_base_address_from_felt252;
```

```
...
```

```
let storage_address =  
storage_base_address_from_felt252(3534535754756246375475423547453)  
storage_read_syscall(0, storage_address).unwrap_syscall()
```

```
----
```

```
[id="storage_write"]  
== `storage_write`
```

```
[discrete]  
=== Function signature
```

```
[source,cairo,subs="+quotes,+macros"]
```

```
----
```

```
extern fn storage_write_syscall(  
    address_domain: u32, address: StorageAddress, value: felt252  
) -> SyscallResult<()> implicits(GasBuiltin, System) nopanic;
```

```
----
```

```
[discrete]  
=== Description
```

Sets the value of a key in the storage of the calling contract.

This system call provides direct access to any possible key in storage, in contrast with ``var.write()``, which enables you to write to storage variables that are defined explicitly in the contract.

For information on accessing storage by using the storage variables, see [xref:./contract-storage.adoc#storage_variables\[storage variables\]](#).

```
[discrete]  
=== Arguments
```

```
[horizontal,labelwidth=35]
```

``_address_domain_``: `u32``:: The domain of the key, used to separate between different data availability modes. This separation is used in Starknet to offer different data availability modes. Currently, only the onchain mode (where all updates go to L1), indicated by domain ``0``, is supported. Other address domains which will be introduced in the future will behave differently in terms of publication (in particular, they will not be posted on L1, creating a tradeoff between cost and security).

`_address_`: StorageAddress`:: The requested storage address.
`_value_`: felt252`:: The value to write to the key.

[discrete]
=== Return values

None.

.Common library

link:<https://github.com/starkware-libs/cairo/blob/cca08c898f0eb3e58797674f20994df0ba641983/corelib/src/starknet/syscalls.cairo#L70>[_syscalls.cairo`^]
[id="solidity_verifier"]
= Solidity verifier

include::ROOT:partial\$snippet-note-content-from-sn-book.adoc[]

Starknet's Solidity verifier ensures the truth of transactions and smart contracts.

```
// == Review: SHARP and SHARP Jobs
//
// The Shared Prover, SHARP, aggregates various Cairo programs from distinct users in
// Starknet. These programs, each with unique logic, run together, producing a common
// proof for all, optimizing cost and efficiency.
//
// === SHARP workflow
//
// image::verifier-sharp-workflow.png[SHARP Workflow]
//
// Furthermore, SHARP supports combining multiple proofs into one, enhancing its
// efficiency by allowing parallel proof processing and verification.
//
// SHARP verifies numerous Starknet transactions, like transfers, trades, and state
// updates. It also confirms smart contract executions.
//
// To illustrate SHARP: Think of commuting by bus. The bus driver, the prover,
// transports passengers, the Cairo programs. The driver checks only the tickets of
// passengers alighting at the upcoming stop, much like SHARP. The prover forms a
// single proof for all Cairo programs in a batch but verifies only the proofs of programs
// executing in the succeeding block.
//
// === SHARP Jobs
//
// Known as Shared Prover Jobs, SHARP Jobs let multiple users present their Cairo
// programs for combined execution, distributing the proof generation cost. This shared
```

approach makes Starknet more economical for users, enabling them to join ongoing jobs and leverage economies of scale.

//

// == Solidity verifiers

The Solidity verifier is an L1 Solidity smart contract, designed to validate STARK proofs from SHARP.

== Current architecture: Multiple smart contracts

The current Verifier is a set of multiple smart contracts, rather than being a singular, monolithic structure.

Some key smart contracts associated with the Verifier are:

* <https://etherscan.io/address/0x47312450b3ac8b5b8e247a6bb6d523e7605bdb60> `[+GpsStatementVerifier+`]`: This is the primary contract of the SHARP verifier. It verifies a proof and then registers the related facts using ``+verifyProofAndRegister+``. It acts as an umbrella for various layouts, each named ``+CpuFrilessVerifier+``. Every layout has a unique combination of built-in resources.

+

.Verifier layouts

image::verifier-gps-statement-verifier.png[The system routes each proof to its relevant layout]

+

The system routes each proof to its relevant layout.

* <https://etherscan.io/address/0xfd14567eaf9ba941cb8c8a94eec14831ca7fd1b4> `[+MemoryPageFactRegistry+`]`: This registry maintains facts for memory pages, primarily used to register outputs for data availability in rollup mode. The Fact Registry is a separate smart contract ensuring the verification and validity of attestations or facts. The verifier function is separated from the main contract to ensure each segment works optimally within its limits. The main proof segment relies on other parts, but these parts operate independently.

* <https://etherscan.io/address/0x5899efea757e0dbd6d114b3375c23d7540f65fa4> `[+MerkleStatementContract+`]`: This contract verifies Merkle paths.

* <https://etherscan.io/>

[address/0x3e6118da317f7a433031f03bb71ab870d87dd2dd](https://etherscan.io/address/0x3e6118da317f7a433031f03bb71ab870d87dd2dd) `[+FriStatementContract+`]`: It focuses on verifying the FRI layers.

== SHARP Verifier contract map

The SHARP Verifier Contract Map contains roughly 40 contracts, detailing various components of the Solidity verifier. The images below display the contracts and their Ethereum Mainnet addresses.

.SHARP verifier contract map

These are currently the entire stack of contracts that comprise the SHARP verifier, with their Ethereum Mainnet addresses.

[horizontal,labelwidth="25", role="stripes-odd"]

Proxy:: 0x47312450B3Ac8b5b8e247a6bB6d523e7605bDb60

CallProxy:: 0xD4C4044ACa68ebBcB81B13cC2699e1Bca2d3F458

GpsStatementVerifier:: 0x6cB3EE90C50a38A0e4662bB7e7E6e40B91361BF6

MemoryPageFactRegistry:: 0xFD14567eaf9ba941cB8c8a94eEC14831ca7fD1b4

MerkleStatementContract:: 0x5899Efea757E0Dbd6d114b3375C23D7540f65fa4

FriStatementContract:: 0x3E6118DA317f7A433031F03bB71ab870d87dd2dd

CairoBootloaderProgram:: 0x5d07afFAfc8721Ef3dEe4D11A2D1484CBf6A9dDf

PedersenHashPointsXColumn:: 0xc4f21318937017B8aBe5fDc0D48f58dBc1d18940

PedersenHashPointsYColumn:: 0x519DA5F74503dA351EbBED889111377d33096002

EcdsaHashPointsXColumn:: 0x593a71DC43e9B67FE009d7C76B6EfA925FB329B1

EcdsaHashPointsYColumn:: 0xcA59f6FD499ffF50c78Ffb420a9bcd0d273abf29

CpuFriessVerifier0:: 0x217750c27bE9147f9e358D9FF26a8224F8aCC214

CpuOods0:: 0x3405F644F9390C3478f42Fd205CE6920CcAF3280

CpuConstraintPoly0:: 0x943248dA0FFd5834Da56c5AD5308E2E2991378EB

CpuFriessVerifier1::

0x630A97901Ac29590DF83f4A64B8D490D54caf239

CpuOods1::

0x8518F459A698038B4CCED66C042c48C6bB5B17fe

CpuConstraintPoly1::

0x4CF5c11321d54b83bDAE84bBbd018c26621d2950

CpuFriessVerifier2::

0x8488e8f4e26eBa40faE229AB653d98E341cbE57B

CpuOods2::

0x52314e0b25b024c34480Ac3c75cfE98c2Ed6aa4a

CpuConstraintPoly2::

0xBE8bd7a41ba7DC7b995a53368e7fFE30Fd2BC447

CpuFriessVerifier3::

0x9E614a417f8309575fC11b175A51599661f2Bd21

CpuOods3::

0xED219933b58e9c00E66682356588d42C7932EE8E

CpuConstraintPoly3::

0x297951a67D1BF7795500C3802d21a8C846D9C962

CpuFriessVerifier4:: 0xC879aF7D5eD80e4676C203FD300E640C297F31e3

CpuOods4:: 0x4bf82e627D57cB3F455E740bcDA25848cDbd2FF7

CpuConstraintPoly4:: 0x0C099caf7a87e4eB28bcd8D0608063f8a69bb434

CpuFrilessVerifier5::
0x78Af2BFB12Db15d35f7dE8DD77f29C299C78c590
CpuOods5::
0x43A1C0bBa540e1C98d4b413F876250bdCFd0b9e0
CpuConstraintPoly5::
0x691ca565B7416B681e4f9Fb56A1283Ae8b34E55e

CpuFrilessVerifier6::
0xe9664D230490d5A515ef7Ef30033d8075a8D0E24
CpuOods6::
0x68293272FEA2D6e74572BC18ffaD11F21344e090
CpuConstraintPoly6::
0xd0aAdECA2d25AEFde0da214d27b04b6ea20D7418

PoseidonPoseidonFullRoundKey0Column6::
0x37070Fd8051f63E5A6D7E87026e086Cc19db1aBe
PoseidonPoseidonFullRoundKey1Column6::
0xb4711a4614368516529d6118C97905aB4B28e267
PoseidonPoseidonFullRoundKey 2Column6::
0x4FB05b7CC348C5a72C59a3f307baf66e3CA1F835
PoseidonPoseidonPartialRoundKey0Column6::
0x812c2AD2161D099724A99C8114c539b9e5b449cd
PoseidonPoseidonPartialRoundKey1Column6::
0x4d0E80AB34ee2B19295F2CaC3101d03452D874b8
CpuFrilessVerifier7::
0x03Fa911dfCa026D9C8Edb508851b390accF912e8
CpuOods7::
0xc9E067AF5d00eb4aA2E73843ac36AfF83C5CeeD3
CpuConstraintPoly7::
0x89B7a7276cBc8Cb35Ec11fAE9da83b20Db3edf20

These contracts function as follows:

* ***Proxy***: This contract facilitates upgradability. It interacts with the `+GpsStatementVerifier+` contract using the `+delegate_call+` method. Notably, the state resides in the `+GpsStatementVerifier+` contract, not in the proxy.

* ***CallProxy***: Positioned between the `+Proxy+` and the `+GpsStatementVerifier+` contracts, it functions like a typical proxy. However, it avoids the `+delegate_call+` method and calls the function in the implementation contract directly.

* ***CairoBootloaderProgram***: Comprising numerical Cairo programs, it validates the Cairo program of a statement. The bootloader manages the logic executing Cairo programs to generate proof and program hash.

* ***PedersenHashPoints (X & Y Columns)***: These lookup tables store vast amounts of data. Validation functions consult them to compute the Pedersen hash.

- * *EcdsaPoints (X & Y Column)*: Similar to the Pedersen hash, these tables assist in calculating the elliptic curve.
- * *CpuFrilessVerifier/CpuOods/CpuConstantPoly (0 - 7)*: These Verifier contracts vary in layout as shown in the GpsStatementVerifier layout image. Each layout encompasses resources, built-ins, constraints, and more, designed for a specific task. Each has unique parameters for its constructor.
- * *PoseidonPoseidon*: These contracts back the new Poseidon built-in and contain Poseidon-specific lookup tables.

== Constructor parameters of key contracts

When constructing the primary Verifier contracts, specific parameters are employed to facilitate functionality. These parameters reference other auxiliary contracts, decentralizing the logic and ensuring the main contract remains under the 24kb deployment limit.

Below is a visual representation of these parameters in relation to key contracts CpuFrilessVerifiers and GpsStatementVerifier:

image::verifier-constructor-params.png[Constructor Parameters]

=== CpuFrilessVerifier constructor parameters

CpuFrilessVerifiers is designed to handle a diverse range of tasks. Its parameters encompass:

- * *Auxiliary Polynomial Contracts:* These include ``+CpuConstraintPoly+``, ``+PedersenHashPointsXColumn+``, ``+PedersenHashPointsYColumn+``, ``+EcdsaPointsXColumn+``, and ``+EcdsaPointsYColumn+``.
- * *Poseidon-Related Contracts:* Several ``+PoseidonPoseidonFullRoundKey+`` and ``+PoseidonPoseidonPartialRoundKey+`` contracts.
- * *Sampling and Memory:* The contract uses ``+CpuOods+`` for out-of-domain sampling and ``+MemoryPageFactRegistry+`` for memory-related tasks.
- * *Verification:* It integrates with ``+MerkleStatementContract+`` for Merkle verification and ``+FriStatementContract+`` for Fri-related tasks.
- * *Security:* The ``+num_security_bits+`` and ``+min_proof_of_work_bits+`` contracts ensure secure operation.

[NOTE]

=====

For instances like ``+CpuFrilessVerifier0+``, specific contracts (e.g., ``+CpuConstraintPoly0+``, ``+PoseidonPoseidonFullRoundKeyColumn0+``, ``+CpuOods0+``) become particularly relevant.

=====

=== GpsStatementVerifier constructor parameters

The ``+GpsStatementVerifier+`` functions as the hub of verifier operations, necessitating various parameters for effective functioning:

- * **Bootloader:** It references the ``+CairoBootloaderProgram+`` to initiate the system.
- * **Memory Operations:** This is facilitated by the ``+MemoryPageFactRegistry+`` contract.
- * **Sub-Verifiers:** It integrates a series of sub-verifiers (``+CpuFriessVerifier0+`` through ``+CpuFriessVerifier7+``) to decentralize tasks.
- * **Verification:** The hashes, ``+hashed_supported_cairo_verifiers+`` and ``+simple_bootloader_program_hash+``, are essential for validation processes.

== Interconnection of contracts

The ``+GpsStatementVerifier+`` serves as the primary verifier contract, optimized for minimal logic to fit within deployment size constraints. To function effectively:

- * It relies on smaller verifier contracts, which are already deployed and contain varied verification logic.
- * These smaller contracts, in turn, depend on other contracts, established during their construction.

In essence, while the diverse functionalities reside in separate contracts for clarity and size efficiency, they are all interlinked within the ``+GpsStatementVerifier+``.

For future enhancements or adjustments, the proxy and callproxy contracts facilitate upgradability, allowing seamless updates to the ``+GpsStatementVerifier+`` without compromising its foundational logic.

== SHARP verification flow

image::verifier-sharp-verification-flow.png[SHARP verification flow]

- . The SHARP dispatcher transmits all essential transactions for verification, including:
 - * ``+MemoryPages+`` (usually many).
 - * ``+MerkleStatements+`` (typically between 3 and 5).
 - * ``+FriStatements+`` (generally ranging from 5 to 15).
- . The SHARP dispatcher then forwards the proof using ``+verifyProofAndRegister+``.
- . Applications, such as the Starknet monitor, validate the status. Once verification completes, they send an ``+updateState+`` transaction.

```
// :invoke:  
// :declare:  
// :deploy_account:  
// :pre_v3:
```

```
[cols="1,1,3"]
|===
|Field name |Type |Description
```

```
ifdef::declare,invoke[]
// -----NEW TABLE ROW-----
|`account_deployment_data` |`List<FieldElement>` |
// UNTIL FURTHER NOTICE USE THIS TEXT:
_For future use._
```

Currently this value is always empty.

For more information, see link:<https://github.com/starknet-io/SNIPs/blob/main/SNIPS/snip-8.md>[SNIP 8: Transaction V3 Structure] +

```
// DO NOT PUT IN DOCS YET:
// Used for enabling a paymaster.
// The list will contain the class_hash and the calldata needed for the constructor.
// In the future, we might want to use Invoke instead of deploy_account, same as in
EIP-4337. In that case, the sender address does not exist - the sequencer will try to
deploy a contract with the class hash specified in account_deployment_data.
```

```
// Transaction versions that support this field
// Declare 3
// Invoke 3
endif::declare,invoke[]
```

```
ifdef::invoke[]
// -----NEW TABLE ROW-----
|`calldata` |`List<FieldElement>` |The arguments that are passed to the validate and
execute functions.
// Transaction versions that support this field
// Invoke: 0, 1, 3
endif::invoke[]
```

```
// -----NEW TABLE ROW-----
|`chain_id` |`FieldElement` |The id of the chain to which the transaction is sent.
// Transaction versions that support this field
// Declare: 0, 1, 2, 3
// Invoke: 0, 1, 3
// Deploy account: 1, 3
```

```
ifdef::deploy_account[]
// -----NEW TABLE ROW-----
|`class_hash` |`FieldElement` |The hash of the desired account class. For more
information, see xref:architecture-and-concepts:smart-contracts/class-hash.adoc[Class
```

```

hash].
// Transaction versions that support this field +
// Deploy account: 1, 3
endif::deploy_account[]

ifdef::declare[]
// -----NEW TABLE ROW-----
|`compiled_class_hash` |`FieldElement` |The hash of the compiled class. For more
information, see xref:architecture-and-concepts:smart-contracts/class-hash.adoc[Class
hash].
// Transaction versions that support this field +
// Declare 2, 3
endif::declare[]

ifdef::deploy_account[]
// -----NEW TABLE ROW-----
|`constructor_calldata` |`List<FieldElement>` |The arguments to the account
constructor. +
// Transaction versions that support this field +
// Deploy account: 1, 3
endif::deploy_account[]

ifdef::deploy_account[]
// -----NEW TABLE ROW-----
|`contract_address_salt` |`FieldElement` |A random salt that determines the account
address. +
// Transaction versions that support this field +
// Deploy account: 1, 3
endif::deploy_account[]

ifdef::declare[]
// -----NEW TABLE ROW-----
|`contract_class` |`ContractClass` |The class definition. For more information, see
xref:architecture-and-concepts:smart-contracts/class-hash.adoc[Class hash].

// Transaction versions that support this field +
// Declare 0, 1, 2, 3
endif::declare[]

// -----NEW TABLE ROW-----
|`fee_data_availability_mode` |`FieldElement` |
// UNTIL FURTHER NOTICE USE THIS TEXT:
_For future use._

```

Currently this value is always `0`.

For more information, see link:<https://github.com/starknet-io/SNIPs/blob/main/SNIPS/snip-8.md>[SNIP 8: Transaction V3 Structure] +

```
// DO NOT PUT IN DOCS YET:  
// Used for enabling Volition mode.  
// 0=L1DA (Default: 0)  
// 1=L2DA
```

```
// Transaction versions that support this field +  
// Declare: 3 +  
// Invoke: 3 +  
// Deploy account: 3
```

```
ifdef::pre_v3[]  
// -----NEW TABLE ROW-----  
|`max_fee`|`FieldElement`|The maximum fee that the sender is willing to pay for the  
transaction. +
```

```
// Deprecated.  
// v3 Transactions use `resource_bounds`  
// Transaction versions that support this field +  
// Declare: 0, 1, 2 +  
// Invoke: 0, 1 +  
// Deploy account: 1  
endif::pre_v3[]
```

```
// -----NEW TABLE ROW-----  
|`nonce`|`FieldElement`|The transaction nonce.  
// Transaction versions that support this field +  
// Declare: 1, 2, 3 +  
// Invoke: 1, 3 +  
// Deploy account: 1, 3
```

```
// -----NEW TABLE ROW-----  
|`nonce_data_availability_mode`|`FieldElement`|  
// UNTIL FURTHER NOTICE USE THIS TEXT:  
_For future use._
```

Currently this value is always `0`.

For more information, see link:<https://github.com/starknet-io/SNIPs/blob/main/SNIPS/snip-8.md>[SNIP 8: Transaction V3 Structure]

```
// NOT SUPPORTED YET:  
// Used for enabling Volition mode.
```

```
// 0=L1DA (Default: 0)
// 1=L2DA

// Transaction versions that support this field +
// Declare: 3 +
// Invoke: 3 +
// Deploy account: 3

// -----NEW TABLE ROW-----
|`paymaster_data` |`List<FieldElement>` |
// UNTIL FURTHER NOTICE USE THIS TEXT:
_For future use._
```

Currently this value is always empty.

For more information, see link:<https://github.com/starknet-io/SNIPs/blob/main/SNIPS/snip-8.md>[SNIP 8: Transaction V3 Structure]

```
// NOT SUPPORTED YET: Used for enabling a paymaster. Represent the address of
paymaster sponsoring the transaction, followed by extra data to send to the paymaster
(empty for self-sponsored transaction) +
// The default value is an empty list, indicating no paymaster. +
```

```
// Transaction versions that support this field +
// Declare: 3 +
// Invoke: 3 +
// Deploy account: 3
```

```
// -----NEW TABLE ROW-----
|`resource_bounds` |`Dict[Resource, ResourceBounds]` a|Used for enabling the fee
market.
```

A dictionary that maps resource type to resource bounds. The `_resource_` is the amount of L1 or L2 gas used to pay for the transaction.

``Resource``:: A felt. Possible values are the felt representation of the strings ``L1_GAS`` or ``L2_GAS``.

``ResourceBounds``:: A struct containing the following felts:

* ``max_amount``: The maximum amount of the resource allowed for usage during the execution. +

* ``max_price_per_unit``: The maximum price the user is willing to pay for the resource. +
``L1_GAS`` and ``L2_GAS`` are specified in units of fri, where $1 \text{ fri} = 10^{-18} \text{ STRK}$.

```
// Transaction versions that support this field +
// Declare: 3 +
// Invoke: 3 +
```

```

// Deploy account: 3

ifdef::declare,invoke[]
// -----NEW TABLE ROW-----
|`sender_address` |`FieldElement` |The address of the account initiating the transaction.

// Transaction versions that support this field
// Declare 0, 1, 2, 3
// Invoke 0, 1, 3
endif::declare,invoke[]

// -----NEW TABLE ROW-----
|`signature` |`List<`FieldElement`>` |Additional information given by the sender, used to
validate the transaction.

// Transaction versions that support this field +
// Declare: 0, 1, 2, 3
// Invoke: 0, 1, 3
// Deploy account: 1, 3

// -----NEW TABLE ROW-----
|`tip` |`FieldElement`
| For future use. Currently this value is always `0`.

// NOT SUPPORTED YET: +
// Used for enabling the fee market. +
//
// The amount of a tip you can offer when sending a transaction. The prioritization
metric determines the sorting order of transactions in the mempool. +

// Until further notice use this text:
// Transaction versions that support this field +
// Declare: 3 +
// Invoke: 3 +
// Deploy account: 3

// -----NEW TABLE ROW-----
|`version` |`FieldElement` a|The transaction's version. +
When the fields that comprise a transaction change, either with the addition of a new
field or the removal of an existing field, then the transaction version increases. +

Transaction version, where `n` specifies version `n` transaction. For example:

[horizontal,labelwidth="4"]
`3`:: version 3 transaction

```



```

// Transaction versions that support this field +
// Declare: 0, 1, 2, 3 +
// Invoke: 0, 1, 3 +
// Deploy account: 1, 3
|===
* `chain_id` is a constant value that specifies the network to which this transaction is
sent. See xref:chain-id[Chain-Id].
* `l1_gas_bounds` is constructed as follows:
+
[stem]
++++
\underbrace{\text{L1_GAS}}_{\text{60 bits}} | \underbrace{\text{max\_amount}}_{\text{64
bits}} |
\underbrace{\text{max\_price\_per\_unit}}_{\text{128 bits}}
++++
* `l2_gas_bounds` is constructed as follows:
+
[stem]
++++
\underbrace{\text{L2_GAS}}_{\text{60 bits}} | \underbrace{\text{max\_amount}}_{\text{64
bits}} |
\underbrace{\text{max\_price\_per\_unit}}_{\text{128 bits}}
++++
* `data_availability_modes` is a concatenation of `fee_data_availability_mode`
and `nonce_data_availability_mode`, as follows:
+
[stem]
++++
\underbrace{0\cdots0}_{\text{188 bits}} |
\underbrace{\text{nonce\_data\_availability\_mode}}_{\text{32 bits}} |
\underbrace{\text{fee\_data\_availability\_mode}}_{\text{32 bits}}
++++
* `_h_` is the xref:../cryptography/hash-functions.adoc#poseidon_hash[Poseidon] hash.
[id="legacy"]
= Legacy Starknet CLI reference

[WARNING]
====
The Starknet CLI is deprecated. Instead, use xref:tools/devtools.adoc#starkli[StarkLi
CLI].
====

[id="basic_command_line_syntax"]
== Basic command line syntax

```

To enter a Starknet command, use the following syntax:

```
[source,bash]
```

```
----
```

```
$ starknet <command> <options>
```

```
----
```

Where:

`<command>` represents a single command that executes an operation on Starknet.

`<options>` represents zero or more command line options, each of which modifies the operation of the command.

```
[id="setting_the_starknet_network_environment"]
```

```
== Setting the Starknet network environment
```

You need to set your Starknet network environment to use either testnet or Mainnet.

You can set the environment using either a command-line option or an environment variable.

Possible values are:

`sepolia`:: Sets the Starknet network to testnet

`mainnet`:: Sets the Starknet network to Mainnet

.Setting the network environment using a command-line option

When you enter any command, include the `--network` option. For example to use Mainnet, enter a command as follows:

```
[source,bash]
```

```
----
```

```
$ starknet <command> --network alpha-mainnet <options>
```

```
----
```

```
[NOTE]
```

```
=====
```

You can place the `--network` option before or after any other option.

```
=====
```

.Setting the network environment using an environment variable

Set the `STARKNET_NETWORK` environment variable as follows:

```
[source,bash]
```

```
----  
$ export STARKNET_NETWORK=<starknet_network>  
----
```

For example, to use testnet, enter the following command:

```
[source,bash]  
----  
$ export STARKNET_NETWORK=sepolia  
----
```

== Setting custom endpoints

When working with the CLI, you can manually set the endpoints for the gateways that enable you to interact with Starknet, by including the following options:

`--feeder_gateway_url` :: Sets the custom endpoint for read commands.
`--gateway_url` :: Sets the custom endpoint for write commands.

The following are the endpoints for Starknet testnet and Mainnet:

- * Testnet feeder gateway URL: https://alpha4.starknet.io/feeder_gateway/
- * Mainnet feeder gateway URL: https://alpha-mainnet.starknet.io/feeder_gateway/
- * Testnet gateway URL: <https://alpha4.starknet.io/gateway/>
- * Mainnet gateway URL: <https://alpha-mainnet.starknet.io/gateway/>

.Example: Setting a custom read endpoint

The following command returns the ABI using the Mainnet feeder gateway.

```
[source,bash]  
----  
$ starknet get_code --feeder_gateway_url https://alpha-mainnet.starknet.io/  
feeder_gateway/ <options>  
----
```

.Example: Setting a custom write endpoint

The following command sends a transaction to the Starknet sequencer using the Mainnet gateway.

```
[source,bash]  
----  
$ starknet invoke --gateway_url https://alpha-mainnet.starknet.io/gateway/ <options>  
----
```

```
[id="starknet_call"]  
== `starknet call`
```

```
[source,terminal]
```

```
----
```

```
starknet call  
  --address <contract_address>  
  --abi <contract_abi>  
  --function <function_name>  
  --inputs <arguments>  
  --block_hash <block_hash>  
  --block_number <block_number>  
  --signature <signature_information>  
  --wallet <wallet_name>  
  --nonce <nonce>  
----
```

Calls a Starknet contract without affecting the state, accepts the following arguments:

- `contract_address` - address of the contract being called
- `contract_abi` - a path to a JSON file that contains the link:[https://www.cairo-lang.org/docs/hello_starknet/intro.html#the-contract-s-abi\[abi\]](https://www.cairo-lang.org/docs/hello_starknet/intro.html#the-contract-s-abi[abi]) of the contract being called
- `function_name` - name of the function which is called
- `arguments` - inputs to the function being called, represented by a list of space-delimited values
- `block_hash` - the hash of the block used as the context for the call operation. If this argument is omitted, the latest block is used
- `block_number` - same as block_hash, but specifies the context block by number or xref:block_tag[tag]
- `signature_information` - list of field elements as described xref:architecture-and-concepts:network-architecture/transactions.adoc#signature[here]
- `wallet_name` - the name of the desired wallet, use xref:starknet_deploy_account[deploy_account] to set-up new accounts in the CLI
- `nonce` - account nonce, only relevant if the call is going through an account

```
[id="block_tag"]
```

```
[NOTE]
```

```
=====
```

```
.Block Tag
```

A block context can be specified via the `latest` or `pending` tags, where the former

refers to the latest accepted on L2 block and the latter refers to the `xref:architecture-and-concepts:network-architecture/transaction-life-cycle.adoc#the-pending-block[pending block]`.

====

```
[id="starknet_declare"]  
== `starknet declare`
```

```
[source,terminal]  
----  
starknet declare  
--contract <contract_class>  
----
```

Declares a new contract class on Starknet, accepts the following argument:

- ``contract_class`` - path to a JSON file containing the contract's compiled code

```
[id="starknet_deploy"]  
== `starknet deploy`
```

```
[source,terminal]  
----  
starknet deploy  
  --salt <salt>  
  --contract <contract_definition>  
  --inputs <constructor_inputs>  
  --token <token>  
----
```

Deploys a new contract, accepts the following arguments:

- ``salt`` - a seed that is used in the computation of the contract's address (if not specified, the sequencer will choose a random string)
- ``contract_definition`` - path to a JSON file containing the contract's bytecode and abi (can be obtained by executing [link:https://www.cairo-lang.org/docs/hello_starknet/intro.html#compile-the-contract\[starknet-compile\]](https://www.cairo-lang.org/docs/hello_starknet/intro.html#compile-the-contract[starknet-compile]))
- ``constructor_inputs`` - the arguments given to the contract's constructor, represented by a list of space-delimited values
- ``token`` - a token allowing contract deployment (can be obtained by applying [link:https://forms.reform.app/starkware/SN-Alpha-Contract-Deployment/l894lu\[here\]](https://forms.reform.app/starkware/SN-Alpha-Contract-Deployment/l894lu[here])). Only used in the Alpha stages and will be deprecated in the future

[NOTE]

====

The deploy token is a temporary measure which will be deprecated when fees are incorporated in the system. Only relevant for Mainnet.

====

```
[id="starknet_deploy_account"]  
== `starknet deploy_account`
```

[source,terminal]

```
starknet deploy_account  
  --wallet <wallet_provider>  
  --account <account_name>
```

Deploys an account contract, accepts the following arguments:

- `account_name` - the name given to the account, used for managing multiple accounts from the CLI (if not specified, the name `+__default__+` is used.
- `wallet_provider`* - the path to module which manages the account (responsible for key generation, signing, etc.)

[NOTE]

====

Today, the Starknet CLI only works with the link:https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/third_party/open_zeppelin/Account.cairo[OpenZeppelin account contract].

The CLI uses this specific link:https://github.com/starkware-libs/cairo-lang/blob/master/src/starkware/starknet/wallets/open_zeppelin.py[wallet provider].

To use this provider, either set up the following environment variable or pass the same value directly to the `wallet_provider` parameter:

[source,bash]

```
$ export  
STARKNET_WALLET=starkware.starknet.wallets.open_zeppelin.OpenZeppelinAccount
```

====

[CAUTION]

====

Using the built-in wallet providers that are part of the cairo-lang package (starkware.starknet.wallets...) is not secure (for example, the private key may be kept

not encrypted and without backup in your home directory). You should only use them if you're not overly concerned with losing access to your accounts (for example, for testing purposes).

====

```
[id="starknet-estimate_fee"]  
== `starknet estimate_fee`
```

```
[source,terminal]
```

```
starknet estimate_fee  
  --address <contract_address>  
  --abi <contract_abi>  
  --function <function_name>  
  --inputs <arguments>
```

Returns the fee estimation for a given contract call. Accepts the following arguments:

- `address`* - the address of the contract being called
- `contract_abi`* - a path to a JSON file that contains the xref:architecture-and-concepts:smart-contracts/contract-abi.adoc[abi] of the contract being called
- `function_name`* - the name of the function being called
- `arguments`* - inputs to the function being called, represented by a list of space-delimited values`

```
== `starknet estimate_message_fee`
```

```
[source,terminal]
```

```
starknet estimate_message_fee  
  --from_address <sender_address>  
  --to_address <contract_address>  
  --function <function_name>  
  --inputs <arguments>
```

Returns the fee estimation for a given xref:architecture-and-concepts:network-architecture/messaging-mechanism.adoc#l1-l2-message-fees[L1 handler] application. Accepts the following arguments:

- `from_address`* - the L1 address of the sender
- `to_address`* - the L2 address of the recipient
- `contract_abi`* - a path to a JSON file containing the xref:architecture-and-

concepts:smart-contracts/contract-abi.adoc[abi] of the receiving contract on L2

- ``function_name`` - the name of the desired L1 handler
- ``arguments`` - inputs to the called handler, represented by a list of space-delimited values

== ``starknet get_block``

[source,terminal]

```
starknet get_block
  --hash <block_hash>
  --number <block_number>
```

Returns the requested block, exactly one of the following arguments must be given:

- * ``block_hash`` - hash of the requested block
- * ``block_number`` - number or `<<block_tag,tag>>` of the requested block

[id="starknet_get_code"]

== ``starknet get_code``

[source,terminal]

```
starknet get_code
  --contract_address <contract_address>
  --block_hash <block_hash>
  --block_number <block_number>
```

Returns the ABI and the byte code of the requested contract, accepts the following arguments:

- ``contract_address`` - address of the requested contract
- ``block_hash`` - the hash of the block used as the context for the operation. If this argument is omitted, the latest block is used
- ``block_number`` - same as `block_hash`, but specifies the context block by number or `xref:block_tag[tag]`

== ``starknet get_storage_at``

[source,terminal]

```
starknet get_storage_at
  --contract_address <contract_address>
```



```
--key <key>
--block_hash <block_hash>
--block_number <block_number>
----
```

Queries a contract's storage at a specific key, accepts the following arguments:

```
* `contract_address` *- address of the requested contract
* `key` *- the requested key from the given contract's storage
* `block_hash` - the hash of the block relative to which the storage will be provided. In
case this argument is not given, the latest block is used
* `block_number` - same as block_hash, but specifies the context block by number or
<<block_tag,tag>>
```

```
[id="starknet_get_transaction"]
== `starknet get_transaction`
```

```
[source,terminal]
----
starknet get_transaction --hash <transaction_hash>
----
```

Returns the requested transaction, expects the following argument:

```
- `transaction_hash` *- hash of the requested transaction
```

```
== `starknet get_transaction_receipt`
```

```
[source,terminal]
----
starknet get_transaction_receipt --hash <transaction_hash>
----
```

Returns the [xref:architecture-and-concepts:network-architecture/transaction-life-cycle.adoc#transaction-receipt\[receipt\]](#) associated with the transaction, expects the following argument:

```
* `transaction_hash` *- hash of the requested transaction
starknet invoke
starknet tx_status
```

```
== `starknet invoke`
```

[source,terminal]

```
starknet invoke
--address <contract_address>
--abi <contract_abi>
--function <function_name>
--inputs <arguments>
--signature <signature_information>
--wallet <wallet_name>
--nonce <nonce>
```

Sends a transaction to the Starknet sequencer, accepts the following arguments:

- * `address` - the address of the contract being called
- * `contract_abi` - a path to a JSON file that contains the https://www.cairo-lang.org/docs/hello_starknet/intro.html#the-contract-s-abi of the contract being called
- * `function_name` - the name of the function being called
- * `arguments` - inputs to the function being called, represented by a list of space-delimited values
- * `signature_information` - list of field elements as described [xref:architecture-and-concepts:network-architecture/transactions.adoc#signature\[here\]](#)
- * `wallet_name` - the name of the desired wallet, use [xref:starknet_deploy_account\[deploy_account\]](#) to set-up new accounts in the CLI.
- * `nonce` - account nonce, only relevant if the call is going through an account

[TIP]

=====

Today, interaction with Starknet may be done either via account or by a direct call. The `signature` argument can only be provided in the case of a direct call, since otherwise providing the signature is the responsibility of the account module. To use an account you must specify `wallet_name`, otherwise a direct call will be used (you may also explicitly perform a direct call by adding `--no_wallet` to the command). Note that in the future direct calls will be deprecated and the only way to interact with the system would be through accounts.

=====

== `starknet tx_status`

[source,terminal]

```
starknet tx_status
--hash <transaction_hash>
--contract <contract_definition>
```

--error_message

Returns the transaction status, accepts the following arguments:

- * `transaction_hash` - hash of the requested transaction
- * `contract_definition` - path to a JSON file containing the compiled contract to which the transaction was addressed. If supplied, the debug information from the compiled contract will be used to add error locations.
- * `error_message` - if specified, only the error message will be returned (or empty response in case the transaction was successful)

The possible statuses of a transaction are:

- * `NOT_RECEIVED`
- * `RECEIVED`
- * `PENDING`
- * `REJECTED`
- * `ACCEPTED_ON_L2`
- * `ACCEPTED_ON_L1`

For more information, see [xref:architecture-and-concepts:network-architecture/transaction-life-cycle.adoc](#)[Transaction lifecycle].

[id="starknet-compiler-options"]

= Legacy compiler CLI reference

When the Starknet compiler is installed, you can view this command-line help in a terminal by entering the following command:

[source,bash]

starknet-compile --help

== Usage

[source,bash,subs="+quotes,+macros"]

```
starknet-compile [-h] [--abi _ABI_] [--disable_hint_validation]
                  [--account_contract] [--dont_filter_identifiers] [-v]
                  [--prime __PRIME__] [--cairo_path CAIRO_PATH]
                  [--preprocess] [--output __OUTPUT__] [--no_debug_info]
                  [--debug_info_with_source]
                  [--cairo_dependencies __CAIRO_DEPENDENCIES__]
                  [--no_opt_unused_functions]
                  file [__file__ ...]
```

== Example

The following example compiles the file ``contract.cairo``. It generates two files:

[horizontal]

`contract_compiled.json::` The contract class. This file contains the bytecode and all other information necessary to execute a contract. For information on contract classes, see [xref:architecture-and-concepts:smart-contracts/contract-classes.adoc\[\]](#).

`contract_abi.json::` The contract's ABI.

[source,shell]

```
starknet-compile contract.cairo \  
  --output contract_compiled.json \  
  --abi contract_abi.json
```

== Description

A tool to compile Starknet contracts.

== Positional arguments

=== file

File names.

== Optional arguments

=== ``-h`, `--help``

Show this help message and exit.

=== ``--abi _ABI_``

Output the contract's ABI.

=== ``--disable_hint_validation``

Disable the hint validation.

=== `--account_contract`

Compile as account contract.

=== `--dont_filter_identifiers`

Disable the filter-identifiers-optimization. If True, all the identifiers will be kept, instead of just the ones mentioned in hints or 'with_attr' statements.

=== `-v`, `--version`

show program's version number and exit

=== `--prime _PRIME_`

The size of the finite field.

=== `--cairo_path _CAIRO_PATH_`

A list of directories, separated by ":" to resolve import paths. The full list will consist of directories defined by this argument, followed by the environment variable CAIRO_PATH, the working directory and the standard library path.

=== `--preprocess`

Stop after the preprocessor step and output the preprocessed program.

=== `--output _OUTPUT_`

The output file name (default: `stdout`).

=== `--no_debug_info`

Don't include debug information in the compiled file.

=== `--debug_info_with_source`

Include debug information with a copy of the source code.

=== `--cairo_dependencies _CAIRO_DEPENDENCIES_`

Output a list of the Cairo source files used during the compilation as a CMake file.

=== `--no_opt_unused_functions`

Disables unused function optimization.* Quick start

```
** xref:environment-setup.adoc[Setting up your environment]
// *** xref:environment-setup.adoc#installing_starkli[Installing Starkli]
// *** xref:environment-
setup.adoc#setting_environment_variables_for_starkli[Environment variables]
// *** xref:environment-setup.adoc#installing_scarb[Installing Scarb]

** xref:set-up-an-account.adoc[Setting up an account]
// *** xref:set-up-an-account.adoc#creating_an_account[Creating an account]
// *** xref:set-up-an-account.adoc#deploying_an_account[Deploying an account]

** xref:declare-a-smart-contract.adoc[Declaring a smart contract]

** xref:deploy-a-smart-contract.adoc[Deploying a smart contract]

** xref:interact-with-a-smart-contract.adoc[Interacting with a smart contract]

** xref:using_devnet.adoc[]

// ** xref:deploy-interact-with-a-smart-contract-remix.adoc[Deploying and interacting
with a smart contract with Remix]
= Declaring a smart contract
```

== Prerequisites

=== Ensure Starkli and Scarb are installed correctly
Ensure that the below commands are working properly on your system.

```
[source, bash]
----
starkli --version
scarb --version
----
```

If either of the above commands fails, see xref:environment-setup.adoc[Setting up your environment].

== Introduction

Deploying a smart contract in Starknet requires two steps:

- * Declaring the class of your contract, i.e. sending your contract's code to the network.
- * Deploying a contract, i.e. creating an instance of the code you previously declared.

[TIP]

====

If you require a smart contract for testing, you can use this sample contract, link:<https://github.com/starknet-edu/starknetbook/blob/main/examples/vote-contracts/src/lib.cairo> [lib.cairo], from the Starknet Book.

====

== Compiling a smart contract

You can compile a smart contract using the Scarb compiler.

To compile a smart contract, create a directory containing a `Scarb.toml` file and a subdirectory named `src` containing your contract source code.

Add the following code to the `Scarb.toml` file:

```
[source, toml]
```

```
----
```

```
[package]
```

```
name = "contracts"
```

```
version = "0.1.0"
```

```
[dependencies]
```

```
starknet = ">=2.2.0"
```

```
[[target.starknet-contract]]
```

```
sierra = true
```

```
----
```

Navigate into the newly created directory:

```
[source, bash]
```

```
----
```

```
cd <dir_name>
```

```
----
```

Run the following command:

```
[source, bash]
```

```
----
```

```
scarb build
```

```
----
```

The compiled contract will be saved in the `target/dev/` directory.

The contract is now compiled and ready to be deployed. Next you will need to declare

an RPC provider within your contract.

== Setting an RPC provider

To interact with the Starknet network, you need to set an RPC endpoint within Starkli.

The following are the RPC providers available for Starknet:

[cols="1,2"]

|===

|Provider name |Description

|Infura or Alchemy

|Use a provider like Infura or Alchemy.

|Custom configuration

|Set up your own node and use the RPC provider of your node. More information on this can be found within the link:https://book.starknet.io/chapter_4/node.html[Starknet Book].

|===

For demonstration purposes, the Starknet Sequencer's Gateway is used in the below steps.

== Declaring a smart contract

A contract can be declared on Starknet using the following command:

[source,bash]

```
starkli declare target/dev/<NAME>.json --network=sepolia --compiler-version=2.1.0
```

[NOTE]

=====

The `--network` flag is used to specify the network you want to use, it could also be `mainnet` for example.

The `--compiler-version` flag is used to specify the version of the compiler you want to use. Starkli is currently running on version 2.6.x of the compiler.

=====

You can find the compiler version supported by Starkli by running:


```
[source,bash]
----
starkli declare --help
----
```

In the `--compiler-version` flag you will see possible versions of the compiler:

```
[source,bash]
----
--compiler-version <COMPILER_VERSION>
    Statically-linked Sierra compiler version [possible values: 2.0.1, 2.1.0]
----
```

However, the Scarb compiler version may be `2.2.0`, you can find this out by running:

```
[source,bash]
----
scarb --version
----
```

This is because Starkli and Scarb are not always in sync.

In this case you would need to use the compiler version that Starkli is using by installing a previous version of Scarb. See the <https://github.com/software-mansion/scarb/releases>[Scarb github repo] for more detail.

You can do this by running the following command for installing Scarb version `0.6.1`:

```
[source,bash]
----
curl --proto '=https' --tlsv1.2 -sSf https://docs.swmansion.com/scarb/install.sh | sh -s -- -
v 0.6.1
----
```

If you were using a provider like Infura or Alchemy, the declaration command would look like this:

```
[source,bash]
----
starkli declare target/dev/contracts_Ownable.sierra.json \
    --rpc=https://starknet-sepolia.infura.io/v3/<API_KEY> \
    --compiler-version=2.6.0
----
```

== Expected result

The result of the declaration command is a contract class hash:

```
[source,bash]
```

```
----
```

```
Class hash declared: <CLASS_HASH>
```

```
----
```

This hash is the identifier of the contract class in Starknet. You can think of it as the address of the contract class. You can use a block explorer like <https://testnet.starkscan.co/class/0x00e68b4b07aeec72f768b1c086d9b0aadce131a40a1067ffb92d0b480cf325d>[StarkScan] to see the contract class hash in the blockchain.

If the contract you are declaring has previously been declared by someone else, you will get an output like this:

```
[source,bash]
```

```
----
```

```
Not declaring class as its already declared. Class hash: <CLASS_HASH>
```

```
----
```

```
= Deploying a smart contract
```

```
== Prerequisites
```

```
=== Ensure Starkli and Scarb are installed correctly
```

Ensure that the below commands are working properly on your system.

```
[source, bash]
```

```
----
```

```
starkli --version
```

```
scarb --version
```

```
----
```

If either of the above commands fail, please visit [xref:environment-setup.adoc](#)[Setting up your environment].

```
== Introduction
```

Deploying a smart contract in Starknet requires two steps:

- * [xref:declare-a-smart-contract.adoc](#)[Declaring] the class of your contract, i.e. sending your contract's code to the network.

- * Deploying a contract, i.e. creating an instance of the code you previously declared.

```
== Deploying a smart contract
```

Deploying a smart contract involves instantiating it on Starknet. The deployment

command requires the class hash of the smart contract and any arguments expected by the constructor.

For our example, the constructor expects an address to assign as the owner:

```
[source,bash]
----
starkli deploy \
  <CLASS_HASH> \
  <CONSTRUCTOR_INPUTS> \
  --network=sepolia
----
```

With the class hash and constructor inputs, the command looks like this:

```
[source,bash]
----
starkli deploy \
  0x00e68b4b07aeec72f768b1c086d9b0aadce131a40a1067ffb92d0b480cf325d \
  0x02cdAb749380950e7a7c0deFf5ea8eDD716fEb3a2952aDd4E5659655077B8510 \
  --network=sepolia
----
```

== Expected result

After running the command and adding your password, you will see an output similar to this:

```
[source,bash]
----
Deploying class
0x00e68b4b07aeec72f768b1c086d9b0aadce131a40a1067ffb92d0b480cf325d with
salt 0x04bc3fc2284c8e41fb3d2a37bb0354fd0506131cc77a8c91e4e67ce3aed1d19e...
The contract will be deployed at address
0x014825acb37c36563d3b96c450afe363d2fdfa3cfbd618b323f95b68b55ebf7e
Contract deployment transaction:
0x0086972e7463d5673d8b553ae521ec2df974a97c2ce6aafc1d1c20d22c6b96c6
Contract deployed:
0x014825acb37c36563d3b96c450afe363d2fdfa3cfbd618b323f95b68b55ebf7e
----
```

The smart contract has now been deployed to Starknet.

[id="interacting-with-a-smart-contract-with-remix"]

= Deploying and interacting with a Starknet smart contract with Remix


include::ROOT:partial\$snippet-note-content-from-sn-book.adoc[]

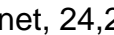
To compile, deploy, and interact with our smart contract, use the Starknet Remix plugin. This tool enables you to start to develop for Starknet without the need for any local installations on your machine.

== Setting up your environment in Remix

. Go to the <https://remix.ethereum.org/#activate=Starknet&lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.24+commit.e11b9ed9.js> [Remix IDE website with the Starknet plugin activated].

// . If Remix requests access to your file system, accept the request.

. Click the **File Explorer**  [File Explorer, 24, 24] tab to review the details of the example project.

. Click the **Starknet**  [Starknet, 24, 24] tab, and click the **Settings**. Select the latest version of Cairo available in Remix.

. In the File explorer, open the ``Scarb.toml`` file to verify the version of your project. Ensure it matches the version specified originally, and correct any discrepancies, if necessary.

== Customizing your environment for the ``Ownable`` contract

[NOTE]

====

For this tutorial, a default example project is provided. Modify or remove certain files and directories as needed.

====

. Rename the root directory to ``ownable``. In your ``Scarb.toml``, under the ``[package]`` section, set the name to ``ownable``.

. Under ``src/``, delete the ``balance.cairo`` and ``forty_two.cairo`` files, if they exist.

. Open ``lib.cairo`` and delete all its contents.

== Exploring the ``ownable`` Starknet smart contract

Explore the [xref:#example-cairo-contract](#) [Example ``Ownable`` contract], crafted in Cairo for Starknet. It includes:

- * An ownership system.
- * A method to transfer ownership.
- * A method to check the current owner.
- * An event notification for ownership changes.

[#example-cairo-contract]

=== Example: ``ownable`` Cairo contract

[source,cairo]

```

----
use starknet::ContractAddress; <1>

#[starknet::interface]
trait OwnableTrait<T> { <1>
    fn transfer_ownership(ref self: T, new_owner: ContractAddress); <2>
    fn get_owner(self: @T) -> ContractAddress; <2>
}

#[starknet::contract]
mod Ownable {
    use super::ContractAddress;
    use starknet::get_caller_address;

    #[event]
    #[derive(Drop, starknet::Event)]
    enum Event {
        OwnershipTransferred: OwnershipTransferred, <3>
    }

    #[derive(Drop, starknet::Event)]
    struct OwnershipTransferred { <3>
        #[key]
        prev_owner: ContractAddress,
        #[key]
        new_owner: ContractAddress,
    }

    #[storage]
    struct Storage { <4>
        owner: ContractAddress,
    }

    #[constructor]
    fn constructor(ref self: ContractState, init_owner: ContractAddress) { <5>
        self.owner.write(init_owner);
    }

    #[abi(embed_v0)]
    impl OwnableImpl of super::OwnableTrait<ContractState> {
        fn transfer_ownership(ref self: ContractState, new_owner: ContractAddress) {
            self.only_owner();
            let prev_owner = self.owner.read();
            self.owner.write(new_owner);
            self.emit(Event::OwnershipTransferred(OwnershipTransferred {
                prev_owner: prev_owner,
            }));
        }
    }
}

```

```

        new_owner: new_owner,
    }));
}

fn get_owner(self: @ContractState) -> ContractAddress {
    self.owner.read()
}

}

#[generate_trait]
impl PrivateMethods of PrivateMethodsTrait {
    fn only_owner(self: @ContractState) { <6>
        let caller = get_caller_address();
        assert(caller == self.owner.read(), 'Caller is not the owner');
    }
}
}
----

```

=== Contract component breakdown

[horizontal,labelwidth="25",role="stripes-odd"]

<1> Dependencies and Interface: +

`starknet::ContractAddress`: Represents a Starknet contract address.

`OwnableTrait`: Defines essential functions for transferring and retrieving ownership.

<2> External Functions: Contains functions for transferring ownership and fetching information about the current owner.

<3> Events: `OwnershipTransferred` indicates changes in ownership, providing details about the previous and new owners.

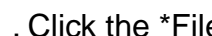
<4> Storage: Stores the contract's state, including the address of the current owner.

<5> Constructor: Sets up the contract by assigning an initial owner.

<6> Private Methods: `only_owner` validates if the caller is the current owner.

== Compiling the contract

To compile using Remix:

. Click the **File Explorer** [File Explorer,24,24] tab, open the `lib.cairo` file and insert the code from [xref:#example-cairo-contract\[\]](#).

. Click the **Starknet** [Starknet, 24,24] tab,

then click `*Home*`.

. Under `*(1) Compile*`, click `*Compile lib.cairo*`.

. Grant the necessary permissions when prompted. Select `*Remember my choice*` for a smoother compilation process in the future.

The compilation process creates an ``artifacts`` directory containing the compiled contract in two formats: a Sierra file, in JSON format, and a Casm file. For Starknet deployment, Remix uses the Sierra file.

== Deploying your contract on the development network (devnet)

Deploying a smart contract in Starknet requires two high-level steps:

. Declare the class of your contract, that is, send your contract's code to the network.

+

When you declare the contract class, you establish an initial owner by calling the class's ``constructor`` function.

. Deploy an instance of the contract class.

This tutorial uses a development network (devnet) to deploy your smart contract. A devnet is a Starknet instance that you run as a local node. A devnet enables much quicker development than is possible using testnet, as well as providing privacy prior to launching on testnet.

.Declaring the contract class

. Select the network by clicking the `*Starknet*` tab, and then clicking the `*Remote Devnet*` menu.

. Under `*Devnet account selection*`, open the menu to view a list of accounts specific to the designated devnet.

. Select a devnet account from the list and note its address for later use.

. Click the `*Declare `lib.cairo`*` button.

+

Remix's terminal provides various logs with important details such as:

+

--

`* `transaction_hash`:` This unique hash identifies the transaction and can be used to track its status.

`* `class_hash`:` Similar to an identifier, the class hash contains the definition of the smart contract.

--

+

.Remix terminal output after declaring the ``ownable`` contract
[source,json]

----- Declaring contract: lib.cairo -----

```
{
  "transaction_hash":
  "0x36dabf43f4962c97cf67ba132fb520091f268e7e33477d77d01747eeb0d7b43",
  "class_hash":
  "0x540779cd109ad20f46cb36d8de1ce30c75469862b4dc75f2f29d1b4d1454f60"
}
```

----- End Declaring contract: lib.cairo -----

After Remix declares the contract class, the **Declare** button says **Declared lib.cairo**.

Now you're ready to deploy an instance of the contract class.

.Deploying a contract instance

. Paste the Devnet account address you used into the ``init_owner`` variable.

+

image:quick-start:init_owner_field.png[]

. Click **Deploy**.

After deployment, Remix's terminal displays various logs, including a transaction receipt, containing important details, such as:

* ``transaction_hash``: This unique hash identifies the transaction and can be used to track its status.

* ``contract_address``: The address of the deployed contract. You can use this address to interact with your contract.

* ``data``: Contains the ``init_owner`` address provided to the constructor.

.Remix terminal output after deploying an instance of the ``Ownable`` class in ``lib.cairo``

[source,bash]

----- Deploying contract: lib.cairo -----

```
{
  "transaction_hash":
  "0x624f5b9f57e53f6b5b62e588f0f949442172b3ad5d04f0827928b4d12c2fa58",
  "contract_address": [
    "0x699952dc736661d0ed573cd2b0956c80a1602169e034fdaa3515bfbc36d6410"
  ]
  ...
  "data": [
    "0x6b0ee6f418e47408cf56c6f98261c1c5693276943be12db9597b933d363df",
    ...
  ]
}
```



```

    ...
}
----- End Deploying contract: lib.cairo -----
----
```

== Interacting with the contract

Now that the contract is operational on the development network, you can start interacting with it on the **Starknet** tab, under **(3) Interact**.

=== Identifying the owner of the contract instance

** Under *Read* you should see the ``get_owner()`` function. Click the **Call** button. The function doesn't require any arguments so the calldata field remains empty. This function reads data, so its invocation is referred to as a `_call_`.*

The terminal displays the output, showing the owner's address, which you provided during the contract's deployment within the calldata for the constructor:

```

[source,json]
-----
----- Calling get_owner -----
{
  "resp": {
    "result": [
      "0x6b0ee6f418e47408cf56c6f98261c1c5693276943be12db9597b933d363df"
    ]
  },
  "contract": "lib.cairo",
  "function": "get_owner"
}
----- End calling get_owner -----
----
```

This call doesn't consume gas because the function doesn't modify the contract's state.

=== Transferring ownership of the contract instance

. Under **(3) Interact**, select **Write**, where functions that alter the contract's state are listed.

. Select the ``transfer_ownership()`` function, which requires providing the new owner address as input.

. Fill in the ``new_owner`` field with any Devnet address other than the one you used to deploy the contract.

+

[TIP]

====

Under **Devnet account selection**, open the menu, select a Devnet account from the list, and copy its address.

====

. Click the **Call** button. The terminal displays the transaction hash indicating the change in the contract's state. Because this interaction is an ``INVOKE`` transaction, and it modifies the contract's state. An ``INVOKE`` transaction requires the signature of the account executing the function.

+

For ``INVOKE`` transactions, the terminal logs include a ``finality_status`` parameter indicating the outcome. A status of ``ACCEPTED_ON_L2`` indicates approval by the Sequencer, the entity responsible for receiving and processing transactions, indicating inclusion in an upcoming block. Conversely, a ``REJECTED`` status indicates that the Sequencer did not approve the transaction, preventing its inclusion in the next block. Typically, transactions of this nature are approved, resulting in a modification of the contract's state.

[source,json]

----- Invoke transfer_ownership transaction receipt -----

```
{
  "resp": {
    "transaction_hash":
"0x5495d56633745aa3b97bdb89c255d522e98fd2cb481974efe898560839aa472"
  },
  "contract": "lib.cairo",
  "function": "get_owner"
}
```

-----End Invoke transfer_ownership transaction receipt -----

== Deployment on Starknet testnet

After testing your smart contract on a development network, the next step is deploying it onto the Starknet testnet. The Starknet testnet is a public platform accessible to all, providing an excellent environment for testing smart contracts and encouraging collaboration among developers.

=== Setting up a smart wallet and a Starknet account on testnet

Before deploying your smart contract on Starknet, it's crucial to address transaction costs. While deploying on the Starknet testnet is free, having an operational smart wallet account is essential. You can set up a smart wallet and a Starknet account using either of the following platforms:

- * <https://www.argent.xyz/argent-x/>[Argent]
- * <https://braavos.app/>[Braavos]

Both options offer robust Starknet wallets with advanced security measures and enhanced accessibility features enabled by the capabilities of the Cairo VM.

.Here's how to set up your smart wallet:

- . Install the recommended browser extension corresponding to your chosen wallet.
- . Follow the instructions provided by your wallet provider to deploy your account on testnet.
- . Use the <https://starknet-faucet.vercel.app/>[Starknet Faucet] to fund your account.
- . Execute the deployment of your account onto the network, typically completed within approximately 10 seconds.

Once the setup is complete, you are primed to deploy your smart contracts onto the Starknet testnet.

=== Deployment and Interaction

- . Proceed as per the aforementioned deployment steps.
- . Within the **Environment selection** tab, Select **Wallet**.
- . Select your Starknet account and proceed with the deployment and interaction processes for your contract.

You can monitor transaction hashes and addresses by using various Starknet block explorers such as:

- * <https://testnet.starkscan.co/>[Starkscan]
- * <https://sepolia.voyager.online/>[Voyager]
- * <https://viewblock.io/starknet/>[ViewBlock]
- * <https://www.oklink.com/starknet/>[Oklink]

These block explorers offer a graphical depiction of transactions and modifications to the contract state. Noteworthy is the visibility provided when altering contract ownership through the `transfer_ownership()` function, as the emitted event by the contract becomes observable within the block explorer. This mechanism serves as a potent means to monitor contractual events.

include::1.0.0@docs-common-content:ROOT:partial\$partial_environment_setup.adoc[]
[id="interacting_with_a_smart_contract_with_starkli"]
= Interacting with a smart contract with Starkli

== Prerequisites

=== Ensure Starkli and Scarb are installed correctly

Ensure that the below commands are working properly on your system.

```
[source, bash]
```

```
----
```

```
starkli --version
```

```
scarb --version
```

```
----
```

If either of the above commands fail, see [xref:environment-setup.adoc\[Setting up your environment\]](#).

== Introduction

Starkli enables interaction with smart contracts via two primary methods:

- * ``call`` for read-only functions.

- * ``invoke`` for write functions that modify the state.

== Calling a function

The ``call`` command allows querying a smart contract function without sending a transaction.

As an example you can use the ``get_owner`` function which doesn't expect any arguments and returns the address of the current owner:

```
[source,bash]
```

```
----
```

```
starkli call \
```

```
    0x014825acb37c36563d3b96c450afe363d2fdfa3cfbd618b323f95b68b55ebf7e \
    get_owner --network=sepolia
```

```
----
```

This will return the address that we passed to the constructor during deployment:

```
[source,bash]
```

```
----
```

```
[
```

```
    "0x02cdab749380950e7a7c0deff5ea8edd716feb3a2952add4e5659655077b8510"
```

```
]
```

```
----
```

== Invoking a function

To modify the state of the smart contract, use the ``invoke`` command. Unlike the ``call`` command, ``invoke`` submits a transaction to the network.

In this example, we'll invoke the `transfer_ownership` function to transfer the ownership from our deployer address to a different smart wallet address:

```
[source,bash]
----
starkli invoke \
  0x014825acb37c36563d3b96c450afe363d2fd3a3cfbd618b323f95b68b55ebf7e \
  transfer_ownership \
  0x011088d3cbe4289bc6750ee3a9cf35e52f4fa4e0ac9f42fb0b62e983139e135a \
  --network=sepolia
----
```

After the transaction is accepted on L2, you can confirm the state transition by calling the `get_owner` function again:

```
[source,bash]
----
starkli call \
  0x014825acb37c36563d3b96c450afe363d2fd3a3cfbd618b323f95b68b55ebf7e \
  get_owner \
  --network=sepolia
----
```

The `get_owner` function now returns the new owner address, confirming the successful ownership transfer.

= Setting up an account

[TIP]

=====

This guide shows you how to set up a Starknet account and wallet in the context of a smart contract deployment.

For information on creating a Starknet wallet as an end user, see link:<https://www.starknet.io/en/posts/stark-math/getting-started-using-starknet-setting-up-a-starknet-wallet>[_Getting Started Using Starknet: Setting Up a Starknet Wallet_].

=====

== Prerequisites

* Starkli is installed correctly. Ensure that the following command shows the version information for Starkli:

```
+
[source, bash]
----
starkli --version
----
```

+

If this command fails, see [xref:environment-setup.adoc\[Setting up your environment\]](#).

* A Starknet wallet is installed, either Argent X or Braavos. For information on a specific wallet, including installation instructions, see that wallet's site.

== Creating an account

A smart wallet is composed of two parts:

* A Signer: A smart contract that can sign transactions.

* An Account Descriptor: A json file that contains information about the smart wallet, such as its address and public key.

After creating and funding your smart wallet with ETH you can deploy it to Starknet. For demonstration purposes, this page uses Starknet's testnet.

For testnet transactions you can fund your wallet using one of the <https://www.starknet.io/en/ecosystem/bridges-and-onramps> [Starknet Sepolia faucets].

=== Creating a Signer

A Signer is a smart contract that can sign transactions. It's a crucial component of accounts in Starknet. To create a Signer you will need the private key of your smart wallet (the public key can be derived from it).

Starkli has the ability to create a keystore file that securely stores the private key of smart wallets each with a password. The accounts in the keystore file can be used to sign transactions using Starkli. The main advantage of this approach is that it prevents storing the private key as plain text on your computer. Instead, a password is used to create an encrypted file in a location of choice.

Normally, the keystore file is stored in the default location of the Starkli CLI.

==== Creating a keystore file

The following command creates a keystore file for a smart wallet in the default location in `~/starkli-wallets/deployer`:

Create a new directory:

```
[source,shell]
```

```
----
```

```
mkdir -p ~/starkli-wallets/deployer
```

```
----
```

Create a keystore file within the directory:

```
[source,shell]
```

```
----
```

```
starkli signer keystore from-key ~/.starkli-wallets/deployer/keystore.json
```

Enter private key:

Enter password:

Created new encrypted keystore file: /home/parallels/.starkli-wallets/deployer/keystore.json

Public key: 0x0550...

```
----
```

[NOTE]

```
=====
```

In the private key prompt, paste the private key of your smart wallet.

In the password prompt, enter a password of your choice.

You will need this password to sign transactions using Starkli.

```
=====
```

```
===== Export the private key from your wallet
```

Next export the private key from your Argent X or Braavos wallet:

```
===== Argent X
```

Navigate to: `Settings` section -> Select your Account -> `Export Private Key`.

```
===== Braavos
```

Navigate to: `Settings` section -> `Privacy and Security` -> `Export Private Key`.

While knowing the private key of a smart wallet is necessary to sign transactions, it's not sufficient. We also need to inform Starkli about the signing mechanism employed by our smart wallet created by Argent X or Braavos.

```
=== Creating an Account Descriptor
```

Starkli offers a command to collect all the required information from a smart wallet by providing its onchain address. Using this data, the CLI generates a json file that can be used to sign transactions:

```
[source,shell]
```

```
----
```

```
starkli account fetch --help
```

Fetch account config from an already deployed account contract

The `fetch` command supports both Argent X and Braavos smart wallets. Make sure your wallet address is already deployed and enter the following command to create and save the account descriptor file:

[source,shell]

```
starkli account fetch <SMART_WALLET_ADDRESS> --output ~/.starkli-wallets/
deployer/account.json --rpc <YOUR_RPC_ENDPOINT_HERE>
```

You can obtain access to a JSON-RPC endpoint in one of the following ways:

- Host your own node with Pathfinder, Juno, Deoxys, or Papyrus.
- Use a third-party JSON-RPC API provider. For information on providers, see [xref:tools:api-services.adoc\[Full nodes and API services\]](#).

The following command shows the details of the account descriptor:

[source,shell]

```
cat ~/.starkli-wallets/deployer/account.json
```

The account descriptor should have the following structure:

[source,json]

```
{
  "version": 1,
  "variant": {
    "type": "argent",
    "version": 1,
    "implementation": "<ARGENT_CLASS_HASH>",
    "signer": "<SMART_WALLET_PUBLIC_KEY>",
    "guardian": "0x0"
  },
  "deployment": {
    "status": "deployed",
    "class_hash": "<SMART_WALLET_CLASS_HASH>",
    "address": "<SMART_WALLET_ADDRESS>"
  }
}
```

[NOTE]

====

If you are working with Braavos wallet, the type is defined as `braavos` and the account descriptor structure might be slightly different.

====

== Deploying an account

Once you have an account file, you can deploy the account contract with the `starkli account deploy` command.

This command sends a `DEPLOY_ACCOUNT` transaction, which requires the account to contain enough ETH to pay for the transaction fee.

To deploy your account, run the following command:

[source,bash]

```
starkli account deploy ~/.starkli-wallets/deployer/account.json
```

[NOTE]

====

This command requires a signer. If you receive an error after running this command, ensure you have the `STARKNET_KEYSTORE` environment variable set as per [xref:environment-setup.adoc#setting_up_starkli_environment_variables](#)[these instructions].

====

When run, the command shows:

- * The address where the contract will be deployed.
- * Instructions for the user to fund the account before proceeding.

Here's an example command output:

[source,bash]

The estimated account deployment fee is 0.000011483579723913 ETH. However, to avoid failure, fund at least:

0.000017225369585869 ETH

to the following address:

0x01cf4d57ba01109f018dec3ea079a38fc08b789e03de4df937ddb9e8a0ff853a

Press [ENTER] once youve funded the address.

You have now successfully deployed a new account to Starknet.

[id="using_starknet_devnet]

= Using a development network

For a faster and more private development process, it is often preferable to use a local version of Starknet – also known as a development network (devnet) – which can be easily set up by either link:<https://0xspaceshard.github.io/starknet-devnet-rs/>[Starknet Devnet] or link:<https://book.dojoengine.org/toolchain/katana>[Katana].

== Prerequisites

Starkli is installed correctly. Ensure that the following command shows the version information for Starkli:

[source, bash]

starkli --version

If this command fails, see xref:environment-setup.adoc[Setting up your environment].

== Using Starknet Devnet (Devnet)

. Install Devnet:

+

[source, shell]

cargo install starknet-devnet

+

and start it using:

+

[source, shell]

starknet-devnet --seed 42

+

[NOTE]

=====

Upon initialization, Devnet predeploys a fee token, universal deployer, and a set of funded accounts. By default, the set of predeployed accounts changes on each initialization, but specifying the same `--seed` value ensures consistent account addresses between executions.

=====

The result should be similar to the following:

+

```
[source,bash]
```

```
----
```

```
Predeployed FeeToken
```

```
ETH Address:
```

```
0x49D36570D4E46F48E99674BD3FCC84644DDD6B96F7C741B1562B82F9E004DC7
```

```
STRK Address:
```

```
0x04718f5a0fc34cc1af16a1cdee98ffb20c31f5cd61d6ab07201858f4287c938d
```

```
Class Hash:
```

```
0x046ded64ae2dead6448e247234bab192a9c483644395b66f2155f2614e5804b0
```

```
Predeployed UDC
```

```
Address:
```

```
0x41A78E741E5AF2FEC34B695679BC6891742439F7AFB8484ECD7766661AD02BF
```

```
Class Hash:
```

```
0x7B3E05F48F0C69E4A65CE5E076A66271A527AFF2C34CE1083EC6E1526997A69
```

```
Chain ID: SN_SEPOLIA (0x534e5f5345504f4c4941)
```

```
| Account address |
```

```
0x34ba56f92265f0868c57d3fe72ecab144fc96f97954bbbc4252cef8e8a979ba
```

```
| Private key      | 0xb137668388dbe9acdfa3bc734cc2c469
```

```
| Public key       |
```

```
0x5a5e37c60e77a0318643b111f88413a76af6233c891a0cfb2804106372006d4
```

```
...
```

```
----
```

```
+
```

```
. xref:set-up-an-account.adoc[As previously described], import Devnet's first  
predeployed accounts to an account file (notice that the address of the fetched account  
is the same as the one printed out by Devnet):
```

```
+
```

```
[source,bash]
```

```
----
```

```
starkli account fetch --rpc http://127.0.0.1:5050
```

```
0x34ba56f92265f0868c57d3fe72ecab144fc96f97954bbbc4252cef8e8a979ba --output  
~/.starkli-wallets/devnet/account.json
```

```
----
```

```
+
```

```
[NOTE]
```

```
=====
```

```
`http://127.0.0.1:5050` are Devnet's default host and port, which can be configured  
upon initialization using the `--host` and `--port` options
```

```
=====
```

```
and create the corresponding keystore file by executing:
```

```
+
```

```
[source,bash]
```

```
----
starkli signer keystore from-key ~/.starkli-wallets/devnet/keystore.json
----
```

+
and entering `0xb137668388dbe9acdfa3bc734cc2c469` as private key (the same one as the one printed out by Devnet)
. Now that you have an account set up, you can use Starkli to freely interact with Devnet. For example, you can redeploy Devnet's predeployed universal deployer using the following command:

```
[source,bash]
----
starkli deploy --rpc http://127.0.0.1:5050 --account ~/.starkli-wallets/devnet/account.json
--keystore ~/.starkli-wallets/devnet/keystore.json
0x7B3E05F48F0C69E4A65CE5E076A66271A527AFF2C34CE1083EC6E1526997A69
----
```

== Using Katana

. Install Katana:

+
[source,bash]

git clone https://github.com/dojoengine/dojo
cd dojo
cargo install --path ./bin/katana --locked --force

and start it using:

+
[source,bash]

katana

+
[NOTE]

====
Upon initialization, Katana predeploys a fee token, universal deployer contract (UDC), and a set of funded accounts. By default, Katana predeploys the same set of accounts.

====
The result should be similar to the following:

+
[source,bash]

PREDEPLOYED CONTRACTS
=====

Contract	Fee Token
----------	-----------

```

| Address      |
0x49d36570d4e46f48e99674bd3fcc84644ddd6b96f7c741b1562b82f9e004dc7
| Class Hash   |
0x02a8846878b6ad1f54f6ba46f5f40e11cee755c677f130b2c4b60566c9003f1f

| Contract     | Universal Deployer
| Address      |
0x41a78e741e5af2fec34b695679bc6891742439f7afb8484ecd7766661ad02bf
| Class Hash   |
0x07b3e05f48f0c69e4a65ce5e076a66271a527aff2c34ce1083ec6e1526997a69

| Contract     | Account Contract
| Class Hash   |
0x05400e90f7e0ae78bd02c77cd75527280470e2fe19c54970dd79dc37a9d3645c

```

PREFUNDED ACCOUNTS

=====

```

| Account address |
0xb3ff441a68610b30fd5e2abbbf3a1548eb6ba6f3559f2862bf2dc757e5828ca
| Private key     |
0x2bbf4f9fd0bbb2e60b0316c1fe0b76cf7a4d0198bd493ced9b8df2a3a24d68a
| Public key      |
0x640466ebd2ce505209d3e5c4494b4276ed8f1cde764d757eb48831961f7cdea

```

...

+

. Starkli comes with several built-in accounts for Katana's default initialization, which can be used to freely interact with Katana without any setup (for the full list of account addresses, see `BUILTIN_ACCOUNT` in link:<https://github.com/xJonathanLEI/starkli/blob/master/src/account.rs> [Starkli's accounts.rs file]). For example, you can redeploy Katana's predeployed universal deployer using the following command:

+

```
[source,bash]
```

```
starkli deploy --rpc http://0.0.0.0:5050 --account katana
0x07b3e05f48f0c69e4a65ce5e076a66271a527aff2c34ce1083ec6e1526997a69
```

+

[NOTE]

====

`http://0.0.0.0:5050` are Katana's default host and port, which can be configured upon initialization using the `--host` and `--port` options

====[id="staking_architecture"]

= Staking Architecture (WIP) Ø=Þ§

Staking on Starknet is designed to enhance network security and decentralization by allowing users to stake their STRK tokens directly or delegate them to other validators. The architecture is modular, with different contracts handling specific responsibilities to ensure flexibility, security, and ease of upgrades. For more details on the staking architecture, visit the following link: <https://github.com/starkware-libs/starknet-staking>[Starknet Staking Repository].

== Components of the Staking Architecture

Staking is divided into several key components, each responsible for different aspects of the staking process.

The following contracts handle the core functionalities of the staking system:

.Contracts of the Staking Architecture

[cols='1,8']

|===

| Contract

| Description

| Staking Contract

| The central contract that manages the staking process. It handles direct staking, rewards distribution, and interactions with delegation pools. Validators interact with this contract to stake their tokens, claim rewards, and initiate unstaking.

| Delegation Pooling Contract

| This contract manages the delegation process, allowing delegators to assign their tokens to a validator for staking. The contract is responsible for tracking each delegator's share, calculating their rewards, and managing the delegation and unstaking processes.

| Reward Supplier Contract

| A dedicated contract that calculates and supplies the rewards for validators and delegators. It interfaces with the staking contract to distribute rewards based on the defined minting curve.

| Minting Curve Contract

| A contract responsible for implementing the minting curve logic that governs the reward distribution mechanism. It adjusts rewards dynamically based on the total staked amount and the overall supply of STRK tokens.

|===

The following data structures are stored within the contracts and play a crucial role in managing validators' and delegators' information within the staking protocol:

.Key Structures in the Staking Architecture

[cols='1,8']

|===

| Structure

| Description

| StakerInfo Structure

| A data structure stored within the staking contract that holds detailed information about each validator, including their staked amount, unclaimed rewards, delegation details, and operational parameters. This structure ensures that validators' information is accurately tracked and updated.

| PoolMemberInfo Structure

| A data structure stored within the delegation pooling contract that holds information about each delegator's contributions, rewards, and status within the pool. This structure helps manage and calculate the delegation and reward distribution processes for pool members.

|===

For more technical details, you can refer to the full staking specification document available in: <https://github.com/starkware-libs/starknet-staking/blob/main/docs/spec.md>[Staking Repository Spec].

== Staking Contract

The staking contract is the core of the staking system. It manages the lifecycle of validators, from the initial stake to claiming rewards and unstaking. The contract ensures that the validator's tokens are securely locked and that rewards are distributed according to the minting curve.

***Key Functions and Responsibilities*:** +

`stake`: Allows users to stake their STRK tokens directly into the contract. The function initializes the validator's information, locks the tokens, and optionally deploys a delegation pool if pooling is enabled. +

`increase_stake`: Lets existing validators add more tokens to their stake. The contract recalculates rewards and updates the validator's information accordingly. +

`unstake_intent`: Initiates the unstaking process by locking the validator's tokens for a specified exit period. No rewards are earned during this period. +

`unstake_action`: Finalizes the unstaking process after the exit period has passed, releasing the staked tokens back to the validator. +

`claim_rewards`: Calculates and transfers the validator's accumulated rewards to their designated reward address. +

== Delegation Pooling Contract

The delegation pooling contract enables users to delegate their tokens to a validator

without having to manage the staking process themselves. This contract tracks each delegator's contribution, calculates their rewards, and manages the delegation lifecycle.

***Key Functions and Responsibilities*:** +

``enter_delegation_pool``: Allows users to delegate their tokens to the pool associated with a validator. This function transfers the tokens, updates the delegator's record, and integrates them into the validator's pool. +

``add_to_delegation_pool``: Enables existing delegators to increase their delegation amount. The contract updates the pool's total and recalculates the member's rewards. +

``exit_delegation_pool_intent``: Initiates the process for a delegator to exit the pool. Similar to validators, the delegator's funds are locked for a period before they can be withdrawn. +

``exit_delegation_pool_action``: Finalizes the exit process for a delegator, returning their tokens and any unclaimed rewards. +

``switch_delegation_pool``: Allows a delegator to transfer their delegated stake from one validator's pool to another, facilitating dynamic delegation strategies. +

``claim_rewards``: Transfers the delegator's earned rewards to their specified reward address. +

== Reward Supplier Contract

The reward supplier contract is responsible for calculating and supplying the staking rewards based on the minting curve. It ensures that the rewards are distributed fairly and in accordance with the protocol's economic parameters.

***Key Functions and Responsibilities*:** +

``calculate_staking_rewards``: Computes the rewards based on the current staking rate and the minting curve, updating the staking contract with the amount to be distributed. +

``claim_rewards``: Handles the transfer of rewards to the staking contract, ensuring that the correct amount is distributed to validators and delegators. +

== Minting Curve Contract

The minting curve contract defines the economic model that governs reward distribution. It ensures that the network's inflation is managed while incentivizing participation in staking.

***Key Functions and Responsibilities*:** +

``yearly_mint``: Returns the amount of STRK tokens to be minted annually based on the current staking rate. This function uses a square root formula to balance rewards and inflation. +

``update_total_supply``: Updates the total supply of STRK tokens, ensuring that the minting calculations remain accurate.

== Security and Upgradability

The staking architecture on Starknet is designed with security and upgradability in mind. Each contract is modular, allowing for targeted upgrades and improvements without affecting the entire system. Access control mechanisms are in place to ensure that only authorized parties can make critical changes, further enhancing the security of the staking process.[id="claiming-rewards"]

= Claiming Rewards (WIP) Ø=Þ§

:description: How to claim your staking rewards on Starknet by directly interacting with the staking or delegation pooling contracts.

Staking rewards on Starknet accumulate over time as your staked STRK tokens contribute to network security. To claim these rewards, you need to interact with the staking contract (if you are a validator) or the specific delegation pooling contract associated with the validator you have staked with (if you are a delegator) and execute the appropriate functions.

.Prerequisites

- * The validator or delegator address must have an existing stake in the respective contract.
- * The caller must be either the validator/delegator or their respective reward address.

.Procedure

. Using a Starknet block explorer, navigate to the appropriate contract:

+

- * For validators: Navigate to the staking contract.
- * For delegators: Navigate to the delegation pooling contract associated with the validator you have staked with.

. In the contract interface, locate and select the `claim_rewards` function.

. Enter the following parameters:

+

- * In *`staker_address`* or *`pool_member`*, enter the address of the validator or delegator for whom you are claiming the rewards.

. Submit the transaction to initiate the rewards claim.

[NOTE]

====

Ensure that the address entered is either the validator's or delegator's main address or their respective reward address. If the address is not authorized, the function will fail.

====

Once the transaction is successful, the rewards will be transferred to the designated reward address associated with the staked tokens.[id="delegating-stake"]

= Delegating Stake (WIP) Ø=Þ§

:description: How to delegate your stake to a validator on Starknet by interacting directly with the staking and delegation pooling contracts.

Delegating your stake on Starknet involves adding your stake to a validator's delegation pool managed by the staking contract. This approach offers the advantage of lower capital requirements and relieves you of the need to manage the operational aspects of staking, as the validator handles these responsibilities.

The delegation process includes joining a validator's delegation pool by interacting with the `enter_delegation_pool` function.

.Procedure

- . Using a Starknet block explorer, navigate to the delegation pooling contract associated with the validator you want to delegate to.
- . In the contract interface, locate and select the `enter_delegation_pool` function.
- . Enter the following parameters:
 - +
 - * In `amount`, enter the number of STRK tokens you want to delegate to the validator.
 - * In `reward_address`, enter the address where you wish to receive your staking rewards.
- . Submit the transaction to join the delegation pool.

Upon successful execution, your STRK tokens will be added to the validator's delegation pool, and you will begin earning rewards as part of the pooled staking process.

[id="entering-staking"]

= Becoming a Validator (WIP) Ø=Þ§

:description: How to enter the staking protocol on Starknet by interacting directly with the staking contract.

Using the Starknet staking contract to stake STRK tokens requires interacting with the `stake` function. The `stake` function does the following:

- . Locks the specified amount of STRK tokens from the validator's account into the staking contract.
- . Records the validator's details, including reward and operational addresses, in the staking contract.
- . If pooling is enabled, deploys a new delegation pool contract associated with the validator.

Subsequently, the validator's tokens will be locked in the staking contract, and the validator will begin earning rewards based on their stake.

For more information on what happens during the staking process, see

xref:architecture.adoc#staking-contract[Staking Contract Architecture].

.Prerequisites

- * A Starknet-compatible block explorer or CLI tool.
- * Sufficient STRK token balance in your wallet.
- * Pre-approval of the STRK ERC20 contract on Starknet for the transfer of tokens from your address to the staking contract.

.Procedure

- . Using a Starknet block explorer, navigate to the staking contract.
- . In the contract interface, locate and select the ``stake`` function.
- . Enter the following parameters:
 - +
 - * In ``reward_address``, enter the address where the rewards will be sent.
 - * In ``operational_address``, enter the operational address associated with this stake.
 - * In ``amount``, enter the number of STRK tokens you want to stake.
 - * In ``pooling_enabled``, enter ``true`` if you wish to enable delegation pooling, otherwise enter ``false``.
 - * In ``commission``, enter the commission rate for any delegated staking. The rate should be entered as a percentage with precision, where 10000 represents 100%. For example, to set a 5% commission, you would enter 500.
 - . Submit the transaction to execute the staking operation.[id="exiting-staking"]
- = Exiting the Staking Protocol (WIP) Ø=Þ§

:description: How to exit the staking protocol on Starknet by unstaking as a validator or undelegating as a delegator through direct interaction with the staking or delegation pooling contracts.

Exiting the staking protocol involves either unstaking your STRK tokens as a validator or undelegating your stake as a delegator. Both processes require you to first signal your intent to exit, followed by an action to finalize the process after a waiting period.

== Unstaking as a Validator

Validators can unstake their STRK tokens, which involves pausing rewards and exiting the staking contract.

.Prerequisites

- * The validator must not currently be in the unstake process.

.Procedure

1. ****Signal Unstake Intent:****

- . Using a Starknet block explorer, navigate to the staking contract.
- . In the contract interface, locate and select the ``unstake_intent`` function.
- . Submit the transaction to initiate the unstake process. This will record the unstake intent, pause rewards collection, and set a waiting period.

2. ****Finalize Unstake:****

- . After the waiting period has passed, return to the staking contract.
- . In the contract interface, locate and select the ``unstake_action`` function.
- . Enter the following parameters:
 - +
 - * In ``staker_address``, enter the validator's address.
- . Submit the transaction to finalize the unstaking process and transfer the staked STRK tokens back to the validator's account.

[NOTE]

====

Any address can initiate the ``unstake_action`` function once the waiting period has passed.

====

[NOTE]

====

Ensure that enough time has passed since signaling your unstake intent. If the waiting period has not expired, the ``unstake_action`` function will fail.

====

== Undelegating as a Delegator

Delegators can undelegate their stake from a validator's delegation pool by following a similar process.

.Prerequisites

- * A Starknet-compatible block explorer or CLI tool.
- * The delegator must not currently be in the undelegation process.
- * The contract address of the delegation pooling contract. `__` ******(TODO: Check if this really needs to be mentioned.) ****** `__`

.Procedure

1. ****Signal Undelegation Intent:****

- . Using a Starknet block explorer, navigate to the delegation pooling contract.
- . In the contract interface, locate and select the ``exit_delegation_pool_intent`` function.
- . Submit the transaction to initiate the undelegation process. This will record the undelegation intent, pause rewards collection, and set a waiting period.

2. ****Finalize Undelegation:****

- . After the waiting period has passed, return to the delegation pooling contract.
- . In the contract interface, locate and select the ``exit_delegation_pool_action`` function.
- . Enter the following parameters:
 - +
 - * In `*`pool_member`*`, enter the delegator's address.
 - . Submit the transaction to finalize the undelegation process and transfer the undelegated STRK tokens back to the delegator's account.

[NOTE]

=====

Any address can initiate the ``exit_delegation_pool_action`` function once the waiting period has passed.

=====

[NOTE]

=====

Ensure that enough time has passed since signaling your undelegation intent. If the waiting period has not expired, the ``exit_delegation_pool_action`` function will fail.

=====

== Additional Notes

- When a validator unstakes, any unclaimed rewards are automatically transferred to the reward address before the stake is returned.
- Similarly, when a delegator undelegates, any unclaimed rewards are automatically transferred to the delegator's reward address.

For more details on the staking and delegation processes, see [xref:architecture.adoc#staking-contract](#) [Staking Contract Architecture].

[id="handling-staking-events"]

= Handling Staking Events (WIP) Ø=Þ§

:description: How to manage and respond to events emitted by the Starknet staking protocol.

Staking events in Starknet provide critical information about changes in the staking process, such as balance updates, delegation pool creation, and exit intents. This section guides you through the process of handling these events and responding to them.[id="increasing-stake"]

= Increasing Stake (WIP) Ø=Þ§

:description: How to increase your stake on Starknet by interacting directly with the staking or delegation pooling contracts.

The staking and pooling contracts allow validators and delegators to increase their

existing stake. Validators use the ``increase_stake`` function, which is called from the staking contract, while delegators use the ``add_to_delegation_pool`` function, which is called from the delegation pooling contract. These functions add the specified amount of STRK tokens to the current stake, recalculate rewards before the staked amount is updated, and update the total staked amount.

.Prerequisites

- * Sufficient STRK token balance in your Starknet wallet.
- * The validator/delegator must have pre-approved the relevant contract (staking contract for validators, delegation pooling contract for delegators) to transfer the specified STRK amount from their account.
- * The validator/delegator must not be in an unstake/undelegate process.
- * The caller must be either the validator/delegator or their respective reward address.

.Procedure

- . Using a Starknet block explorer, navigate to the relevant contract (staking contract for validators, delegation pooling contract for delegators).
- . In the contract interface, locate and select the ``increase_stake`` function (for validators) or the ``add_to_delegation_pool`` function (for delegators).
- . Enter the following parameters:
 - +
 - * In ``address``, enter the address of the validator (for ``increase_stake``) or the delegator (for ``add_to_delegation_pool``) whose stake you want to increase. This can be either the validator's/delegator's address or their respective reward address.
 - * In ``amount``, enter the number of STRK tokens you want to add to the existing stake.
- . Submit the transaction to execute the stake increase.

[NOTE]

====

Ensure that the validator/delegator is not in an unstake/undelegate process before attempting to increase the stake. If they are currently unstaking/undelegating, this function will fail.

====

[id="managing-staking-and-delegation-operations"]

= Managing Staking and Delegation Operations (WIP) Ø=Þ§

:description: How validators and delegators can manage staking and delegation settings on Starknet by interacting directly with the staking and delegation pooling contracts.

This guide outlines how both **validators** and **delegators** can manage various aspects of their staking and delegation activities on Starknet. Validators can perform operations such as opening delegation pools, updating commission rates, and changing operational and reward addresses. Delegators can manage their participation

by changing their reward addresses associated with delegation pools.

== Managing Staking Operations as a Validator

Validators have several functions available to effectively manage their staking and delegation settings.

.Prerequisites

- * An existing stake in the staking contract.
- * An existing delegation pool if updating pool-specific settings.

=== Opening a Delegation Pool

If a validator does not yet have a delegation pool, they can open one by calling the ``set_open_for_delegation`` function.

.Procedure

- . Using a Starknet block explorer, navigate to the **staking contract**.
- . In the contract interface, locate and select the ``set_open_for_delegation`` function.
- . Enter the following parameter:
 - +
 - * ``commission``: Enter the commission rate for the delegation pool, expressed as a percentage with precision (where 10000 represents 100%). For example, to set a 5% commission, enter ``500``.
- . Submit the transaction to create the delegation pool.

Once created, the delegation pool will be associated with the validator's staking contract, allowing delegators to delegate their stake to the validator.

=== Updating Commission Rate

Validators can update the commission rate of their delegation pool using the ``update_commission`` function. **Note:** The commission rate can only be decreased or kept the same; it cannot be increased.

.Procedure

- . Using a Starknet block explorer, navigate to the **staking contract**.
- . In the contract interface, locate and select the ``update_commission`` function.
- . Enter the following parameter:
 - +
 - * ``commission``: Enter the new commission rate, which must be equal to or less than the current rate, expressed as a percentage with precision (where 10000 represents 100%). For example, to set a 5% commission, enter ``500``.

. Submit the transaction to update the commission rate.

=== Changing the Operational Address

Validators can change their operational address by interacting with the ``change_operational_address`` function.

.Procedure

- . Using a Starknet block explorer, navigate to the ****staking contract****.
- . In the contract interface, locate and select the ``change_operational_address`` function.
- . Enter the following parameter:
 - +
 - * ****`operational_address`****: Enter the new operational address.
- . Submit the transaction to update the operational address.

=== Changing the Reward Address

Validators can update the reward address associated with their staking contract using the ``change_reward_address`` function.

.Procedure

- . Using a Starknet block explorer, navigate to the ****staking contract****.
- . In the contract interface, locate and select the ``change_reward_address`` function.
- . Enter the following parameter:
 - +
 - * ****`reward_address`****: Enter the new reward address.
- . Submit the transaction to change the reward address.

== Managing Delegation Settings as a Delegator

=== Changing the Reward Address

Delegators can change their reward address by interacting with the ``change_reward_address`` function in the delegation pooling contract.

.Procedure

- . Using a Starknet block explorer, navigate to the delegation pooling contract associated with your delegation.
- . In the contract interface, locate and select the ``change_reward_address`` function.
- . Enter the following parameter:
 - +
 - * ****`reward_address`****: Enter the new reward address.
- . Submit the transaction to update the reward address.

== Additional Information

For more information on how the staking and delegation pooling systems work, refer to the [xref:architecture.adoc#staking-contract](#)[Staking Contract Architecture].

[id="staking_overview"]

= Staking Overview (WIP) Ø=Þ§

Staking on Starknet is a pivotal component in enhancing the network's security and decentralization. Developed by StarkWare, this staking protocol introduces both direct staking and stake delegation, allowing users to earn rewards while supporting the network.

Staking on Starknet involves locking STRK tokens in the staking protocol, contributing to network security and performance. Users can either stake directly or delegate their tokens to others, with rewards based on their level of participation and contribution.

The key terms used in this document are:

- * **_Stake_**: Locking STRK tokens into the staking protocol.
- * **_Delegate_**: Assigning STRK tokens to a validator to participate indirectly.
- * **_Reward_**: Earnings from participating in staking.

[NOTE]

=====

STRK tokens never leave the Starknet protocol during these operations.

=====

== Staking Protocol Details

=== Overview

The staking protocol features two main options:

- * **_Staking_**: Users can stake any amount of STRK, with a minimum threshold set at 20,000 STRK. Validators are expected to run full nodes and eventually handle additional responsibilities as the protocol evolves.
- * **_Stake Delegation_**: Users can delegate their STRK to validators without running their own nodes. Delegators share in the rewards earned by the validators they choose.

Validators and delegators can both unstake their funds, subject to network-defined latencies for security.

=== Staking Rewards

Rewards are distributed based on the amount staked and the commission policy constant $stem:[CP]$ set by the validator. The rewards are calculated using the following formulas:

```
[stem]
++++
\begin{align}
&\text{Validators: } \& \; \left[ \text{self\_stake} + \text{total\_stake\_delegated} \right] \times \\
&\text{CP} \times \text{rewards\_constant} \times \text{time\_interval} \setminus \\
&\text{Stake Delegates: } \& \; \text{stake\_delegated} \times (1 - CP) \times \\
&\text{rewards\_constant} \times \text{time\_interval} \\
\end{align}
++++
```

Here, $stem:[\text{rewards_constant}]$ is determined by the minting curve, which adjusts rewards based on the total staked amount.

=== Minting Curve

The minting curve balances participation and inflation by adjusting rewards based on the total STRK locked in the protocol. It is defined by the formula:

```
[stem]
++++
M = \frac{C}{10} \times \sqrt{S}
++++
```

Where:

- * $stem:[S]$ is the staking rate as a percentage of the total token supply.
- * $stem:[M]$ is the annual minting rate as a percentage of the total token supply.
- * $stem:[C]$ is the maximum theoretical inflation percentage.

For the first stage, $stem:[C]$ is proposed to be 1.6%.

=== Latencies

- * **Current Version**: Immediate entry and exit from the staking protocol. However, funds are subject to a 21-day security lockup after withdrawal.
- * **Future Versions**: Introduction of epochs to determine entry/exit latencies and continued 21-day lockup after withdrawal.

[NOTE]

=====

Stake delegators can switch between validators without waiting for the full lockup

period, promoting a competitive delegation market.

====

=== Economic Parameters

The proposed economic parameters are:

- * **Minimum STRK for Staking:** 20,000 STRK
- * **Withdrawal Security Lockup:** 21 days
- * **Minting Curve Yearly Inflation Cap (stem:[C]):** 1.6% (see link:[https://community.starknet.io/t/staking-on-starknet-voting-proposal/114442/\[here\]](https://community.starknet.io/t/staking-on-starknet-voting-proposal/114442/[here]) for the relevant voting proposal)
- * **Commission Policy Parameter (stem:[CP]):** Set by the validator (0 - 1)

These values are our proposed starting points for this version of the protocol. As part of the rationale behind this version, they are subject to change and may be adjusted to better suit the protocol's needs under the proper governance procedures.[id="switching-delegation-pools"]

= Switching Delegation Pools (WIP) Ø=Þ§

:description: How to switch your delegated stake from one validator's pool to another on Starknet by interacting directly with the delegation pooling contract.

Switching your delegated stake from one validator's delegation pool to another allows you to optimize your staking strategy by moving your funds to a different validator. This process involves interacting with the `switch_delegation_pool` function in the delegation pooling contract, which coordinates with the staking contract to move your stake.

.Prerequisites

- * An existing delegation in a validator's pool with a STRK token balance equal to or greater than the amount you wish to switch.
- * The delegator must have initiated an undelegation process before attempting to switch pools.
- * The target validator must have an active delegation pool associated with their staking contract.

.Procedure

. Using a Starknet block explorer, navigate to the delegation pooling contract associated with the validator from whose pool you wish to switch.

. In the contract interface, locate and select the `switch_delegation_pool` function.

. Enter the following parameters:

+

* In `*_to_staker`, enter the address of the validator whose pool you want to switch to.

* In `*_pool`, enter the address of the target delegation pooling contract associated

with the new validator.

* In `*`amount`*`, enter the number of STRK tokens you want to move to the new delegation pool.

. Submit the transaction to execute the delegation pool switch.

[NOTE]

====

Ensure that you have initiated the undelegation process before attempting to switch pools. If the undelegation intent has not been set, this function will fail.

====

Upon successfully switching pools, any unclaimed rewards from your previous delegation pool will be transferred to your designated reward address. Your subsequent rewards will be based on the performance of the new validator's delegation pool.

For more details on the staking architecture and how delegation pools work, see [xref:architecture.adoc#staking-contract](#) [Staking Contract Architecture].

[id="StarkGate_adding_a_token"]

= Adding a token to StarkGate

StarkGate supports permissionless bridging.

Want to learn how to enroll a token bridge on StarkGate? Check out this link: <https://research.lazer1.xyz/blog/making-sense-of-starknet-architecture-and-l1-l2-messaging/#enroll-a-token-bridge> [Community Guide].

== Additional resources

* [xref:function-reference.adoc#enrollTokenBridge](#) [enrollTokenBridge] in the `_StarkGate` function reference_

* [xref:architecture.adoc](#) [StarkGate architecture]

[id="StarkGate_architecture"]

= StarkGate architecture

While StarkGate is referred to as `_a bridge_`, technically, each supported token has its own bridge, each of which is defined in a corresponding pair of L1 and L2 contracts, as described in [xref:#bridge_explanation_table](#) [].

.Components of a bridge for an individual token

[#bridge_explanation_table]

[cols='1,8']

|===

| `*L1*`

a|

* ERC-20 contract that defines the token on Ethereum. This contract must exist before you can bridge it.

* Bridge functionality for ERC-20 tokens is supported in `StarknetTokenBridge.sol`.

| *L2*

a|

* StarkGate Cairo bridge contract instance of the `token_bridge.cairo` contract class.

* ERC-20 Cairo token contract instance of a standard ERC-20 Cairo token contract class. Each new contract uses the hash of this class to define its inheritance. StarkGate creates this contract automatically when creating the bridge in StarkGate.

|===

// Need to add the Ethereum addresses of these contracts

include::partial\$snippet_backwards_compatibility_note.adoc[]

You can check if a token is currently supported with the L1 function `getBridge`.

// Ask Dan if he can add a feature that shows all currently supported ERC-20 tokens.

You can permissionlessly add support for ERC-20 tokens to the multi-token bridge using the `enrollTokenBridge` function.

[#StarkGate_manager_and_registry]

== The StarkGate Manager and Registry

StarkGate includes the following administration components:

// [horizontal]

The `_StarkGate Manager_`:: is responsible for adding bridges

The `_StarkGate Registry_`::

* contains the addresses to all supported bridges

* enables a bridge developer to stop servicing an existing bridge

[#legacy_bridge]

== Legacy bridges

All token bridges that existed prior to StarkGate 2.0 (Mainnet: January 2024) besides supporting all StarkGate 2.0 functionality, are backward compatible. Each token was bridged with a unique, custom pair of L1 and L2 bridge contracts. The L1 bridge is an instance of [link:https://github.com/starknet-io/starkgate-contracts/blob/cairo-1/src/solidity/LegacyBridge.sol](https://github.com/starknet-io/starkgate-contracts/blob/cairo-1/src/solidity/LegacyBridge.sol) [LegacyBridge.sol], and the L2 bridge is an instance of [link:https://github.com/starknet-io/starkgate-contracts/blob/cairo-1/src/cairo/legacy_bridge_eic.cairo](https://github.com/starknet-io/starkgate-contracts/blob/cairo-1/src/cairo/legacy_bridge_eic.cairo) [legacy_bridge_eic.cairo].

[#example]

=== Example: USDC contracts

* L1 ERC-20 contract address: `0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48`

* L2 ERC-20 contract address:

`0x053c91253bc9682c04929ca02ed00b3e423f6710d2ee7e0d5ebb06f3ecf368a8`

* L1 bridge contract address: `0xF6080D9fbEEbcd44D89aFfBFd42F098cbFf92816`

* L2 bridge contract address:

`0x05cd48fccbfd8aa2773fe22c217e808319ffcc1c5a6a463f7d8fa2da48218196`

Consider the `deposit` functions for the L1 bridge contract for USDC on Etherscan:

link:<https://etherscan.io/address/0xf6080d9fbEEbcd44d89affbfd42f098cbff92816#writeProxyContract>[`0xf6080d9fbEEbcd44d89affbfd42f098cbff92816#writeProxyContract`]

This contract has the following two `deposit` functions: `deposit (0x0efe6a8b)` and `deposit (0xe2bbb158)`, which shows that the bridge includes support for the legacy functionality as well as the modern functionality:

[horizontal,labelwidth="25"]

`deposit (0x0efe6a8b)`:: The StarkGate 2.0 contract, which includes support for all tokens within a single contract, requiring that you enter the address of the token in the deposit function.

`deposit (0xe2bbb158)`:: The legacy contract, which is labeled *Support Legacy ABI*. This function does not include the `token (address)` parameter, because the contract that contains this function only supports USDC. Therefore, the address of the token is superfluous.

[#legacy_bridges]

=== List of legacy bridges

The following tokens have legacy contracts as well as StarkGate 2.0 contracts:

- * Starknet Token (STRK)
- * Starknet Voting Token (vSTRK)
- * Wrapped BTC (WBTC)
- * USD Coin (USDC)
- * Tether USD (USDT)
- * Ether (ETH)
- * Dai Stablecoin (DAI)
- * Dai Stablecoin (DAI) (Dai v0)
- * Wrapped liquid staked Ether 2.0 (wstETH)
- * Rocket Pool ETH (rETH)
- * R Stablecoin +(R)+
- * Frax (FRAX)
- * Frax Share (FXS)
- * Staked Frax Ether (sfrxETH)
- * LUSD Stablecoin (LUSD)
- * Uniswap (UNI)

For complete details, see the link:<https://github.com/starknet-io/starknet-addresses/tree/>

master/bridged_tokens[`bridged_tokens`] directory on GitHub.

```
[#stark_gate_withdrawal_limit]
== Withdrawal limit
```

By default, StarkGate imposes no limit on withdrawals. However, in order to mitigate risks associated with critical vulnerabilities that could result in the loss of user funds, StarkGate can enable a withdrawal limit.

If a serious security issue arises, the security agent in the StarkGate contract can limit withdrawals to 5% of the Total Value Locked (TVL) per day for any affected token by calling the `setWithdrawLimitPCT()` function in the `WithdrawalLimit.sol` contract. A dedicated team can then investigate and resolve the issue.

Only a security admin quorum can disable the withdrawal limit. The quorum will consist of Starknet Foundation members, Starknet ecosystem contributors, and StarkWare representatives. This diverse group will ensure that decisions reflect the Starknet community's broad interests.

This approach, blending manual oversight with automated detection, aims to minimize potential losses.

== Additional resources

* xref:adding-a-token.adoc[]

* The xref:function-reference.adoc[_StarkGate function reference_]. Lists functions exposed by the Registry, Manager, and the bridge itself, including:

** xref:function-reference.adoc#getBridge[getBridge]

** xref:function-reference.adoc#enrollTokenBridge[enrollTokenBridge]

** L1 contracts on GitHub

** L2 contracts on GitHub

// Awaiting final URL.

// * link:https://github.com/starknet-io/starkgate-contracts/blob/d62a255307d2f3de65665f18316766a2c69ead78/src/solidity/

WithdrawalLimit.sol#L20[setWithdrawLimitPCT] in WithdrawalLimit.sol

[id="performing_a_smart_deposit"]

= Performing a Smart Deposit

A `_Smart Deposit` is a deposit that moves funds from L1 to L2 and then triggers subsequent actions. For example, a user can deposit funds and transfer those funds to another recipient, such as an exchange.

The `depositWithMessage` function enables a Smart Deposit. `depositWithMessage` is similar to the `deposit` function, with an additional 256-bit message, which can contain

instructions for executing additional actions.

Upon completion, the `depositWithMessage` function triggers a call to a callback function, named `on_receive`, on the L2 contract that receives the deposit. The `on_receive` function receives the deposit message as input.

`on_receive` must return `true` for the deposit to succeed. If `on_receive` returns `false`, or if the recipient contract does not include the `on_receive` function, the `depositWithMessage` function's L1 handler fails. The user can recover their funds using the `depositWithMessageCancelRequest` function.

.To enable Smart Deposits in your application:

- . Implement the `on_receive` function in the L2 contract that should receive deposits.
- . Use the `depositWithMessage` function to transfer funds from L1 to L2.

[discrete]

== Additional resources

- * [xref:function-reference.adoc#depositWithMessageCancelRequest](#)[`depositWithMessageCancelRequest`]
- * [xref:function-reference.adoc#depositWithMessage](#)[`depositWithMessage`]
- * [xref:function-reference.adoc#depositWithMessageReclaim](#)[`depositWithMessageReclaim`]
- * [xref:architecture-and-concepts:network-architecture/messaging-mechanism.adoc#l1-l2-messages](#)[L1 handler]

[id="StarkGate_cancelling_a_deposit"]

= Cancelling a deposit

To ensure self-custody, StarkGate enables you to cancel a deposit if, after depositing funds with the `deposit` function on L1, you don't see your funds appear on L2 within a reasonable amount of time.

You can only cancel a deposit that you yourself deposited.

In order to guard against an attack, it takes approximately five days to cancel a deposit. From the moment StarkGate receives the cancellation request, a counter begins. When exactly five days have passed, and the funds still do not appear on L2, you can reclaim the deposit.

// a deposit, and only before `depositReclaim` was performed.

// If five days pass and the deposit request has still not been serviced, you can reclaim the deposit.

.Procedure

. To cancel a deposit, call the ``depositCancelRequest`` request function.

+

When StarkGate receives the cancellation request, a counter begins to count five days.

. When exactly five days have passed, and the funds still do not appear on L2, you can reclaim the deposit by calling ``depositReclaim``.

[NOTE]

====

As long as the ``depositReclaim`` was not performed, the deposit may be processed, even if the cancellation delay time has already passed.

Only the depositor is allowed to cancel a deposit, and only before `depositReclaim` was performed.

====

== Additional resources

* [xref:function-reference.adoc#depositCancelRequest\[`depositCancelRequest`\]](#) in the `_StarkGate` function reference_.

* [xref:function-reference.adoc#depositReclaim\[`depositReclaim`\]](#) in the `_StarkGate` function reference_.
[id="depositing-funds-with-starkgate"]

= Depositing funds with StarkGate

:description: How to deposit funds in StarkGate using a function in a block explorer, and what happens when you deposit funds into StarkGate.

Using StarkGate to deposit L1 funds into the L2 Starknet requires StarkGate's ``deposit`` function. The ``deposit`` function does the following:

. Transfers the funds from the user's Ethereum account to the StarkGate L1 contract.

. Emits a ``Deposit`` event that includes the L1 and L2 addresses of the user, and the amount deposited.

. Sends a message to the corresponding L2 bridge with the amount deposited, and the recipient's address.

Subsequently, the funds should be transferred to Starknet so that you can begin using them.

For more information on what happens during the transfer process, see [xref:overview.adoc#l1l2_transfer_deposit\[L1->L2 transfer \(deposit\)\]](#).

.Prerequisites

- * An Ethereum block explorer, such as link:<https://etherscan.io>[Etherscan].
- * Funds to transfer from L1 to L2, including enough to pay the fees required for the transfer.
- * The L1 address of the StarkGate bridge for the token you want to deposit. To view the token addresses for tokens on Mainnet or Sepolia testnet, see [xref:tools:bridged-tokens.adoc\[\]](#).

.Procedure

. Using an Ethereum block explorer, go to the StarkGate contract and click **Write as Proxy**. For example, using Etherscan, go to link:[https://etherscan.io/address/0xae0ee0a63a2ce6baeef56e7714fb4efe48d419#writeProxyContract\[0xae0ee0a63a2ce6baeef56e7714fb4efe48d419\]](https://etherscan.io/address/0xae0ee0a63a2ce6baeef56e7714fb4efe48d419#writeProxyContract[0xae0ee0a63a2ce6baeef56e7714fb4efe48d419])

. Click the StarkGate 2.0 ``deposit`` function (0x0efe6a8b).

. Enter the following:

+

* In `*`payableAmount`*`, enter the maximum amount of ETH that you're willing to pay for the ``deposit`` transaction fee.

* In `*`token (address)`*`, enter the address of the L1 contract for the token that you want to deposit to L2.

* In `*`amount (uint256)`*`, enter an integer for the amount of the token that you want to deposit to L2.

* In `*`l2Recipient (uint256)`*`, enter the address of the recipient on L2.

. Click **Write**. The ``deposit`` function initiates a deposit.

[id="starkgate_estimating_fees"]

= Estimating StarkGate fees

StarkGate enforces a minimum fee for all transactions to account for the L1 -> L2 message costs. For more information, see [xref:architecture-and-concepts:network-architecture/messaging-mechanism.adoc#l1-l2-message-fees\[L1 -> L2 message fees\]](#).

You can estimate the fee using the following L1 functions:

[horizontal,labelwidth="25",role="stripes-odd"]

``estimateDepositFeeWei`` :: Estimates the fee for a deposit transaction.

``estimateEnrollmentFeeWei`` :: Estimates the fee for an enrollment transaction.

.Additional resources

* [xref:function-reference.adoc#estimateDepositFeeWei\[`estimateDepositFeeWei`\]](#)

* [xref:function-reference.adoc#estimateEnrollmentFeeWei\[`estimateEnrollmentFeeWei`\]](#)

[id="StarkGate_function_reference"]

= StarkGate function and event reference

:description: Comprehensive function and event reference for StarkGate.

The StarkGate smart contracts include functions that you use to implement various flows in a dApp.

For information on the movement of funds between Ethereum and Starknet, see [xref:overview.adoc\[StarkGate bridge overview\]](#).

include::partial\$snippet_backwards_compatibility_note.adoc[]

The L1 functions and their interfaces, where available, are defined in the following smart contracts:

[cols=" , , ,]

|===

|Contract |Name |Functions

|<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarkgateManager.sol> |StarkgateManager.sol` |The StarkGate Manager.

Use the Manager to enroll a new token.

An interface is available through <https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/IStarkgateManager.sol> |IStarkgateManager.sol`. a| *

xref:#enrollTokenBridge[enrollTokenBridge`]

|<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarkgateRegistry.sol> |StarkgateRegistry.sol` a|The StarkGate Registry.

Use the Registry to view the addresses of existing bridges and to stop servicing a specific token.

An interface is available through <https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/IStarkgateRegistry.sol> |IStarkgateRegistry.sol`. a| *

xref:#getBridge[getBridge`]

* xref:getWithdrawalBridges[getWithdrawalBridges`]

|<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol> |StarknetTokenBridge.sol` |The primary StarkGate bridge contract. The functions and events in this reference that provide the main user bridge functionality are defined in this contract. a|

* xref:#deposit[deposit`]

* xref:#depositCancelRequest[depositCancelRequest`]

* xref:#depositReclaim[depositReclaim`]

* xref:#depositWithMessage[depositWithMessage`]

* xref:#depositWithMessageCancelRequest[depositWithMessageCancelRequest`]

* xref:#depositWithMessageReclaim[depositWithMessageReclaim`]

```
* xref:#getStatus[`getStatus`]
* xref:#withdraw[`withdraw`]
```

[<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/IStarkgateService.sol>] | An interface to check if a contract is servicing a token. a|

```
* xref:#isServicingToken[`isServicingToken`]
|===
```

The L2 functions and their interfaces, where available, are defined in the following smart contracts:

```
[cols=",,",]
|===
```

Contract	Description	Functions
----------	-------------	-----------

[https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/token_bridge.cairo] |The StarkGate bridge implementation on L2. This contract includes the standard functions for a token bridge.

An interface is available through https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/token_bridge_interface.cairo | token_bridge_interface.cairo`. a|

```
* xref:#get_erc20_class_hash[`get_erc20_class_hash`]
* xref:#get_identity[`get_identity`]
* xref:#get_l1_token[`get_l1_token`]
* xref:#get_l2_token[`get_l2_token`]
* xref:#get_remaining_withdrawal_quota[`get_remaining_withdrawal_quota`]
* xref:#get_version[`get_version`]
* xref:#initiate_token_withdraw[`initiate_token_withdraw`]
* xref:#on_receive[`on_receive`]
```

```
|===
```

// Need to add Backward compatibility bridge.

== L1 function reference

Functions are listed in alphabetical order.

'''

```
[#deposit]
=== `deposit`
```

```
[discrete]
```

==== Description

Deposits the specified amount of an ERC-20 token to the L1 StarkGate bridge contract.

The deposit function does the following:

- * Transfers the funds from the caller's account to the Starknet bridge contract
- * Emits the `Deposit` event with the sender's address on L1, the recipient's address on L2, and the amount

[discrete]

==== Visibility

`external`

[discrete]

==== State Mutability

`payable`

[discrete]

==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]

`address_token`::

// tag::address_token[]

The address of the contract for the desired token.

// end::address_token[]

`uint256_amount`::

// tag::uint256_amount_deposit[]

The amount of the deposit.

// end::uint256_amount_deposit[]

`uint256_l2Recipient`::

// tag::uint256_l2Recipient[]

The L2 address of the recipient.

// end::uint256_l2Recipient[]

[discrete]

==== Returns

None.

[discrete]

==== Emitted event

`Deposit`

.Event attributes

[horizontal,labelwidth="35",role=stripes-odd]

`address indexed _sender_`:: The L1 address of the account that sent the deposit.

`address indexed _token_`::

include::function-reference.adoc[tag=address_token]

`uint256 _amount_`::

include::function-reference.adoc[tag=uint256_amount_deposit]

`uint256 indexed _l2Recipient_`::

include::function-reference.adoc[tag=uint256_l2Recipient]

`uint256 _nonce_`:: The nonce for the L1 transaction.

`uint256 _fee_`:: The Starknet fee sent with the transaction.

[discrete]

==== Function and event definitions

Contract: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol>[`StarknetTokenBridge.sol`]

* Function: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L284>[`deposit`]

* Event: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L65>[`Deposit`]

'''

[#depositCancelRequest]

=== `depositCancelRequest`

[discrete]

==== Description

Sends a request to StarkGate to cancel a deposit.

You can send a cancellation request if the funds you transfer from L1 to L2 do not appear on L2 within a reasonable amount of time.

In order to guard against an attack on Starknet, it takes approximately five days to cancel a deposit. When StarkGate receives the cancellation request, a counter begins. When exactly five days have passed, and the funds still do not appear on L2, you can reclaim the deposit using the xref:#depositReclaim[`depositReclaim`] function.

The `depositReclaim` function can only be used once for any deposit cancellation

request.

[discrete]
==== Visibility

`external`

[discrete]
==== State Mutability

`nonpayable`

[discrete]
==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]
`address _token_`:: The address of the contract for the desired token.
`uint256 _amount_`:: The amount of the deposit.
`uint256 _l2Recipient_`::
// tag::uint256_l2Recipient[]
The L2 address of the recipient.
// end::uint256_l2Recipient[]
`uint256 _nonce_`::
// tag::uint256_nonce[]
The nonce of the deposit.
// end::uint256_nonce[]

[discrete]
==== Returns

None.

[discrete]
==== Emitted event

`DepositCancelRequest`

.Event attributes
[horizontal,labelwidth="30",role=stripes-odd]
`address indexed _sender_`:: The L1 address of the account that sent the deposit.
`address indexed _token_`::
include::function-reference.adoc[tag=address_token]
`uint256 _amount_`::
include::function-reference.adoc[tag=uint256_amount_deposit]
`uint256 indexed _l2Recipient_`::

include::function-reference.adoc[tag=uint256_l2Recipient]
`uint256 _nonce`::
include::function-reference.adoc[tag=uint256_nonce]

[discrete]
==== Function and event definitions

Contract: <https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol> [`StarknetTokenBridge.sol`]

* Function: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L515> [`depositCancelRequest`]
* Event: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L73> [`DepositCancelRequest`]

'''

[#depositReclaim]
=== `depositReclaim`

[discrete]
==== Description

Reclaims a deposit after a five day period has passed from the time that StarkGate received a deposit cancellation request from the `depositCancelRequest` function.

[discrete]
==== Visibility

`external`

[discrete]
==== State Mutability

`nonpayable`

[discrete]
==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]
`address _token`:: The address of the contract for the desired token.
`uint256 _amount`:: The amount of the deposit.
`uint256 _l2Recipient`:: The L2 address of the recipient.
`uint256 _nonce`:: The nonce of the deposit.

[discrete]
==== Returns

None.

[discrete]
==== Emitted event

`event DepositReclaimed`

.Event attributes

[horizontal,labelwidth="30",role=stripes-odd]
`address indexed _sender_`:: The L1 address of the account that sent the deposit.
`address indexed _token_`::
include::function-reference.adoc[tag=address_token]
`uint256 _amount_`::
include::function-reference.adoc[tag=uint256_amount_deposit]
`uint256 indexed _l2Recipient_`::
include::function-reference.adoc[tag=uint256_l2Recipient]
`uint256 _nonce_`::
include::function-reference.adoc[tag=uint256_nonce]

[discrete]
==== Function and event definitions

Contract: `StarknetTokenBridge.sol`

* Function: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L588>[`depositReclaim`]
* Event: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L80>[`DepositReclaimed`]

[discrete]
==== See also

* xref:#depositCancelRequest[`depositCancelRequest`]

'''

[#depositWithMessage]
=== `depositWithMessage`

[discrete]
==== Description

Similar to `xref:#deposit[deposit]`, with a message attached.

With this function, a deposit transaction can trigger subsequent actions. For example, you can deposit funds and include a message to transfer those funds to another address. `depositWithMessage` lets you execute these two separate transactions with a single user action.

After depositing to another recipient, the L1 handler in `token_bridge.cairo` calls the `on_receive` function in the contract of the recipient.

If `on_receive` returns `true`, then the `on_receive` function succeeded. If it returns `false`, or if it doesn't return any value because the `on_receive` function is not implemented in the recipient contract, the operation fails and the transaction is reverted.

```
//  
// ##----  
// fn on_receive(  
// self: @ContractState,  
// token_address: ContractAddress,  
// amount: u256,  
// depositor: EthAddress,  
// message: Span<felt252>  
// ) -> bool {  
// ----##
```

```
[discrete]  
==== Visibility
```

```
`external`
```

```
[discrete]  
==== State Mutability
```

```
`payable`
```

```
[discrete]  
==== Parameters
```

```
[horizontal,labelwidth="30",role=stripes-odd]  
`address _token_`:: The address of the contract for the desired token.  
`uint256 _amount_`:: The amount of the deposit.  
`uint256 _l2Recipient_`:: The L2 address of the recipient.  
`uint256[] calldata _message_`::  
// tag::calldata_message[]  
The message attached to the deposit.  
// end::calldata_message[]
```

[discrete]
==== Returns

None.

[discrete]
==== Emitted event

`DepositWithMessage`

.Event attributes

[horizontal,role=stripes-odd]

`address indexed _sender_`:: The L1 address of the account that sent the deposit.

`address indexed _token_`::

include::function-reference.adoc[tag=address_token]

`uint256 _amount_`::

include::function-reference.adoc[tag=uint256_amount_deposit]

`uint256 indexed _l2Recipient_`::

include::function-reference.adoc[tag=uint256_l2Recipient]

`uint256[] _message_`::

include::function-reference.adoc[tag=calldata_message]

`uint256 _nonce_`:: The nonce for the L1 transaction.

`uint256 _fee_`:: The Starknet fee sent with the transaction.

[discrete]
==== Function and event definitions

Contract: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol>[StarknetTokenBridge.sol]

* Function: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L255>[`depositWithMessage`]

* Event: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L37>[`DepositWithMessage`]

'''

[#Continue review with Roe here#]

[#depositWithMessageCancelRequest]
=== `depositWithMessageCancelRequest`

[discrete]

==== Description

Sends a request to StarkGate to cancel a deposit sent with `depositWithMessage`.

Similar to `depositCancelRequest`.

[discrete]

==== Visibility

`external`

[discrete]

==== State Mutability

`nonpayable`

[discrete]

==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]

`address _token_`:: The address of the contract for the desired token.

`uint256 _amount_`:: The amount of the deposit.

`uint256 _l2Recipient_`:: The L2 address of the recipient.

`uint256[] calldata _message_`:: The message attached to the deposit.

`uint256 _nonce_`:: The nonce of the deposit.

[discrete]

==== Returns

None.

[discrete]

==== Emitted event

`DepositWithMessageCancelRequest`

.Event attributes

[horizontal,role=stripes-odd]

`address indexed _sender_`:: The L1 address of the account that sent the deposit.

`address indexed _token_`::

include::function-reference.adoc[tag=address_token]

`uint256 _amount_`::

include::function-reference.adoc[tag=uint256_amount_deposit]

`uint256 indexed _l2Recipient_`::

include::function-reference.adoc[tag=uint256_l2Recipient]

``uint256[] _message_`::`
include::function-reference.adoc[tag=calldata_message]
``uint256 _nonce_`::` The nonce for the L1 transaction.

[discrete]
==== Function and event definitions

Contract: link:https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol[StarknetTokenBridge.sol]

* Function: link:https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L534[``depositWithMessageCancelRequest``]
* Event: link:https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L46[``DepositWithMessageCancelRequest``]

[discrete]
==== See also

* xref:#depositWithMessage[``depositWithMessage``]
* xref:#depositCancelRequest[``depositCancelRequest``]

'''

[#depositWithMessageReclaim]
=== ``depositWithMessageReclaim``

[discrete]
==== Description

Sends a request to StarkGate to cancel a deposit sent with ``depositWithMessage``.

Similar to ``depositCancelRequest``.

Reclaims a deposit sent with a message after a five day period has passed from the time that StarkGate received a deposit cancellation request from the ``depositWithMessageCancelRequest`` function.

[discrete]
==== Visibility

``external``

[discrete]

==== State Mutability

`nonpayable`

[discrete]

==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]

`address _token_`:: The address of the contract for the desired token.

`uint256 _amount_`:: The amount of the deposit.

`uint256 _l2Recipient_`:: The L2 address of the recipient.

`uint256 _nonce_`:: The nonce of the deposit.

[discrete]

==== Returns

None.

[discrete]

==== Emitted event

`DepositWithMessageCancelRequest`

.Event attributes

[horizontal,role=stripes-odd]

`address indexed _sender_`:: The L1 address of the account that sent the deposit.

`address indexed _token_`::

include::function-reference.adoc[tag=address_token]

`uint256 _amount_`::

include::function-reference.adoc[tag=uint256_amount_deposit]

`uint256 indexed _l2Recipient_`::

include::function-reference.adoc[tag=uint256_l2Recipient]

`uint256[] _message_`::

include::function-reference.adoc[tag=calldata_message]

`uint256 _nonce_`:: The nonce for the L1 transaction.

[discrete]

==== Function and event definitions

Contract: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol>[StarknetTokenBridge.sol]

* Function: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L564>[`depositWithMessageReclaim`]

* Event: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L54>[`DepositWithMessageReclaimed`]

[discrete]
==== See also

- * xref:#depositCancelRequest[`depositCancelRequest`]
- * xref:#depositReclaim[`depositReclaim`]
- * xref:depositWithMessageCancelRequest[`depositWithMessageCancelRequest`]

'''

[#enrollTokenBridge]
=== `enrollTokenBridge`

[discrete]
==== Description

Creates a Starknet bridge for the specified ERC-20 token contract address in the multi-token bridge contracts and adds the token to the StarkGate Registry.

Does not work for any ERC-20 token bridge's contract address that is already in the registry.

Enrolling a new bridge creates a new ERC-20 contract on L2. You can see the class hash for this contract using xref:#get_erc20_class_hash[`get_erc20_class_hash`].

include::partial\$snippet_enrollTokenBridge_note.adoc[]

[discrete]
==== Visibility

`external`

[discrete]
==== State Mutability

`payable`

The message payload needs to include funds to cover the Starknet (L2) fee for executing this transaction. You can include this payload using a standard wrapper such as web3.js.

[discrete]
==== Parameters

[horizontal,labelwidth="20",role=stripes-odd]

`address _token_`: The address of the contract for the desired ERC-20 token.

[discrete]

==== Returns

None.

[discrete]

==== Emitted event

`TokenEnrollmentInitiated`

[discrete]

==== Function and event definition

* Function: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarkgateManager.sol#L132>[`enrollTokenBridge`] in `StarkgateManager.sol`

* Event: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L34>[`TokenEnrollmentInitiated`] in `StarknetTokenBridge.sol`

'''

[#estimateDepositFeeWei]

=== `estimateDepositFeeWei`

[discrete]

==== Description

Returns an estimate of the fee, in Wei, for depositing funds to the L1 StarkGate bridge contract.

[discrete]

==== Visibility

`external`

[discrete]

==== State Mutability

`view`

[discrete]

==== Parameters

None.

[discrete]
==== Returns

[horizontal,labelwidth="20",role=stripes-odd]
`uint256`:: An estimate of the fee, in Wei, for depositing funds to the L1 StarkGate bridge contract.

[discrete]
==== Emitted event

None.

[discrete]
==== Function definition

link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L142C28-L142C28> `estimateDepositFeeWei` in
`StarknetTokenBridge.sol`

[discrete]
==== Additional resources

* xref:#deposit[`deposit`]
* xref:#depositWithMessage[`depositWithMessage`]

'''

[#estimateEnrollmentFeeWei]
=== `estimateEnrollmentFeeWei`

[discrete]
==== Description

Returns an estimate of the fee, in Wei, for creating and registering a new bridge using the `enrollTokenBridge` function.

[discrete]
==== Visibility

`external`

[discrete]
==== State Mutability

`view`

[discrete]
==== Parameters

None.

[discrete]
==== Returns

[horizontal,labelwidth="20",role=stripes-odd]
`uint256`:: An estimate of the fee, in Wei, for creating and registering a new bridge.

[discrete]
==== Emitted event

None.

[discrete]
==== Function definition

link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L146> [`estimateEnrollmentFeeWei`] in
`StarknetTokenBridge.sol`

[discrete]
==== Additional resource

* xref:#enrollTokenBridge[`enrollTokenBridge`]

'''

[#getBridge]
=== `getBridge`

[discrete]
==== Description

Returns the address of the bridge for the specified token, or a value indicating if the bridge does not exist, is blocked, or is deactivated.

[discrete]
==== Visibility

`external`

[discrete]
==== State Mutability

`view`

[discrete]
==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]
`address _token_`:: The address of the contract for the desired ERC-20 token.

[discrete]
==== Returns

[horizontal,labelwidth="20",role=stripes-odd]
`address _address_`:: The address of the bridge for the specified token.
`Address(0)`:: The bridge does not exist.
`Address(1)`:: The bridge is blocked or deactivated.

[discrete]
==== Emitted event

None.
[discrete]
==== Function definition

link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/IStarkgateRegistry.sol#L8>[`getBridge`] in `IStarkgateRegistry.sol`

'''

[#getRegistry]
=== `getRegistry`

[discrete]
==== Description

Returns the address of the StarkGate Registry contract.

Only the Manager uses this function.

[discrete]
==== Visibility

`external`

[discrete]
==== State Mutability

`view`

[discrete]
==== Parameters

None

[discrete]
==== Returns

[horizontal,labelwidth="20",role=stripes-odd]
`address` :: The address of the Registry contract.

[discrete]
==== Emitted event

None.

[discrete]
==== Function definition

link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarkgateManager.sol#L24>[`getRegistry`] in `StarkgateManager.sol`

'''

[#getStatus]
=== `getStatus`

[discrete]
==== Description

Returns the status of a token in StarkGate.

`deploy` transaction triggered by the `enrollTokenBridge` API.

[discrete]
==== Visibility

`external`

[discrete]
==== State Mutability

`view`

[discrete]
==== Parameters

[horizontal,labelwidth="20",role=stripes-odd]
`address _token_`:: The address of the contract for the desired ERC-20 token.

[discrete]
==== Returns

[horizontal,labelwidth="20",role=stripes-odd]
`TokenStatus`:: One of the following values:

+

[horizontal]

0::: Unknown. The bridge does not recognize the token.

1::: Pending. The token has been enrolled to StarkGate, but the `deploy` transaction has not yet successfully completed. You can deposit funds.

2::: Active. The `deploy` transaction for this token has completed successfully and StarkGate recognizes the token.

3::: Deactivated. The token has been removed from StarkGate. You cannot deposit funds.

[discrete]
==== Emitted event

None.

[discrete]
==== Function definition

link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L198C23-L198C23> [`getStatus`] in `StarknetTokenBridge.sol`

'''

[#getWithdrawalBridges]
=== `getWithdrawalBridges`

[discrete]
==== Description

Retrieves a list of all bridge addresses that have ever facilitated withdrawals for the specified token.

In a case where an inactive bridge for a specific token might still have funds locked, you can use this function to identify all bridges that ever serviced that token.

If you used a bridge for a given token that subsequently was replaced with a new or updated bridge, but you still have funds locked on the first bridge, you might not know the address of the old bridge. This function returns

[discrete]
==== Visibility

`external`

[discrete]
==== State Mutability

`view`

[discrete]
==== Parameters

[horizontal,labelwidth="20",role=stripes-odd]
`address _token_`:: The address of the contract for the desired token.

[discrete]
==== Returns

[horizontal,labelwidth="30",role=stripes-odd]
`address[] memory _bridges_`:: An array of addresses of all bridges that ever serviced `token`.

[discrete]
==== Emitted event

None.

[discrete]
==== Function definition

link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarkgateRegistry.sol#L110>[`getWithdrawalBridges`] in `StarkgateRegistry.sol`.

""
[#identify]
=== `identify`

[discrete]
==== Description

Returns the name and version of the `StarknetTokenBridge.sol` contract.

[discrete]
==== Visibility

`external`

[discrete]
==== State Mutability

`pure`

[discrete]
==== Parameters

None.

[discrete]
==== Returns

[horizontal,labelwidth="20",role=stripes-odd]
`string _memory_`:: The name and version of the `StarknetTokenBridge.sol` contract.

[discrete]
==== Emitted event

None.

[discrete]

==== Function definition

link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L92>[`identify`] in `StarknetTokenBridge.sol`

'''

[`#isServicingToken`]

==== `isServicingToken`

[discrete]

==== Description

Checks whether the calling contract is currently providing a service for the specified token.

[discrete]

==== Visibility

`external`

[discrete]

==== State Mutability

`view`

[discrete]

==== Parameters

[horizontal,labelwidth="20",role=stripes-odd]

`address _token_`:: The address of the contract for the desired token.

[discrete]

==== Returns

[horizontal,labelwidth="20",role=stripes-odd]

`true`:: The calling contract is currently providing a service for the token.

`false`:: The calling contract is not currently providing a service for the token.

[discrete]

==== Emitted event

None.

[discrete]

==== Function definition

link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/IStarkgateService.sol#L9C14-L9C30> `isServicingToken`` in `IStarkgateService.sol``.

'''

[#withdraw]
=== `withdraw`

[discrete]
==== Description

Transfers the specified amount of the specified token to the address of the recipient specified in the `l1_recipient`` parameter of the `initiate_token_withdraw`` function on L2.

Anyone can call this function, but only after the withdraw message has been recorded on the Starknet Core Contract.

[discrete]
==== Parameters

[horizontal,labelwidth="30",role=stripes-odd]
``address_token_`::` The address of the contract for the desired token.
``uint256_amount_`::`
// tag::uint256_amount_withdrawal[]
The amount of the withdrawal.
// end::uint256_amount_withdrawal[]
``address_recipient_`::`
(Optional)
// tag::address_recipient[]
The recipient.
// end::address_recipient[]
If you don't specify this parameter, the withdraw function uses the sender's address.

[discrete]
==== State mutability

``nonpayable``

[discrete]
==== Returns

None.

[discrete]
==== Emitted event

`Withdrawal`

.Event attributes

[horizontal,role=stripes-odd]

`address indexed _recipient_`::

include::function-reference.adoc[tag=address_recipient]

`address indexed _token_`::

include::function-reference.adoc[tag=address_token]

`uint256 _amount_`::

include::function-reference.adoc[tag=uint256_amount_withdrawal]

[discrete]

==== Function and event definition

Contract: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol>[`StarknetTokenBridge.sol`]

* Function: link:<https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L481>[`withdraw`]

* Event: <https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/solidity/StarknetTokenBridge.sol#L62>[`Withdrawal`]

== L2 function reference

Functions are listed in alphabetical order.

'''

[#get_erc20_class_hash]

=== `get_erc20_class_hash`

[discrete]

==== Description

Returns the current class hash of the implementation used by the ERC-20 contract. Use the class hash as the type when deploying the ERC-20 contract on L2.

[IMPORTANT]

====

If StarkWare changes the class hash such that it is no longer the class hash that you used when deploying your bridge contract, this function returns the new class hash. If you want to refer to the class hash that you used when deploying your contract, see your deployed contract on Starknet.

====

[discrete]

==== Parameters

None.

[discrete]
==== Returns

[horizontal,labelwidth="20",role=stripes-odd]
`ClassHash`:: The class hash of the ERC-20 token contract.

[discrete]
==== Function definition

link:https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/token_bridge.cairo#L376-L378[`get_erc20_class_hash`] in `token_bridge.cairo`.

'''

[#get_identity]
=== `get_identity`

[discrete]
==== Description

Returns a string in a felt252 type with the identity of StarkGate.

[discrete]
==== Parameters

None.

[discrete]
==== Returns

[horizontal,labelwidth="20",role=stripes-odd]
`felt252`:: The identity of StarkGate.

[discrete]
==== Function definition

link:https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/token_bridge_interface.cairo#L8[`get_identity`] in `token_bridge_interface.cairo`.

'''

[#get_l1_token]
=== `get_l1_token`

[discrete]
==== Description

Returns the L1 address that corresponds to the matching L2 address of an ERC-20 token contract.

[discrete]
==== Parameters

[horizontal,labelwidth="20",role=stripes-odd]
`_l2_token_address_`: ContractAddress`:: The L2 address of the ERC-20 token contract.

[discrete]
==== Returns

[horizontal,labelwidth="20",role=stripes-odd]
`EthAddress`:: The L1 address of the ERC-20 token contract.
`EthAddressZeroable::zero()`:: The token is not found in the bridge.

[discrete]
==== Function definition

link:https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/token_bridge_interface.cairo#L9[`get_l1_token`] in `token_bridge_interface.cairo`.

'''

[#get_l2_token]
=== `get_l2_token`

[discrete]
==== Description

Returns the L2 address that corresponds to the matching L1 address of an ERC-20 token contract.

If the token is not found in the bridge, this function returns `0`.

[discrete]
==== Parameters

[horizontal,labelwidth="20",role=stripes-odd]

`l1_token_address`:: The L1 address of the ERC-20 token contract.

[discrete]

==== Returns

[horizontal,labelwidth="20",role=stripes-odd]

`ContractAddress _address_`:: The L2 address of the ERC-20 token contract.

`ContractAddressZeroable::zero`():: The token is not found in the bridge.

[discrete]

==== Function definition

link:https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/token_bridge_interface.cairo#L10[`get_l2_token`] in `token_bridge_interface.cairo`.

'''

[#get_remaining_withdrawal_quota]

=== `get_remaining_withdrawal_quota`

[discrete]

==== Description

Returns the amount that the user can withdraw within the current 24-hour time period. The time period begins at 00:00 UTC.

[discrete]

==== Parameters

[horizontal,labelwidth="25",role=stripes-odd]

`l1_token_address`:: The L1 address of the ERC-20 token contract.

[discrete]

==== Returns

[horizontal,labelwidth="25",role=stripes-odd]

`u256`:: The amount that can currently be withdrawn from the bridge, in units defined by the ERC-20 token contract.

[discrete]

==== Function definition

link:https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/token_bridge_interface.cairo#L11[`get_remaining_withdrawal_quota`] in
`token_bridge_interface.cairo`

'''

[#get_version]
=== `get_version`

[discrete]
==== Description

Returns the current version of StarkGate.

[discrete]
==== Parameters

None.

[discrete]
==== Returns

[horizontal,labelwidth="25",role=stripes-odd]
`felt252`:: The current version of StarkGate.

[discrete]
==== Function definition

link:https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/token_bridge_interface.cairo#L7[`get_version`] in `token_bridge_interface.cairo`.

'''

[#initiate_token_withdraw]
=== `initiate_token_withdraw`

[discrete]
==== Description

Initiates a withdrawal from L2. After initiating the withdrawal, the function does the following:

- . Burns the transferred amount of tokens from the balance of the withdrawal's initiator.
- . Sends a message to the relevant L1 bridge with the amount to be transferred, and the recipient's address.

[discrete]

==== Parameters

[horizontal,labelwidth="25",role=stripes-odd]

`l1_token: EthAddress`:: The L1 address of the ERC-20 token contract.

`l1_recipient: EthAddress`:: The L1 address of the recipient.

`_amount_ uint256`:: The amount to transfer.

[discrete]

==== Returns

None.

[discrete]

==== Function definition

link:https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/token_bridge_interface.cairo#L13[`initiate_token_withdraw`] in
`token_bridge_interface.cairo`.

'''

[#on_receive]

=== `on_receive`

[discrete]

==== Description

An interface to an implementation of the `on_receive` function that you must provide in your L2 contract in order to enable the `depositWithMessage` function to succeed.

// When calling the `depositWithMessage` function, the resulting transaction triggers an `on_receive` function that you need to implement in your application.

//

The L2 contract that receives the message that is sent with the `depositWithMessage` function must implement a callback function named `on_receive`.

Upon completion, the `depositWithMessage` function triggers a call to the `on_receive` callback function on the receiving L2 contract. The `on_receive` function receives the

deposit message as input, and it must return `true` for the deposit to succeed.

If `on_receive` returns `false`, or if the receiving contract does not implement `on_receive`, the `depositWithMessage` L1 handler fails, and the user can only recover their funds using the

xref:#depositWithMessageCancelRequest[`depositWithMessageCancelRequest`] function.

[discrete]

==== Parameters

[horizontal,labelwidth="35",role=stripes-odd]

`_l2_token_`: ContractAddress:: The L2 address of the ERC-20 token contract.

`_amount_`: uint256:: The amount deposited.

`_depositor_`: EthAddress:: L1 address of the deposit sender.

`_message_`: Span<felt252>:: The message that was sent with the `depositWithMessage` function.

[discrete]

==== Returns

[horizontal,labelwidth="25",role=stripes-odd]

`true`:: The `on_receive` function completed successfully.

`false`:: The `on_receive` function did not complete successfully. The transaction is reverted.

// The recipient L2 contract of the `depositWithMessage` function does not accept the funds on L2.

No value:: If the recipients's smart contract does not implement the `on_receive` function, the call fails to execute, and the transaction is reverted.

[discrete]

==== Function definition

link:[https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/](https://github.com/starknet-io/starkgate-contracts/tree/v2.0.1/src/cairo/receiver_interface.cairo#L6)

receiver_interface.cairo#L6[`on_receive`] in `receiver_interface.cairo`.

[id="starkgate_token_bridge"]

= StarkGate bridge overview

StarkGate, developed by StarkWare, bridges ETH and ERC-20 tokens between Ethereum and Starknet. Each supported token is associated with an L1 and L2 bridge contract that communicates via Starknet's messaging mechanism.

To use the StarkGate web app, go to <https://starkgate.starknet.io>.

A bridge enables you to fund your L2 wallet with ETH and ERC-20 tokens that reside on L1.

The terms `_deposit_`, `_transact_`, and `_transfer_` refer to various operations involving a bridge, even though ETH and ERC-20 tokens never actually leave Ethereum.

include::partial\$snippet_backwards_compatibility_note.adoc[]

[#starkgate_addresses]
== StarkGate addresses

L1 and L2 addresses for StarkGate bridges and supported tokens are listed in the JSON files in the Starknet GitHub repository shown in the table
xref:#table_StarkGate_token_addresses[].

[#table_StarkGate_token_addresses]
.StarkGate bridged tokens and their addresses
|===
| Network | StarkGate bridged tokens JSON file

| Mainnet | link:https://github.com/starknet-io/starknet-addresses/blob/master/bridged_tokens/mainnet.json[mainnet.json]
| Sepolia testnet | link:https://github.com/starknet-io/starknet-addresses/blob/master/bridged_tokens/sepolia.json[sepolia.json]
|===

[#starkgate_supported_tokens]
== Supported tokens in StarkGate

StarkGate supports many tokens, including ETH, WBTC, USDC, DAI, and many more.

For a comprehensive list of tokens that StarkGate supports, including their L1 and L2 addresses, see the JSON files in the Starknet GitHub repository shown in the table
xref:#table_StarkGate_token_addresses[].

[NOTE]

====
Previously, StarkGate placed limitations for each supported token on the amount that could be deposited and the total value locked in the L1 bridge contract on Mainnet. These limits have been removed.
====

[id="l1l2_transfer_deposit"]
== L1->L2 transfer (deposit)

[id="step_1_call_the_deposit_function_on_l1"]
=== Step 1: Call the deposit function on L1

. A call to the L1 `deposit` function initiates a deposit.

. The function does the following:

+

--

* Transfers the funds from the user's account to the Starknet bridge.

* Emits a `Deposit` event that includes the L1 and L2 addresses of the user, and the amount deposited.

* Sends a message to the corresponding L2 bridge with the amount deposited, and the recipient's address.

--

+

Starknet's sequencer is now aware of the deposit transaction.

. The sequencer waits for enough L1 block confirmations to fill its quota to run before the corresponding deposit transaction is initiated on L2. During this period of time, the status of the L2 deposit transaction is `xref:architecture-and-concepts:network-architecture/transaction-life-cycle.adoc#not_received[NOT_RECEIVED]`.

[id="step_2_deposit_triggered_on_starknet"]

=== Step 2: Deposit triggered on Starknet

. The sequencers refer to the deposit

request by triggering the L1 handler using the

[https://github.com/starkware-libs/starkgate-contracts/](https://github.com/starkware-libs/starkgate-contracts/blob/28f4032b101003b2c6682d753ea61c86b732012c/src/starkware/starknet/apps/starkgate/cairo/token_bridge.cairo#L135)

`blob/28f4032b101003b2c6682d753ea61c86b732012c/src/starkware/starknet/apps/starkgate/cairo/token_bridge.cairo#L135[handle_deposit]` function on the L2 bridge.

. The `handle_deposit` function verifies that the deposit indeed came from the corresponding L1 bridge. It then calls the relevant token's contract on Starknet and mints the specified amount of the token on L2 for the user.

. The sequencers complete constructing the block.

The status of the deposit request is now `xref:architecture-and-concepts:network-architecture/transaction-life-cycle.adoc#accepted_on_l2[ACCEPTED_ON_L2]`.

[id="step_3_the_block_that_includes_the_transfer_is_proved"]

=== Step 3: The block that includes the transfer is proved

. Starknet's provers prove the validity of the block and submit a state update to L1.

. The message confirming transfer of the funds is cleared from the Starknet Core Contract, and the fact that the user has transferred their funds is part of the now finalized state of Starknet.

[NOTE]

====

If the message wasn't on L1 to begin with, that is, if the deposit request was fraudulently created on Starknet, the state update fails.

====

[id="l2l1_transfer_withdraw"]

== L2->L1 transfer (withdrawal)

[id="step_1_call_the_withdraw_function_on_l2"]

=== Step 1: Initiate a withdrawal from L2

. A call to the L2 `initiate_token_withdraw` function initiates a withdrawal.

. The function does the following:

* Burns the transferred amount of tokens from the balance of the withdrawal's initiator.

* Sends a message to the relevant L1 bridge with the amount to be transferred and the recipient's address.

[#proving_the_block_that_includes_the_withdrawal_transaction]

=== Step 2: Proving the block that includes the withdrawal transaction

// Once the sequencer completes the block construction, Starknet's provers prove the validity of the block and submit a state update to L1. The message from the previous step is then stored in the Starknet Core Contract.

. The sequencer completes the block construction

. Starknet's provers prove the validity of the block and submit a state update to L1.

. The message from the previous step is stored in the Starknet Core Contract.

[id="step_3_transferring_the_funds_on_l1"]

=== Step 3: Transferring the funds on L1

After the withdrawal message has been recorded on the Starknet Core Contract, anyone can finalize the transfer on L1 from the bridge back to the user, by calling the `xref:function-reference.adoc#withdraw[withdraw]` function.

[NOTE]

====

This step is permissionless, anyone can do it. The recipient's address is part of the recorded message on L1, so they receive the funds regardless of who calls the `withdraw` function on L1.

====

== Additional resources

- * xref:architecture-and-concepts:network-architecture/messaging-mechanism.adoc[L1-L2 messaging]
- * The `_StarkGate` developer's reference_:
 - ** xref:function-reference.adoc#deposit[`deposit`` function, ``Deposit`` event]
 - ** xref:function-reference.adoc#withdraw[`withdraw``]
 - ** xref:function-reference.adoc#initiate_token_withdraw[`initiate_token_withdraw``]
- * link:https://github.com/starkware-libs/starkgate-contracts/blob/28f4032b101003b2c6682d753ea61c86b732012c/src/starkware/starknet/apps/starkgate/cairo/token_bridge.cairo#L135[`handle_deposit``] function on the L2 bridge [id="withdrawing-funds-with-starkgate"]

= Withdrawing funds with StarkGate

:description: How to use StarkGate to withdraw funds from Starknet using a function in a block explorer, and what happens when you use StarkGate to withdraw funds.

Using StarkGate to withdraw funds from Starknet requires StarkGate's `initiate_token_withdraw`` function to initiate a withdrawal. The function does the following:

- * Burns the transferred amount of tokens from the L2 balance of the withdrawal's initiator.
- * Sends a message to the relevant L1 bridge with the amount to be transferred and the recipient's Ethereum address.

Subsequently, the funds should be transferred to the recipient's Ethereum address.

For more information on what happens during the transfer process, see xref:overview.adoc#l2l1_transfer_withdraw[L2->L1 transfer (withdrawal)].

.Prerequisites

- * A Starknet block explorer. For a list of Starknet block explorers, see link:https://www.starknet.io/en/ecosystem/block-explorers-indexers-and-enhanced-api[Block explorers, indexers & Enhanced API] on the Starknet site.
- * An Ethereum block explorer, such as link:https://etherscan.io[Etherscan].
- * Funds to transfer from L2 to L1, including enough to pay the fees required for the transfer.
- * The L2 address of the StarkGate bridge for the token you want to withdraw. To view the token addresses for tokens on Mainnet or Sepolia testnet, see xref:tools:bridged-tokens.adoc[.].

.Procedure

. Using a Starknet block explorer, go to the StarkGate contract for the token you want to withdraw. For example, to withdraw USDC:

+

* If using Voyager, go to the link:<https://voyager.online/contract/0x05cd48fccbfd8aa2773fe22c217e808319ffcc1c5a6a463f7d8fa2da48218196>[USDC StarkGate bridge] and click *Write Contract*.

* If using StarkScan, go to the link:<https://starkscan.co/contract/0x05cd48fccbfd8aa2773fe22c217e808319ffcc1c5a6a463f7d8fa2da48218196>[USDC StarkGate bridge] and click menu:Read/Write Contract[Write].

. Click the `initiate_token_withdraw` function and enter the following:

+

* In *l1_token*, enter the L1 address of the ERC-20 contract for the token you want to withdraw, in this case, USDC.

* In *l1_recipient*, enter the L1 address of the recipient.

* In *amount*, enter the amount to transfer.

. Click *Transact* for Voyager, or *Write* for StarkScan. The function initiates a withdrawal.

. Using an Ethereum block explorer, go to the StarkGate contract and click *Write as Proxy*. For example, using Etherscan, go to link:[https://etherscan.io/address/0xae0ee0a63a2ce6baeeffe56e7714fb4efe48d419#writeProxyContract\[0xae0ee0a63a2ce6baeeffe56e7714fb4efe48d419\]](https://etherscan.io/address/0xae0ee0a63a2ce6baeeffe56e7714fb4efe48d419#writeProxyContract[0xae0ee0a63a2ce6baeeffe56e7714fb4efe48d419])

. Click the StarkGate 2.0 *withdraw (0x69328dec)* function.

. Enter the following:

+

[horizontal,role=stripes-odd]

`recipient (address)`::

include::function-reference.adoc[tag=address_recipient]

`token (address)`::

include::function-reference.adoc[tag=address_token]

`amount (uint256)`::

include::function-reference.adoc[tag=uint256_amount_withdrawal]

. Click *Write*.

The `withdraw` function withdraws the funds to the recipient's L1 address. The funds should be available after the next L1 state update.

[NOTE]

====

StarkGate 2.0 provides a contract that enables seamless backward compatibility with the previous version of StarkGate.

When you update the code in your contract, make sure that you use the most up-to-date versions of all StarkGate contracts.

====[NOTE]

====

Enrolling a new bridge does not add it to the StarkGate GUI. You can use a block explorer to use the newly created bridge.

====* Release information

** xref:version-notes.adoc[Release notes]

** xref:deprecated.adoc[Deprecated, unsupported, and removed features]

[id="eol"]

= Deprecated, unsupported, and removed features

The features on this page are deprecated, unsupported, or removed from Starknet.

[horizontal,labelwidth="15"]

Deprecated:: Refers to a feature or capability that is still supported, but support will be removed in a future release of Starknet.

Future fixes or enhancements are unlikely. If necessary, an alternative is available.

Unsupported:: Refers to a feature or capability that is no longer supported.

Removed:: Refers to a feature or capability that has been entirely removed.

== Deprecated features

[cols="1,3",]

|===

|Name|Description

|Starknet CLI | Support for the Starknet CLI has been removed. Instead use
xref:cli:starkli.adoc[Starkli].

Support for Starknet CLI is removed in Starknet v0.13.0.

|Cairo 0 | xref:starknet-versions:version-notes.adoc#version0.11.0[Starknet v0.11.0]
introduces Cairo 1.0 smart contracts.

|===

== Unsupported and removed features

[cols="1,3"]

|===

|Name|Description

| Goerli testnet

a| Goerli testnet support was removed April 2, 2024. Sepolia testnet replaces Goerli
testnet.

Starknet started migrating to Sepolia testnet on November 15th, 2023. For more
information on the Goerli deprecation, see [https://ethereum.org/nb/developers/docs/
networks/#ethereum-testnets](https://ethereum.org/nb/developers/docs/networks/#ethereum-testnets)[the deprecation announcement on Ethereum's site].

Full nodes, API services, SDKs, and other Starknet developer tools have migrated to
Sepolia as well.

[NOTE]

=====

Sepolia's state and history are relatively small. Sepolia xref:version-

notes.adoc[supports declaring classes of CairoZero and Cairo v2.0.0 and higher].

====

| Starknet feeder gateway a| The Starknet feeder gateway, a temporary solution for querying the sequencer's state, is being replaced by Starknet full nodes (Pathfinder, Juno, Deoxys, Papyrus) and RPC services. For more information, see xref:tools:api-services.adoc[Full nodes and API services].

Support for the feeder gateway queries that are not required for full nodes to synchronize on the state of Starknet will stop according to the following schedule:

[%autowidth.stretch]

!===

!Environment !Date

!Integration

!1 November 2023

!Testnet

!15 November 2023

!Mainnet

!19 December 2023

!===

Queries that are required for full nodes to synchronize on the state of Starknet are still supported.

For more information, see the Community Forum post link:<https://community.starknet.io/t/feeder-gateway-deprecation/100233>[_Feeder Gateway Deprecation_].

// | Goerli testnet 2 | Goerli testnet 2 is removed. Use Goerli testnet.

|Free L1-> L2 messaging |Previously, sending a message from L1 to L2 had an optional fee associated.

From xref:starknet-versions:version-notes.adoc#version0.11.0[Starknet v0.11.0], the fee mechanism is enforced and the ability to send L1->L2 messages without the corresponding L2 fee has been removed.

See xref:architecture-and-concepts:network-architecture/messaging-mechanism.adoc#l1-l2-message-fees[here] for more details.

|`invoke` transaction v0 |`invoke` transaction v0 has been removed since xref:starknet-versions:version-notes.adoc#version0.11.0[Starknet v0.11.0].

|`declare` transaction v0 |`declare` transaction v0 has been removed since xref:starknet-versions:version-notes.adoc#version0.11.0[Starknet v0.11.0].

|`deploy` transaction|The `deploy` transaction has been removed since xref:starknet-versions:version-notes.adoc#version0.10.3[Starknet v0.10.3].

To deploy new contract instances, you can use the `xref:architecture-and-concepts:smart-contracts/system-calls-cairo1.adoc#deploy` [`deploy` system call].

|===

[id="juno"]

= Juno release notes

Juno

image::juno_banner.png[width=800]

Juno is a golang Starknet node implementation by <https://nethermind.io/> [Nethermind] with the aim of decentralizing Starknet.

See the official <https://github.com/NethermindEth/juno> [Juno GitHub repository] for more details.

== <https://github.com/NethermindEth/juno/releases/tag/v0.7.3> [v0.7.3]

Support for upcoming Starknet v0.12.3, improved RPC performance with a new global class cache, allowing for higher request throughput and optimized resource usage. We've updated blockifier, which includes an important wallet integration fix. On top of that, expect new metrics and ongoing enhancements to the P2P layer, among other improvements. Here's what's new:

== Added

- * Support for Starknet v0.12.3

- * A global class cache to the VM that enhances overall RPC throughput. +

PR by @omerfirmak in link:<https://github.com/NethermindEth/juno/pull/1401> [1401]

- * A new flag, `+max-vm+`, to control the maximum number of VM instances for concurrent RPC calls, optimizing resource usage +

PR by @omerfirmak in link:<https://github.com/NethermindEth/juno/pull/1378> [#1378].

== Changed

- * The blockifier library has been updated, now supporting the query bit in the version field for transactions. +

PR by @joshklop in link:<https://github.com/NethermindEth/juno/pull/1401> [#1401].

- * Subscription handling has been moved to the synchronizer for improved efficiency. +

PR by @joshklop in link:<https://github.com/NethermindEth/juno/pull/1373> [#1373].

- * Ongoing enhancements to the P2P layer, including the implementation of a Receipt Handler and Adapter and a new `+GetBlockBodies+` feature. +

PRs by @IronGauntlets in link:<https://github.com/NethermindEth/juno/pull/1352> [#1352]

and by @kirugan in link:<https://github.com/NethermindEth/juno/pull/1359> [#1359].

- * Prometheus metrics have been expanded to include version information, latency on `+Transaction.Commit()+`, and read metrics on blockchain operations. +

PRs by @omerfirmak in link:<https://github.com/NethermindEth/juno/pull/1394> [#1394],

link:<https://github.com/NethermindEth/juno/pull/1396> [#1396], and link:<https://github.com/>

NethermindEth/juno/pull/1395[#1395].

* Kubernetes pods now have a correctly set `+GOMAXPROCS+` setting, aligning performance with CPU resources. +

PR by @omerfirmak in link:[#1397](https://github.com/NethermindEth/juno/pull/1397).

* Fallback to feeder traces for blocks `<= 0.12.2`. +

PR by @omerfirmak in link:[#1405](https://github.com/NethermindEth/juno/pull/1405).

== Fixed

* Resolved an issue with event emission on sync step failure for more reliable synchronization. +

PR by @aminsato in link:[#1387](https://github.com/NethermindEth/juno/pull/1387).

* Refactored websocket error conditions for improved stability and error handling. +

PR by @joshklop in link:[#1400](https://github.com/NethermindEth/juno/pull/1400).

== Docker Image

You can pull the Docker image for this release with the following command:

...

```
docker pull nethermind/juno:v0.7.3
```

...

== <https://github.com/NethermindEth/juno/releases/tag/v0.7.0>[v0.7.0]

The primary goal of this release is to introduce support for [link:https://github.com/starkware-libs/starknet-specs/releases/tag/v0.5.0](https://github.com/starkware-libs/starknet-specs/releases/tag/v0.5.0)[Starknet JSON-RPC v0.5.0]. Juno now supports multiple versions via `/v0_5` and `/v0_4` endpoints. The default version at the root `/` endpoint has been updated from 0.4.0 to 0.5.0.

=== Ø<ß Added

* ****Starknet v0.5.0 Compatibility****:

- Implemented ``starknet_specVersion`` @omerfirmak

- Renamed ``juno_getTransactionStatus`` to ``starknet_getTransactionStatus``.

@omerfirmak

- Removed ``pendingTransactions`` endpoint for cleanup. @omerfirmak

- Added new fields like execution resources and message hash to RPC receipt.

@omerfirmak

- Building and calculating state diffs. @omerfirmak

- Make ``starknet_traceBlockTransactions`` get a block id @kirugan

- Add txn type to traces

- Add `message_hash` field for `L1_HANDLER_TXN_RECEIPT`

- Add ``starknet_getTransactionStatus`` and remove ``starknet_pendingTransactions``

* ****Support multiple RPC versions****: [link:https://github.com/starkware-libs/starknet-specs/releases/tag/v0.4.0](https://github.com/starkware-libs/starknet-specs/releases/tag/v0.4.0)[v0.4.0] and [link:https://github.com/starkware-libs/starknet-specs/releases/tag/v0.5.0](https://github.com/starkware-libs/starknet-specs/releases/tag/v0.5.0)[v0.5.0] @omerfirmak

* ****Performance Metrics****: Moved metric counting out of various components for

cleaner code. @omerfirmak

* **Websocket Enhancements**: Full-duplex comms and fixes related to over-reading websocket requests. @joshklop

=== Ø=Ý Changed

* **RPC Optimization**: Reduced allocations in RPC requests for better performance. @joshklop

* **Refactored Error Handling**: Improved global error usage and better error handling in various components. @omerfirmak

=== Ø=Þà Fixed

* **Websocket Reading**: Fixed over-reading issues in Websocket requests. @joshklop

* **Error Handling**: Resolved potential nil pointer dereferences and panic issues. @omerfirmak

=== Ø=Þ€ Deployment and CI/CD

* **Various CI/CD pipeline improvements** for better automation. @wojciechos and @ToluwalopeAyo

=== &™Þ Docker Image

You can pull the Docker image for this release with:

...

docker pull nethermind/juno:v0.7.0

...

== <https://github.com/NethermindEth/juno/releases/tag/v0.6.0>[v0.6.0]

=== Ø<ß Added

* **New Trace RPC Methods**:

- `starknet_traceTransaction`
- `starknet_traceBlockTransactions`
- `starknet_simulateTransactions`

* **Juno RPC Schema**: A dedicated schema to streamline RPC interactions for Juno's method.

* **Juno Console Enhancement**: Pretty printing of Juno console logs for an enriched user experience.

* **Comprehensive Documentation**: Official documentation now hosted on [**https://juno.nethermind.io/\[GitHub Pages\]**](https://juno.nethermind.io/).

=== Ø=Þà Fixed

* **RPC Schema Consistency**:

 Revised to ensure our RPC schema is consistent with the Starknet specification.

=== &™p Command-line Switches Update

Command-line switches have been restructured to provide clearer access control:

...

```
docker run -d
--name juno
-p $httpPort:$httpPort
-p $metricsPort:$metricsPort
-v /root/juno:/var/lib/juno
nethermind/juno:v0.6.0
--db-path /var/lib/juno
--http
--http-port $httpPort
--metrics
--metrics-port $metricsPort
--eth-node <YOUR-ETH-NODE>
...
```

(Note: Ensure to adjust the variables like ` \$httpPort ` , ` \$metricsPort ` and others as per your configuration.)

=== Ø=Ý Migration Notes

* **Database Migration**:

 This version introduces database changes due to our work focus on peer-to-peer (p2p) communication. These changes may result in extended migration times. For faster sync, we recommend users to utilize snapshots.

== <https://github.com/NethermindEth/juno/releases/tag/v0.5.1>[v0.5.1]

This release adds support for the Starknet v0.12.2.

=== Added

* Support for Starknet v0.12.2

== <https://github.com/NethermindEth/juno/releases/tag/v0.5.0>[v0.5.0]

This release adds support for the upcoming Starknet v0.12.1 upgrade and includes compatibility with v0.4.0 of the RPC specification.

=== Added

- * Support for Starknet v0.12.1
- * Compatibility with v0.4.0 of the RPC specification
- * New RPC method: `starknet_estimateMessageFee`
- * Health Check Endpoint: A GET request to the / endpoint will now return a 200 status code for a healthy Juno node
- * Added Prometheus metrics support: Use `--metrics` and `--metrics-port` to enable this feature

=== Changed

- * Adjusted worker number for sync process, improving performance
- * Updated blockifier for starknet v0.12.1

=== Fixed

- * Resolved issues causing context canceled errors in writing RPC methods
- * Mapped gateway errors to write API RPC errors, improving error handling

== <https://github.com/NethermindEth/juno/releases/tag/v0.4.1>[v0.4.1]

== Added

- * Log the incoming RPC requests in <https://github.com/NethermindEth/juno/pull/907>[Pull Request #907]

== Changed

- * Update types for 0.12.1 in <https://github.com/NethermindEth/juno/pull/895>[Pull Request #895]
- * Parallelize per-contract storage updates in <https://github.com/NethermindEth/juno/pull/900>[Pull Request #900]

== Fixed

- * Add missing From field to rpc.MsgToL1 in <https://github.com/NethermindEth/juno/pull/908>[Pull Request #908]

== <https://github.com/NethermindEth/juno/releases/tag/v0.4.0>[v0.4.0]

WARNING: This release has breaking changes and database is not compatible with the previous version.

=== Added

- * **New RPC Methods**:
- ** `starknet_call`

- ** `starknet_estimateFee`
- ** `starknet_addDeclareTransaction`
- ** `starknet_addDeployAccountTransaction`
- ** `starknet_addInvokeTransaction`
- ** `juno_getTransactionStatus`
- ** `juno_version`
- * **L1 Verifier**: Verification of state from Layer 1 has been implemented.
- * **Block Reorg Detection and Handling**: A feature to detect and handle block reorganizations has been implemented.
- * **gRPC Service**: To accommodate users requiring direct access to the database, a gRPC service has been exposed.
- * **Database Migration**: The system has been improved to handle database changes more gracefully. It's no longer necessary to sync from the start when some database changes occur.
- * **Starknet v0.12.0 support**: includes integration with the Rust VM.

=== Changed

- * **Performance Enhancements**: Several adjustments and improvements have been made to increase the performance. These changes have resulted in ~30% reduction in sync time.

== <https://github.com/NethermindEth/juno/releases/tag/v0.3.1>[v0.3.1]

=== Added

- * Fetch and store compiled classes for each Sierra class.

=== Changed

- * Updated the behavior of synced nodes, which will now return false to `starknet_syncing`.

=== Fixed

- * Resolved issue with `NumAsHex(0)` being omitted in RPC.
- * Fixed a Goerli sync issue by relaxing decoder max array elements limit.

Full Changelog: <https://github.com/NethermindEth/juno/compare/v0.3.0...v0.3.1>[[v0.3.0...v0.3.1]]

== <https://github.com/NethermindEth/juno/releases/tag/v0.3.0>[v0.3.0]

=== Added

- * Starknet v0.11.2 support
- * History for contracts, nonce, and class hash.
- * Implemented StateSnapshot.
- * New RPC endpoints:
 - ** `starknet_syncing`
 - ** `starknet_getNonce`

```
** `starknet_getStorageAt`  
** `starknet_getClassHashAt`  
** `starknet_getClass`  
** `starknet_getClassAt`  
** `starknet_getEvents`
```

NOTE: For new RPC endpoints to fully work with data before the new version, the node needs to be resynced.

=== Changed

- * Optimized TransactionStorage encoding and refactored memStorage.
- * Refactored RPC implementation for better organization and maintainability.
- * Parallelized and refactored sync tests for faster execution, improved readability, and maintainability.

=== Fixed

- * Updated handling of non-existent keys to return a zero value.

****Full Changelog****: [https://github.com/NethermindEth/juno/compare/v0.3.0...v0.3.1\[v0.3.0...v0.3.1\]](https://github.com/NethermindEth/juno/compare/v0.3.0...v0.3.1[v0.3.0...v0.3.1])

== [https://github.com/NethermindEth/juno/releases/tag/v0.2.2\[v0.2.2\]](https://github.com/NethermindEth/juno/releases/tag/v0.2.2[v0.2.2])

This patch release fixes handling of block versioning and ensures compatibility with non-sem-ver compliant Starknet.

=== Fixed

- * Ignore or add digits to block version string as necessary.

****Full Changelog****: [https://github.com/NethermindEth/juno/compare/v0.2.1...v0.2.2\[v0.2.1...v0.2.2\]](https://github.com/NethermindEth/juno/compare/v0.2.1...v0.2.2[v0.2.1...v0.2.2])

== [https://github.com/NethermindEth/juno/releases/tag/v0.2.1\[v0.2.1\]](https://github.com/NethermindEth/juno/releases/tag/v0.2.1[v0.2.1])

This minor release introduces an important optimization that enhances sync performance.

- * Update gnark-crypto version:
- ** Implement precomputed point multiplication results for Pedersen hash operations.

****Full Changelog****: [https://github.com/NethermindEth/juno/compare/v0.2.0...v0.2.1\[v0.2.0...v0.2.1\]](https://github.com/NethermindEth/juno/compare/v0.2.0...v0.2.1[v0.2.0...v0.2.1])

== [https://github.com/NethermindEth/juno/releases/tag/v0.2.0\[v0.2.0\]](https://github.com/NethermindEth/juno/releases/tag/v0.2.0[v0.2.0])

This release adds support for Starknet `v0.11.0`.

WARNING: This release has breaking changes and database is not compatible with the previous version.

=== Added

- * Starknet `v0.11.0` support:
- ** Add Poseidon hash for new state commitment.
- ** Add `DeclareTransaction` version 2.
- ** Add and Store Cairo 1/Sierra class definition and hash calculations.
- * `pprof` option is added for profiling and monitoring.
- * Verify Class Hashes.

=== Changed

- * Starknet `v0.11.0` support:
- ** Update `InvokeTransaction` version 1's `contract address` to `sender address`.
- ** Update current JSON RPC endpoints to [v0.3.0-rc1](https://github.com/starkware-libs/starknet-specs/tree/v0.3.0-rc1).
- * Rename the `verbosity` option to `log-level` and `log-level` accepts `string` instead of `uint8`. See `help` for details.
- * `network` option accepts `string` instead of `uint8`. See `help` for details.
- * Database table is updated to account for Starknet `v0.11.0` changes.

=== Removed

- * Remove `metrics` and `eth-node` options since they are not used.

=== Fixed

- * Graceful shutdown: ensure all services have returned before exiting.

Full Changelog: [https://github.com/NethermindEth/juno/compare/v0.1.0...v0.2.0\[v0.1.0...v0.2.0\]](https://github.com/NethermindEth/juno/compare/v0.1.0...v0.2.0[v0.1.0...v0.2.0])

== [https://github.com/NethermindEth/juno/releases/tag/v0.1.0\[v0.1.0\]](https://github.com/NethermindEth/juno/releases/tag/v0.1.0[v0.1.0])

This is Juno's first release (compatible with Starknet `v0.10.3`) with the following features:

- * Starknet state construction and storage using a path-based Merkle Patricia trie.
- * Pedersen and `starknet_keccak` hash implementation over starknet field.
- * Feeder gateway synchronization of Blocks, Transactions, Receipts, State Updates and Classes.
- * Block and Transaction hash verification.
- * JSON-RPC Endpoints:
 - ** `starknet_chainId`
 - ** `starknet_blockNumber`
 - ** `starknet_blockHashAndNumber`
 - ** `starknet_getBlockWithTxHashes`

```
** `starknet_getBlockWithTxn`  
** `starknet_getTransactionByHash`  
** `starknet_getTransactionReceipt`  
** `starknet_getBlockTransactionCount`  
** `starknet_getTransactionByBlockIdAndIndex`  
** `starknet_getStateUpdate`  
[id="pathfinder"]  
= Pathfinder release notes  
# Pathfinder
```

Pathfinder is a Starknet full node giving you a safe view into Starknet.

It provides the following features:

- * Access the full Starknet state history
- * Verifies state using Ethereum
- * Implements the Starknet JSON-RPC API
- * Run Starknet functions without requiring a Starknet transaction
- * Ability to do fee estimation for transactions

See the official <https://github.com/eqlabs/pathfinder> [Pathfinder GitHub repository] for more details.

[0.6.6] - 2023-07-10 (latest)

Fixed

- stack overflow while compiling Sierra to CASM

[0.6.5] - 2023-07-07

Fixed

- pending data from the gateway is inconsistent
 - this could exhibit as RPC data changing status between `pending | L2 accepted | not found`, especially noticeable for transactions.

Changed

- substantially increase the character limit of execution errors
 - previously, the RPC would return a highly truncated error message from the execution vm

[0.6.4] - 2023-07-05

Fixed

- Pending data is not polled for starknet v0.12 due to an HTTP error code change from the gateway.
- Transaction receipts missing `from_address` in `MSG_TO_L1`.

[0.6.3] - 2023-06-29

Fixed

- Sierra class hash not in declared classes sync bug

Changed

- use all libfunc list instead of experimental for sierra compilation

[0.6.2] - 2023-06-29

Added

- `starknet_estimateMessageFee` for JSON-RPC v0.3.1 to estimate message fee from L1 handler.
- sync-related metrics
 - `current_block`: the currently sync'd block height of the node
 - `highest_block`: the height of the blockchain
 - `block_time`: timestamp difference between the current block and its parent
 - `block_latency`: delay between current block being published and sync'd locally
 - `block_download`: time taken to download current block's data excluding classes
 - `block_processing`: time taken to process and store the current block
- configuration for new block polling interval: `--sync.poll-interval <seconds>`
- Starknet v0.12.0 support
 - sierra v2.0.0 support
 - `cairo-lang` upgraded to 0.12.0a0

Fixed

- reorgs fail if a class declaration is included in the reorg
- sync can fail if db connection pool is held saturated by rpc queries
- uses `finalized` (reorg-safe) L1 state instead of `latest`
- `starknet_getEvents` times out for queries involving a large block range

Changed

- dropped upgrade support for pathfinder v0.4 and earlier
- separate db connection pools rpc, sync and storage
- increased the number of rpc db connections

[0.6.1] - 2023-06-18

Fixed

- class hash mismatch for cairo 0 classes with non-ascii text

[0.6.0] - 2023-06-14

Fixed

- ``starknet_simulateTransaction`` requires ``transactions`` instead of ``transaction`` as input field.
- gateway's error message is hidden when submitting a failed transaction
- ``starknet_getEvents`` is very slow for certain filter combinations

Changed

- default RPC API version changed from v0.2 to v0.3
- disallow JSON-RPC notification-style requests

[0.5.6] - 2023-05-25

Added

- Starknet v0.11.2 support
 - Sierra compiler v1.1.0-rc0
 - ``cairo-lang`` upgraded to 0.11.2a0
- Subscription to ``newHead`` events via websocket using the method ``pathfinder_subscribe_newHeads``, which can be managed by the following command line options
 - ``rpc.websocket``, which enables websocket transport
 - ``rpc.websocket.capacity``, which sets the maximum number of websocket subscriptions per subscription type

Authors: [Shramee Srivastav](<https://github.com/shramee>) and [Matthieu Auger](<https://github.com/matthieuauger>)

[0.5.5] - 2023-05-18

Added

- ``cairo-lang`` upgraded to 0.11.1.1

Fixed

- RPC emits connection logs and warnings

- Fee estimate mismatch between gateway and pathfinder
 - Gateway uses a new gas price sampling algorithm which was incompatible with pathfinders.
- Fee estimate returns error when submitting Cairo 1.0.0-rc0 classes.
- Historic L1 handler transactions are served as Invoke V0
 - Older databases contain L1 handler transactions from before L1 handler was a specific transaction type. These were stored as Invoke V0. These are now correctly identified as being L1 Handler transactions.

Fixed

- RPC emits connection logs and warnings
- Fee estimate mismatch between gateway and pathfinder
 - Gateway uses a new gas price sampling algorithm which was incompatible with pathfinders.
- Historic L1 handler transactions are served as Invoke V0
 - Older databases contain L1 handler transactions from before L1 handler was a specific transaction type. These were stored as Invoke V0. These are now correctly identified as being L1 Handler transactions.

v0.5.4

The primary focus of this release is to provide support for Starknet v0.11.1, and will continue to work for v0.11.0. Since this release is required for v0.11.1, you should update your node before the network is updated.

Added

- Starknet v0.11.1 support
- CORS support via the `rpc.cors-domains` configuration option
- Transaction hashes are now verified as part of the sync process. Previously, these were not verified as the exact algorithm was underdocumented and the transaction format was still evolving.

Fixed

- RPC server panic for unprefixed unregistered method names
- Data can temporarily appear to go missing when transitioning from `PENDING` to `ACCEPTED ON L2`
 - This was commonly seen when rapidly monitoring a new transaction, which would go from `PENDING` to `TXN_HASH_NOT_FOUND` to `ACCEPTED_ON_L2` as pathfinder moved the ephemeral pending data to latest data on disk.

v0.5.3

Fixes for minor issues and inconsistencies.

Added

- ``max-rpc-connections`` command-line argument. This sets the maximum number incoming RPC connections the pathfinder node will accept. This defaults to 1024 if not specified.
- ``cairo-lang`` upgraded to 0.11.0.2

Fixed

- ``starknet_simulateTransaction`` data model inconsistency
- ``poll-pending`` default value restored to ``false``
- incoming RPC connections limited to 100. This limit was accidentally introduced in v0.5.2 as part of a dependency upgrade, whereas before it was unlimited. The default is now 1024 and can be configured using ``--max-rpc-connections``.
- handling of invalid JSON-RPC requests

v0.5.2

This release fixes a few RPC bugs and adds support for bulk fee estimation and transaction simulation (traces) as part of v0.3 RPC specification.

In addition it also adds a ``pathfinder_getTransactionStatus`` endpoint which lets you track a transactions status -- including ``REJECTED`` and ``RECEIVED`` -- in the same fashion as the gateway.

Added

- support ``starknet_estimateFee`` in the JSON-RPC v0.3 API
 - supports estimating multiple transactions
 - this includes declaring and immediately using a class (not currently possible via the gateway)
- support ``starknet_simulateTransaction`` for JSON-RPC v0.3
 - supports simulating multiple transactions
 - this includes declaring and immediately using a class (not currently possible via the gateway)
- support ``pathfinder_getTransactionStatus`` which is exposed on all RPC routes
 - this enables querying a transactions current status, including whether the gateway has received or rejected it

Fixed

- RPC returns int for entrypoint offsets instead of hex
- RPC rejects Fee values with more than 32 digits

- RPC does not expose `pathfinder_getProof` on v0.3 route

v0.5.1

This is a minor bugfix release, primarily to fix an issue with syncing on `testnet2`.

Fixed

- * pathfinder sometimes spams nethermind L1 nodes
- * pathfinder stops syncing `testnet2` at block 95220 due to a Sierra class compilation issue

v0.5.0

Highlights

- starknet v0.11.0 support
- RPC API v0.3 partial support
- removed several deprecated config options
- requires python 3.9 or 3.10 (no longer 3.8)

Added

- support for state commitment and class commitment in `pathfinder_getProof`
- support for starknet v0.11
- partial support for RPC specification v0.3
 - exposed on `/rpc/v0.3/` route
 - missing support for `starknet_estimateFee` and `starknet_simulate`

Changed

- `starknet_call` and `starknet_estimateFee` JSON-RPC methods return more detailed error messages
- `python` version requirement has changed to `3.9` or `3.10` (was `3.8` or `3.9` previously)

Fixed

- RPC accepts hex inputs for Felt without '0x' prefix. This led to confusion especially when passing in a decimal string which would get silently interpreted as hex.
- using a Nethermind Ethereum endpoint occasionally causes errors such as `- sync can miss new block events by getting stuck waiting for pending data.

Removed

- `--config`` configuration option (deprecated in [v0.4.1](https://github.com/eqlabs/pathfinder/releases/tag/v0.4.1))
- `--integration`` configuration option (deprecated in [v0.4.1](https://github.com/eqlabs/pathfinder/releases/tag/v0.4.1))
- `--sequencer-url`` configuration option (deprecated in [v0.4.1](https://github.com/eqlabs/pathfinder/releases/tag/v0.4.1))
- `--testnet2`` configuration option (deprecated in [v0.4.1](https://github.com/eqlabs/pathfinder/releases/tag/v0.4.1))
- `starknet_addDeployTransaction`` as this is no longer an allowed transaction since starknet v0.10.3
- RPC api version `0.1``, which used to be served on path `/rpc/v0.1``

RPC API

We added support for v0.3 and removed v0.1. We still support v0.2 at both `/rpc/v0.2`` and `/rpc`` (default) routes. In summary:

```

/           # serves v0.2
/rpc/v0.2/  # serves v0.2
/rpc/v0.3/  # serves v0.3

```

We are missing `starknet_estimateFee`` and `starknet_simulate`` support for v0.3, which will be added in an upcoming release.

Python requirement

Note: this only applies if you are building from source. This does not impact docker users.

Pathfinder requires python to support the starknet VM used to simulate starknet transactions and function calls. Previous versions of the VM only worked with python 3.8 or 3.9 which was a hassle because most operating systems no longer directly support it. The new version of the VM bundled with starknet v0.11 now requires python version 3.9 or 3.10.

Configuration changes

Several configuration options are now removed, after they were deprecated in pathfinder v0.4.1. Here is a migration guide:

- `--testnet2``: use `--network testnet2`` instead
- `--integration``: use `--network integration`` instead
- `--sequencer-url``: use `--network custom`` in combination with `--feeder-gateway-url`` and `gateway-url``
- `--config``: use environment variables or env files as an alternative

v0.4.5

Hotfix for a bug introduced in the previous version v0.4.4, which prevented a new node from syncing on blocks near genesis.

Added

Added Newton FAQ links to readme

Fixed

Node fails to sync old blocks

New contributors

@SecurityQQ made their first contribution in #799

v0.4.4

This minor release contains some nice performance improvements for ``starknet_call`` and ``starknet_estimateFee`` as well as some minor bug fixes.

Also included is a major new feature: storage proofs - big thanks @pscott for his hard work on this feature! This is available via the ``pathfinder_getProof`` method which is served from both the pathfinder and Starknet endpoints for convenience:

[source]

<node-url>/rpc/pathfinder/v0.1/pathfinder_getProof

<node-url>/rpc/v0.2/pathfinder_getProof

The method is specified https://github.com/eqlabs/pathfinder/blob/main/pathfinder_rpc_api.json#L22-L113[here].

Its results can be used to formally verify what a contract's storage values are without trusting the pathfinder node.

This is achieved by validating the merkle-proof that pathfinder returns and confirming that it correctly matches the known Starknet state root.

Added

storage proofs via ``pathfinder_getProof`` by @pscott

Fixed

* ``starknet_getEvents`` returns all events when `from_block="latest"`

* v0.1 ``starknet_getStateUpdate`` does not contain nonces

Changed

* Improved performance for ``starknet_call`` and ``starknet_estimateFee`` by caching classes

* Improved performance for ``starknet_call`` and ``starknet_estimateFee`` by using Rust for hashing

New contributors

@pscott made their first contribution in #726

v0.4.3

The primary purpose of this release is to properly support testnet2 after the [xref:starknet-versions:version-notes.adoc#version0.10.3](#)[Starknet v0.10.3] update.

The v0.10.3 update changed the testnet2 chain ID which impacts transaction signatures which in turn meant that ``starknet_estimateFee`` would fail for any signed transaction.

This release updates pathfinder to use the correct chain ID.

Fixed

- * Testnet2 and integration flags are ignored
- * ``starknet_estimateFee`` uses wrong chain ID for testnet2

Changed

Updated to cairo-lang 0.10.3

v0.4.2

Contains several bug fixes, mostly hotfixes for bugs introduced in v0.4.1.

Added

Document that ``--chain-id`` expects text as input

Fixed

- * Testnet2 and integration L1 addresses are swapped (bug introduced in v0.4.1)
- * Proxy network setups can't sync historical blocks (bug introduced in v0.4.1)
- * ABI serialization for ``starknet_estimateFee`` for declare transactions

v0.4.1

Highlights

- * Soft deprecation of some configuration options
- * Support custom Starknet gateways
- * Pathfinder RPC extensions at ``/rpc/pathfinder/`` with ``pathfinder_version`` method
- * ``starknet_events`` optimisations
- * fix block timestamp in pending calls
- * Custom Starknet gateway support

This release introduces support for custom Starknets. You can select this network by setting ``--network`` custom and specifying the ``--gateway-url`` and ``--feeder-gateway-url``

options.

Configuration option deprecation

Several configuration options have been soft deprecated. This means using them will continue to work as before (no breaking change), but they will emit a warning when used. They will be removed in a future version, so please migrate to the newer options.

To re-emphasize: your current configuration setup will continue to work as is.

Network selection

`--testnet2`` and `--integration`` have been deprecated in favor of `--network` testnet2`` and `--network` integration``.

Gateway proxy

`--sequencer-url`` has been deprecated in favor of `--network` custom`` along with `--gateway-url``, `--feeder-gateway-url`` and `--chain-id``. In addition, you will need to rename your existing database file to `custom.sqlite` as this will be the expected filename for custom networks.

Configuration file

`--config`` has been deprecated and will not be supported in the future. The utility this provided was valuable. Unfortunately it is starting to severely hinder how fast we can implement configuration changes and we decided to remove it.

We suggest using environment variables along with environment files to configure pathfinder in a similar fashion.

Changed

The following configuration options are now marked as deprecated: `--testnet2``, `--integration``, `--config``, `--sequencer-url``

Optimized starknet_events for queries with both a block range and a from address

Fixed

Block timestamps for pending in `starknet_call`` and `starknet_estimateFee`` were using the latest timestamp instead of the pending one. This meant contracts relying on accurate timestamps could sometimes fail unexpectedly.

Added

- * Custom Starknet support

- * Pathfinder specific RPC extensions hosted at `<rpc-url>/rpc/pathfinder/v0.1``.

Currently, this only contains `pathfinder_version`` which returns the pathfinder version of the node.

v0.4.0- (breaking release)

This release contains a breaking change, and also adds support for xref:starknet-

versions:version-notes.adoc#version0.10.2[Starknet v0.10.2].

The changes themselves are quite simple, but please read through each section as there are some caveats which might impact you when you apply this update.

Default RPC version change

This release changes the version of the RPC that is served at the root route, from v0.1 to v0.2 of the RPC specification. Version v0.1 is still available at the `/rpc/v0.1/` endpoint. This is the only breaking change in this release.

Here is a summary of what routes are currently available, and what's changed:

- * `/` serves v0.2 (changed from v0.1)
- * `/rpc/v0.1/` serves v0.1 (no change)
- * `/rpc/v0.2/` serves v0.2 (no change)

If possible, we recommend that you use the version specific routes as this will prevent such breaking changes from impacting you.

Starknet v0.10.2 support

This release includes an update to the cairo-vm embedded in pathfinder in order to support the upcoming v0.10.2 Starknet release. This bundled vm is a pre-release and may therefore contain differences to the final version used once Starknet updates testnet and mainnet. We will of course issue a new release if / when there is a new vm.

[NOTE]

====

Since these changes are not yet live on testnet nor mainnet, this means upgrading to this release will cause deviations between what pathfinder outputs and what can be expected on the network. More specifically, `starknet_estimateFee` will compute different fees until the network has upgraded to xref:starknet-versions:version-notes.adoc#version0.10.2[Starknet v0.10.2].

If you don't need the RPC route changes, it may be pertinent to delay updating until closer to the xref:starknet-versions:version-notes.adoc#version0.10.2[v0.10.2] release dates on testnet and mainnet. The expected timeline for these upgrades is ~17/11 for testnets and ~24/11 for mainnet.

====

[id="upcoming_versions"]

= Upcoming Starknet versions

[id="what_to_expect"]

== Roadmap and version updates

You can subscribe to get the latest version updates delivered to your inbox at link:<https://www.starknet.io/en/roadmap>[Starknet Roadmap & version updates].

For information on the current version of Starknet, see the `xref:version-notes.adoc`[Starknet release notes].[id="upcoming"]
= Starknet release notes

The following release notes cover the ongoing version changes to Starknet. You can subscribe to get the latest version updates delivered to your inbox at [link:https://www.starknet.io/en/roadmap](https://www.starknet.io/en/roadmap)[Starknet Roadmap & version updates].

== Starknet environments

Within Starknet's deployment pipeline, there are separate and distinct networks that operate independently of each other for testing before deployment.

include::ROOT:partial\$snippet_important_goerli_removed.adoc[]

.Current versions supported in each environment

[%autowidth.stretch]

|===

|Environment |Starknet version|Sierra version|Cairo version

|Mainnet|0.13.2|1.6.0|2.0.0 - 2.7.1

|Sepolia Testnet|0.13.2|1.6.0|2.0.0 - 2.7.1

|===

[id="version0.13.2"]

== Starknet v0.13.2 (August 28, 24)

[discrete]

=== New features and enhancements

* Optimistic parallelization in the sequencer.

* Applicative recursion ("blockpacking"): pack a contiguous sequence of blocks into a single L1 state update, instead of just one block. This change also affects the Starknet core contract on L1 and the structure of data availability:

** The contract will store also the hash of the "aggregator" program, alongside the hash of the Starknet OS.

** The [link:https://github.com/starkware-libs/cairo-lang/blob/efa9648f57568aad8f8a13fbf027d2de7c63c2c0/src/starkware/starknet/solidity/](https://github.com/starkware-libs/cairo-lang/blob/efa9648f57568aad8f8a13fbf027d2de7c63c2c0/src/starkware/starknet/solidity/)

`Starknet.sol#L36[LogStateUpdate]` event will be emitted once for every sequence of blocks in an applicative tree, rather than for every block.

** The data posted on L1 is now the output of the aggregator, instead of the OS. The

[link:https://github.com/starkware-libs/cairo-lang/blob/efa9648f57568aad8f8a13fbf027d2de7c63c2c0/src/starkware/starknet/core/os/](https://github.com/starkware-libs/cairo-lang/blob/efa9648f57568aad8f8a13fbf027d2de7c63c2c0/src/starkware/starknet/core/os/)

`output.cairo#L25[OsOutputHeader]` struct will contain four new fields:

``prev_block_number`, `prev_block_hash`, `os_program_hash`, `full_output`.`

* New block hash definition: see the reference implementation link:https://github.com/starkware-libs/sequencer/blob/b6955bbd59635d37a12f9070a3f0b8a8db74f7c1/crates/starknet_api/src/block_hash/block_hash_calculator.rs#L67[here].

[discrete]

=== API changes

==== Feeder gateway

* All objects containing the block hash will use the new block hash computation.

* Receipts will contain a new property ``total_gas_consumed``, only for transactions after v0.13.2.

* Three new builtins can appear in the ``builtin_instance_counter`` property of ``execution_resources`` in the transaction receipts: ``add_mod``, ``mul_mod`` and ``range_check96``.

* End support for the endpoints ``get_block_traces`` and ``get_transaction_trace``.

==== JSON RPC

No changes in v0.13.2. The new JSON RPC version v0.8.0 will be released alongside Starknet v0.13.3.

[discrete]

=== Cairo

A new compiler version will be released for v0.13.2, Cairo v2.7.0. This includes a Sierra upgrade to v1.6.0, i.e. contracts compiled with the new compiler will only be accepted on Starknet v0.13.2 onwards.

The Starknet-related features that will be added in this Cairo version include:

* sha256 syscall - syscall for computing sha256 on an arbitrary length input:

** link:<https://github.com/starkware-libs/cairo/blob/f5ac2d1d24ae0a626e9925db0c564bd0c4fea433/corelib/src/sha256.cairo#L24>[High level code] for using sha256/

** link:https://github.com/starkware-libs/blockifier/blob/b22fb076a7db5e0fcdd2048a6fb579b0b1d25561/crates/blockifier/resources/versioned_constants.json#L284[Syscall cost] - the dominant part of the syscall is ~1.1k bitwise builtin applications which today costs ~180 L1 gas (the 2k steps are negligible in comparison). The syscall is applied once for ~14 ``u32``.

* Circuit builtin - the new compiler version will introduce a way to define ad-hoc

algebraic circuits in Cairo. Circuits use the new `mul_mod` and `add_mod` builtins under the hood.

** Example usage can be found in link:https://github.com/starkware-libs/cairo/blob/39095a2a717b5bf3a76c813186f0a9cd0e087948/corelib/src/test/circuit_test.cairo#L24[circuit_test.cairo].

[discrete]
=== Others

Errors prettifying: execution errors are becoming more link:https://github.com/starkware-libs/blockifier/blob/8417325e6990af64e93253b1f76cb70611516cd2/crates/blockifier/src/execution/syscalls/hint_processor.rs#L69[structured], which will be the basis for better error handling in the next JSON RPC version, resulting in nice error displays by wallets.

[discrete]
=== Additional resources

Community Forum Posts:

- * link:<https://community.starknet.io/t/starknet-v0-13-2-pre-release-notes/114223>[Starknet v0.13.2 pre-release notes]
- * link:<https://community.starknet.io/t/optimistic-parallelization-revived/114121>[Optimistic parallelization revived]
- * link:<https://community.starknet.io/t/upcoming-feature-starknet-applicative-recursion/113868>[Upcoming Feature: Starknet Applicative Recursion]
- * link:<https://community.starknet.io/t/cairo-v2-7-0-is-coming/114362>[Cairo v2.7.0 is coming!]

[id="version0.13.1.1"]
== Starknet v0.13.1.1 (April 15, 24)

In response to community feedback, Starknet 0.13.1.1 reduces class declaration fees and increases the calldata limit:

|===
| Resource | Gas cost (0.13.1) | Gas cost (0.13.1.1)

| CASM bytecode | 28 gas/felt | 1 gas/felt
| Sierra bytecode | 28 gas/felt | 1 gas/felt
| ABI | 0.875 gas/character | 0.032 gas/character
|===

|===
|Entity | Limit (0.13.1) | Limit (0.13.1.1)

| Calldata length (felts) | 4,000 | 5,000
|===

[id="version0.13.1"]
== Starknet v0.13.1 (March 13, 24)

[discrete]
=== New features and enhancements

* *Cheaper data availability (DA):* Starknet uses link:<https://eips.ethereum.org/EIPS/eip-4844>[EIP-4844]. State diffs are now blobs, rather than calldata, requiring the addition of ``data_gas`` to the Starknet block header.

* Time-related syscalls when called from an account contract's ``+__validate__+``, ``+__validate_deploy__+``, ``+__validate_declare__+``, or ``constructor`` function:

** ``block_timestamp`` returns the hour, rounded down.

** ``block_number`` returns the block number, rounded down to the nearest multiple of 100.

* *Optimization:* Load into memory only the functions in a contract that are actually used when generating the proof.

[discrete]
=== Block header updates
The block header includes the following new fields:

* ``l1_da_mode``: A string enum that takes the value ``CALLDATA`` or ``BLOB``, and indicates whether EIP-4844 is the data availability solution that is used for the block. Also appears in pending block.

* ``l1_data_gas_price``: Contains ``price_in_wei`` and ``price_in_fri``, where 1 fri is 10^{18} STRK. Also appears in pending block.

* ``l1_gas_price``: Replaces ``eth_l1_gas_price`` and ``strk_l1_gas_price``. Contains the data gas price (EIP-4844) in addition to the regular gas price.

[discrete]
=== API: JSON RPC

[discrete]
==== Starknet API JSON RPC 0.6.0

Starknet 0.13.1 is backward compatible with ``starknet_api_openrpc.json`` v0.6.0. Responses from 0.13.1 can be mapped naturally into v0.6.0 objects.

[discrete]
==== Starknet API JSON RPC 0.7.0

A new version of ``starknet_api_openrpc.json``, 0.7.0, accommodates the changes

introduced by Starknet using EIP-4844.

.`BLOCK_HEADER` includes two new fields to support EIP-4844:

- * `l1_data_gas_price`: contains `price_in_wei` and `price_in_fri` (10^6 - 10^{18} denominations, similar to [https://github.com/starkware-libs/starknet-specs/blob/49665932a97f8fdef7ac5869755d2858c5e3a687/api/starknet_api_openrpc.json#L3766\[v0.6.0\]](https://github.com/starkware-libs/starknet-specs/blob/49665932a97f8fdef7ac5869755d2858c5e3a687/api/starknet_api_openrpc.json#L3766[v0.6.0])).

- * `l1_da_mode`: An enum that indicates whether this block will use calldata or blobdata and can take the following values:

 - ** `CALLDATA`

 - ** `BLOB`

.`FEE_ESTIMATE`

- * Includes two new fields:

 - ** `data_gas_consumed`

 - ** `data_gas_price`

- * `overall_fee` is now: +

 - $\text{gas_consumed} \times \text{gas_price} + \text{data_gas_consumed} \times \text{data_gas_price}$

- * Fee estimates will change depending on the data availability solution used by current Starknet blocks. For example, if you estimate the fee against the pending block, and it's currently using `CALLDATA`, then nodes are expected to return `data_gas_consumed=0` and compute the fee similarly to today, that is, get higher estimates.

.Receipts and traces now include data availability resources

- * `COMMON_RECEIPT_PROPERTIES`, the main receipt object, now includes a new entry: `execution_resources`.

- * The `EXECUTION_RESOURCES` object now includes the field `data_availability`. Note that the resources of internal calls will remain the same/

- * For more information, see the <https://github.com/starkware-libs/starknet-specs/pull/187/files> [PR for the API JSON RPC specs]

.`EXECUTION_RESOURCES`

- * Computation resources are separated from data availability resources. This is done by introducing the `data_availability` property, which includes `l1_gas` and `l1_data_gas`, which were consumed due to DA requirements. One of these will always be zero, depending on whether or not the block uses calldata or blobs, as specified by the `l1_da_mode` field in the block header.

- * Syscall costs are now included in the execution resources of traces and receipts. These are costs that are already being paid for but were not reported so far.

[discrete]

=== Pricing changes

[discrete]

==== Computation

- * A Cairo step now costs 0.0025 gas/step, a 50% reduction.
- * All builtins costs are accordingly reduced by 50%.

[discrete]

==== Calldata and signatures

Each felt in the calldata and signature arrays of all transaction types now costs 0.128 gas/felt.

[discrete]

==== Class declaration

- * Each felt of a sierra_program in the contract class and of bytecode in the compiled contract class now costs 28 gas/felt.

+

[NOTE]

====

v1 `DECLARE` transactions only include bytecode.

====

- * Each character in the ABI costs 0.875 gas.

[discrete]

==== Events

- * An additional felt to the data array of an event now costs 0.128 gas/felt, similar to calldata.
- * An additional felt to the keys array now costs 0.256 gas/felt.

[discrete]

=== Infrastructure updates

Starknet now supports multiple L1 providers.

[discrete]

=== Additional resources

Community Forum Posts:

- * link:<https://community.starknet.io/t/starknet-v0-13-1-eip4844-support-more-fee-reductions-stability-quality-of-life/112951>[Starknet v0.13.1: EIP4844 Support, More Fee Reductions, Stability, Quality of Life]
- * link:<https://community.starknet.io/t/starknet-v0-13-1-fee-reduction/113552>[Starknet

v0.13.1: Fee Reduction]

* link:<https://community.starknet.io/t/data-availability-with-eip4844/113065>[Data availability with EIP4844]

* link:<https://community.starknet.io/t/starknet-v0-13-1-pre-release-notes/113664>[Starknet v0.13.1 pre-release notes]

[id="version0.13.0"]

== Starknet v0.13.0 (Jan 10, 24)

Starknet v0.13.0 is live on Mainnet.

Starknet 0.13.0 includes the following changes:

* v3 transactions, including:

** Fee payment in STRK

** Reserved fields for future features, such as Volition and payment master

* ``get_block`` API: The ``gas_price`` field is replaced by the ``eth_l1_gas_price`` and ``strk_l1_gas_price`` fields. This change applies also to existing blocks. For more information on the new fields, see the link:<https://github.com/starkware-libs/starknet-specs/releases/tag/v0.6.0>[JSON RPC API Spec on GitHub]

* Sierra v1.4.0. This new version of Sierra is part of Crate v2.4.0, in the Cairo 2.4.0 package. For more information, see link:<https://community.starknet.io/t/cairo-v2-4-0-is-out/109275>[Cairo v2.4.0 is out!] on the Community forum.

* Improved performance of ``secp256k1_mul`` and ``secp256r1_mul`` syscalls

* Computation cost is reduced by approximately 50% as a result of reduced Cairo steps and increased use of builtins. L1 data availability cost is reduced by approximately 10%-25%. For an ERC-20 transfer, the DA fee reduction is 25%.

[id="version0.12.3"]

== Starknet v0.12.3 (Nov 19, 23)

Starknet v0.12.3 is live on Mainnet.

This release partially removes support for the Starknet feeder gateway. For details, see link:<https://community.starknet.io/t/feeder-gateway-deprecation/100233/1>[Feeder Gateway Deprecation] in Development Proposals on the Starknet community forum.

Additionally, this version includes the following changes:

* Performance optimizations in the gateway, the computation of the Patricia storage root, and block hash

* Support for ``secp256r1`` syscalls in the Starknet OS.

* Restriction for ``+__validate__+`` and the constructor of ``DeployAccount`` transactions:

** Restrict access to ``sequencer_address`` in the ``get_execution_info`` syscall by

returning ``0``'s for the address.

** Restrict access to the following syscalls:

*** Cairo contracts: `get_block_hash`

*** Cairo 0 contracts: `get_sequencer_address`

This version is available on both Goerli and Sepolia testnets.

[discrete]

=== Cairo 0

Move structs that are common to `secp256k1` and `secp256r1` to a separate file.

[id="version0.12.2"]

== Starknet v0.12.2 (Sep 04, 23)

Starknet v0.12.2 is live on Mainnet.

This version includes the following changes:

- * Enabling P2P Authentication: An additional endpoint in the sequencer gateway to provide a signature on the state diff commitment and block hash.

- * Resolving Mismatches in Queries: An extension to the `get_state_update` endpoint in the sequencer gateway that returns both the pending state diff and the pending block together.

- * Increased maximum Cairo steps per transaction from 1 million to 3 million.

[id="version0.12.1"]

== Starknet v0.12.1 (Aug 21, 23)

Starknet v0.12.1 is live on Mainnet.

This version includes the following changes:

- * Mempool Validation.

- * Inclusion of Failed Transactions.

- * Keccak builtin.

[id="version0.12.0"]

== Starknet v0.12.0 (July 12, 23)

Starknet v0.12.0 is live on Mainnet.

This version contains the following changes:

- * Use the link:<https://github.com/starkware-libs/blockifier>[rust blockifier] and link:<https://github.com/starkware-libs/blockifier>

github.com/lambdaclass/cairo-vm[LambdaClass's Cairo VM] to accelerate the sequencer's time to handle transactions.

- * Support link:<https://github.com/starkware-libs/cairo/releases/tag/v2.0.0>[version 2.0.0] of the Cairo compiler.

- * Replace the `PENDING` status of transactions to `ACCEPTED_ON_L2` - once a transaction is in that status it means that it will be included in a block, this applies to transactions - blocks still have the `PENDING` status.

- * Add an experimental `get_block_hash` syscall.

- * Change HTTP error code from 500 to 400 on API errors.

[id="version0.11.2"]

== Starknet v0.11.2 (May 31, 23)

Starknet v0.11.2 is live on Mainnet.

This version contains the following changes:

- * Upgrade Cairo 1.0 version to v1.0.0-rc0 (Cairo 1.0 activated on Starknet!)

[id="version0.11.1"]

== Starknet v0.11.1 (May 23, 23)

Starknet v0.11.1 is live on Mainnet.

This version contains the following changes:

- * Upgrade Cairo 1.0 version to v1.0.0-rc0.

- * Charged transaction fee is now based on an average Ethereum gas price instead of a single sample (estimation API is unaffected).

- * API changes:

- ** Remove the state root in `get_state_update` for pending blocks to allow faster responses in future versions.

- * Testing framework:

- ** Allow declaring (and interacting with) Cairo 1.0 contracts.

- *** Currently, the Cairo 1.0 ABI is not supported yet, so a Cairo 0 ABI should be supplied to `declare()` manually.

- ** Split `deploy()` to two phases declare and deploy: `deprecated_declare()` (for Cairo 0 contract) or

- `declare()` (for Cairo 1.0 contracts) and `deploy()` (for both).

- * Add current block hash to the Starknet Core Contract (currently not verified by the

Starknet OS):

**** Breaking change:** The ``LogStateUpdate`` event's data is changed to include `blockHash`.

[id="version0.11.0"]

== Starknet v0.11.0 (Mar 29, 23)

Starknet v0.11.0 is live on Mainnet.

[NOTE]

====

``invoke`` and ``declare`` transactions of version 0 are no longer supported on this version.

====

In Starknet v0.11.0, you can declare, deploy and run Cairo 1.0 smart contracts. We also introduce a new system call that allows a smooth transitioning of existing contracts to a Cairo 1.0 implementation.

Historically, contract classes have been defined in terms of Cairo assembly, or Casm for short (the class definition also included more information needed for execution, e.g., hint data). The novelty of Cairo 1.0 is the introduction of Sierra (Safe Intermediate Representation), an intermediate layer between Cairo 1.0 and Casm.

The introduction of Cairo 1.0 and Sierra has several effects on the system. Below we list the effects on each component; of particular note are:

- * A new version of the ``declare`` transaction, which allows sending the new class structure
- * The state commitment will now include contract classes
- * Changes to the onchain data format
- * New system call - ``replace_class``

[id="version0.10.3"]

== Starknet v0.10.3 (Dec 12, 22)

[NOTE]

====

The ``deploy`` transaction is no longer supported on this version.

====

This version contains the following changes:

Starknet

- * Performance - Separate the state commitment computation from the execution of the transactions
- * Add `starknet-class-hash` command to compute the class hash of a compiled Starknet contract

Cairo:

- * Autoformatter: Automatically break lines inside expressions

[id="version0.10.2"]
 == Starknet v0.10.2 (Nov 29, 22)

- This version introduces sequencer parallelization! This is the first step in our roadmap of performance upgrades. Details about the specific mechanism of parallelization and the roadmap in general are described in <https://medium.com/starkware/starknet-performance-roadmap-bb7aae14c7de>[this medium post].

- A new endpoint, `estimate_fee_bulk`, is added to the feeder gateway. This will allow estimating the fee of several transactions at once, where each transaction is executed relative to the resulting state from applying the previous one.

- * Sequencing performance improvements
- * Builtin ratio changes, which affects builtin costs
- * Add `estimate_fee_bulk` API that computes the fee of multiple transactions that will be executed consecutively

As part of this version, we will also increase the finality of transactions in the pending block, by fixing the timestamp at the time of the block creation. This will solve the issue of transactions moving from pending to rejected on account of too old timestamp

[id="version0.10.1"]
 == Starknet v0.10.1 (Oct 25, 22)

This version contains the following changes:

Starknet:

- * Add `DeployAccount` transaction (which will replace the Deploy transaction for deploying account contracts). To use it, you should first add enough funds to your account address to pay the transaction fee, and then you can invoke DeployAccount
- * Split the `starknet deploy_account` CLI command into `starknet new_account` and `starknet deploy_account`
- * Account contracts that are expected to be deployed this way should implement the `__validate_deploy__()` entry point, which should check the signature of the `DeployAccount` transaction

- * Improve L1 fee computation: the fee is computed according to the diff of the storage state
- * API: Remove `entry_point_type` field from transaction information

Cairo:

- * Add `uint256_mul_div_mod` to `uint256.cairo`

[id="version0.10.0"]
 == Starknet v0.10.0 (Sept 05, 22)

This version introduces the next step in Starknet's account abstraction design, specifically the validate/execute separation. See <https://www.notion.so/starkware/Starknet-0-10-0-4ac978234c384a30a195ce4070461257>[here] for more information.

This version contains the following changes:

Starknet:

- * Contract (breaking changes):
 - ** @external and @view functions should be imported directly by the main compiled file. Otherwise, they will not be usable as external functions
 - ** Forbid using the same storage variable name in two modules
 - * New transaction version (version 1) for `invoke` and `declare` transactions:
 - ** Transactions of version 0 are deprecated and will not be supported in Starknet from the next version (v0.11.0). Please update your systems to use the new version

[NOTE]

====

In order to use transactions of version 1 you will need to upgrade your account contracts
 =====

- ** Add nonce field to the transactions. Nonce validation is now part of the Starknet protocol and is enforced to be executed sequentially
- ** `Invoke`:
 - *** Split `__execute__` to two functions: `__validate__` (only validates the transaction) and `__execute__` (only executes the transaction)
 - *** Remove the selector (which is now always `__execute__`) field, following the above change.
- ** Declare:
 - *** `declare` transaction should now be sent from an account (and is validated using `__validate_declare__` in the account contract)
 - * Support fee for sending L1 messages. At this point, it's not mandatory and messages with no fee will still be handled. Starting from the next version it will become mandatory.

Cairo:

Syntax changes in Cairo (to make it more similar to rust and C++):

- * You can use the cairo-migrate script to convert old code to the new syntax. Use the `-i` flag to apply the changes to the files
- * End statements with ``;`

[NOTE]

====

New lines are still part of the language at this point, and you cannot put more than one instruction per line. This will change in Cairo1.0.

====

- * Use ``{ ... }`` for code blocks (instead of ``:`` and ``end``)
- * Add ``()` around the condition of if statements
- * Remove the member keyword in structs
- * Change comment to use ``//`` instead of ``#``
- * Use ``..., ap++`` instead of ``...; ap++`` in low level Cairo code
- * Support return types that are not tuples. For example, ``func foo() -> felt`` (instead of ``func foo() -> (r: felt)``)

As a result, it's now mandatory to specify return types. ``func foo() -> (res)`` should be replaced by ``func foo() -> (res: felt)``. The cairo-migrate tool does that automatically.

- * Return statement accepts expressions, rather than only tuples. For example, you can write ``let x = (5,); return x;``
- * A few standard library functions were changed to return felt. The cairo-migrate script also fixes calls to those functions
- * Support using functions as expressions
- * This only applies to functions with `-> felt` signature, whose ap change is known at compile-time (e.g., recursive functions cannot be used this way)
- * Fix a bug in the secp signature verification code that allowed a malicious prover to ignore the value of ``v`` (this does not let the prover fake a signature, but allows it to claim that a valid signature is invalid).
- * Add Cairo code for the recursive STARK verifier

Technical changes:

- * Move from python3.7 to python3.9

[id="version0.9.1"]

== Starknet v0.9.1 (July 20, 22)

This version contains the following changes:

Starknet:

API changes:

- * Add ``get_block_traces`` API - returns all the transaction traces of a given block
- * Add a list of declared contracts in ``get_state_update``
- * Add a 0x prefix for class hash in the API
- * Add ``starknet_version`` field for blocks (only applies to new blocks)

Starknet CLI:

- * Change the default block number to pending
- * Using a wallet is the default, ``--no_wallet`` must be specified explicitly to override this
- * Deploying contracts:
 - ** Add ``deploy_contract`` function to the account contract created by ``starknet deploy_account``
 - ** Use this function to deploy contract (unless using ``--no_wallet``). In particular, ``deploy`` should be used after declaring the contract (it expects the contract class hash)
- * Support ``--dry_run`` to get the transaction information without signing or sending it
- * Support ``deploy_from_zero`` in the ``deploy`` syscall to deploy a contract to an address that does not depend on the deployer

Cairo:

- * Support and in if statements (``if x == y and z == w``).

[NOTE]

====

At the moment other boolean combinations are not supported

====

[id="version0.9.0"]

== Starknet v0.9.0 (June 06, 22)

This version introduces the contract class/instance paradigm into Starknet. See https://docs.starknet.io/documentation/architecture_and_concepts/Contracts/contract-classes/ [here] for more information.

This version contains the following changes:

Starknet:

- * Enforce fees - ``max_fee`` must not be set to zero, and selector must be ``__execute__``
- * Split the concepts of contract class and contract instance.
- * Add ``declare`` transaction type
- * New API and CLI commands:
 - * ``declare`` - Declares a contract class
 - * ``get_class_by_hash`` - Returns the contract class given its hash
 - * ``get_class_hash_at`` - Returns the class hash for a given contract instance address

- * Rename ``delegate_call`` to ``library_call``, and change the contract address argument to class hash.
- * Add a ``deploy`` system call.
- * Rename ``ContractDefinition`` to ``ContractClass``
- * Reduce the compiled contract file's size by removing unnecessary identifiers (this optimization can be disabled using ``--dont_filter_identifiers``)

Cairo:

- * Initial support for the ``EC-op`` builtin (scalar multiplication over the STARK curve). Not supported in Starknet yet.
- * Add additional helper methods to ``blake2s.cairo``, including big-endian support

Technical changes:

- * Change function's ``return`` type from a struct to a named tuple. In particular, ``foo.Return.SIZE`` is no longer supported.
- * Tools and resources

** Developer Tools

- *** [xref:devtools/overview.adoc](#)[Overview]
- *** [xref:devtools/clis.adoc](#)[Command Line Tools]
- *** [xref:devtools/sdks.adoc](#)[Software Development Kits]
- *** [xref:devtools/smart-contract-tools.adoc](#)[Smart Contract Development]
- *** [xref:devtools/vscode.adoc](#)[Visual Studio Code Extension]
- *** [xref:devtools/devnets.adoc](#)[Local Development Nodes]
- *** [xref:devtools/libs-for-dapps.adoc](#)[Libraries for Dapps]
- *** [xref:devtools/dapp-frameworks.adoc](#)[Dapp Frameworks]
- *** [xref:devtools/utilities.adoc](#)[Utilities]
- *** [xref:devtools/security.adoc](#)[Security Tools]

** [xref:api-services.adoc](#)[Full nodes and API services]

** [xref:ref-block-explorers.adoc](#)[Block explorers]

** [xref:audit.adoc](#)[Audit providers]

* StarkGate bridge guide

- ** [xref:starkgate:overview.adoc](#)[Overview]
- ** [xref:starkgate:architecture.adoc](#)[StarkGate architecture]
- ** Procedures
 - *** [xref:starkgate:depositing.adoc](#)[Depositing funds]
 - *** [xref:starkgate:withdrawing.adoc](#)[Withdrawing funds]
 - *** [xref:starkgate:automated-actions-with-bridging.adoc](#)[Performing a Smart Deposit]
 - *** [xref:starkgate:adding-a-token.adoc](#)[Adding a token]
 - *** [xref:starkgate:cancelling-a-deposit.adoc](#)[Cancelling a deposit]
 - *** [xref:starkgate:estimating-fees.adoc](#)[Estimating StarkGate fees]

- *** xref:dai-token-migration.adoc[Migrating DAI v0 to DAI]
- ** xref:starkgate:function-reference.adoc[StarkGate function and event reference]
- * Staking (WIP) Ø=Þàþ
 - ** xref:staking:overview.adoc[Overview (WIP) Ø=Þàþ]
 - ** xref:staking:architecture.adoc[Staking Architecture (WIP) Ø=Þàþ]
 - ** Procedures (WIP) Ø=Þàþ
 - *** xref:staking:entering-staking.adoc[Becoming a Validator (WIP) Ø=Þàþ]
 - *** xref:staking:increasing-staking.adoc[Increasing Stake (WIP) Ø=Þàþ]
 - *** xref:staking:claiming-rewards.adoc[Claiming Rewards (WIP) Ø=Þàþ]
 - *** xref:staking:delegating-stake.adoc[Delegating Stake (WIP) Ø=Þàþ]
 - *** xref:staking:switching-delegation-pools.adoc[Switching Delegation Pools (WIP) Ø=Þàþ]
 - *** xref:staking:exiting-staking.adoc[Exiting the Staking Protocol (WIP) Ø=Þàþ]
 - *** xref:staking:managing-staking-and-delegation-operations.adoc[Managing Delegation Pools (WIP) Ø=Þàþ]
 - *** xref:staking:handling_staking_events.adoc[Handling Staking Events (WIP) Ø=Þàþ]
- * Important addresses
 - ** xref:important-addresses.adoc[Starknet contracts and sequencer addresses]
 - ** xref:bridged-tokens.adoc[Bridged tokens]
- * xref:limits-and-triggers.adoc[Current limits]
 - = Full nodes and API services

A list of recommended full-nodes, open API endpoints, and API providers.

For complete information on the Starknet Node API in JSON RPC format, see link:https://github.com/starkware-libs/starknet-specs/blob/master/api/starknet_api_openrpc.json [`starknet_api_openrpc.json`] on GitHub.

.API providers

[cols="1,2,2",stripes=even]

|===

|Provider |Open API endpoint, where relevant |Version support, where relevant

|<http://www.alchemy.com/starknet>[Alchemy] | |<https://docs.alchemy.com/reference/starknet-api-faq#what-versions-of-starknet-api-are-supported>[Starknet API FAQ]

|<https://www.allthatnode.com/starknet.dsrv>[All That Node] | |

|<http://blastapi.io/public-api/starknet>[Blast API] |<https://blastapi.io/public-api/starknet> |<https://blastapi.io/public-api/starknet> a|
<https://blastapi.io/public-api/starknet>[Starknet Public API - Blast API]

|<http://blockpi.io/starknet>[BlockPI] |<https://starknet.blockpi.network/v1/rpc/public>[<https://starknet.blockpi.network/v1/rpc/public>]

|<http://chainbase.com/chainNetwork/Starknet>[Chainbase] | |

|<https://chainstack.com/build-better-with-starknet/>[Chainstack] | a| link:<https://docs.chainstack.com/reference/getting-started-starknet#starknet-json-rpc-version-endpoints>[Getting started > Starknet JSON-RPC version endpoints]

|<https://drpc.org/public-endpoints/starknet>[DRPC] |<https://drpc.org/public-endpoints/starknet>[<https://drpc.org/public-endpoints/starknet>] |

|<https://www.dwellir.com/>[Dwellir] |<https://www.dwellir.com/networks/starknet>[<https://www.dwellir.com/networks/starknet>] |

|<https://getblock.io/nodes/strk/>[GetBlock] | |Use the `/rpc/vX_Y` suffix, as explained in <https://github.com/eqlabs/pathfinder?tab=readme-ov-file#json-rpc-api>[JSON-RPC API] in Pathfinder's README.

|<https://www.infura.io/networks/ethereum/starknet>[Infura] | |

|<https://www.lavanet.xyz/>[Lava Protocol] |<https://www.lavanet.xyz/get-started/starknet>[<https://www.lavanet.xyz/get-started/starknet>] | Use the `/rpc/vX_Y` suffix, as explained in <https://github.com/eqlabs/pathfinder?tab=readme-ov-file#json-rpc-api>[JSON-RPC API] in Pathfinder's README.

|<https://data.voyager.online/>[Nethermind] |<https://data.voyager.online/>[<https://data.voyager.online/>]

<https://docs.data.voyager.online/spec>[RPC spec versions]

|link:<https://nownodes.io/starknet>[NOWNodes] | |

|link:<https://omniatech.io/>[OMNIA] | |

|link:<https://www.quicknode.com/chains/strk>[QuickNode] |<https://www.quicknode.com/docs/starknet#supporting-multiple-versions>[Supporting Multiple Versions]

|<https://www.reddio.com/node>[Reddio] | |

|<https://zan.top/home/node-service>[Zan] | |

|===

'''

.Node providers

[cols="1,2,1",stripes=even]

[%header,cols="2,2,1"]

|===

| Provider name | Description | More information

|Deoxys|A Starknet full-node written in Rust and powered by Substrate by Kasar |
link:<https://github.com/kasarlabs/deoxys>
|Juno|A Starknet full-node written in go-lang by Nethermind.

You can use the link:<https://aws-samples.github.io/aws-blockchain-node-runners/docs/Blueprints/Starknet>
|link:<https://github.com/NethermindEth/juno>

|Papyrus|A Starknet full-node written in Rust by StarkWare | link:<https://github.com/starkware-libs/papyrus>
|Pathfinder|A Starknet full-node written in Rust by Equilibrium |link:<https://github.com/eqlabs/pathfinder>

|===

[id="audit_providers"]

= Audit providers

Building a Starknet project and want your contract to be audited?

The companies listed below have designated teams that provide auditing services to Starknet contracts.

.A list of companies providing contract audits for Starknet

[cols="1,2",stripes=even]

[%autowidth.stretch]

|===

| Company name | URL

|ABDK | link:<https://www.abdk.consulting/>

|Beosin | link:<https://beosin.com/>

|Chain Security | link:<https://chainsecurity.com/>

|Consensys Diligence | link:<https://consensys.net/diligence/>

|Extropy | link:<https://security.extropy.io/>

|Nethermind | link:<https://nethermind.io/>

|Open Zeppelin | link:<https://www.openzeppelin.com/>

|OtterSec | link:<https://osec.io/>

|PeckShield | link:<https://peckshield.com/>

|Trail of Bits | link:<https://www.trailofbits.com/>

|Zellic | link:<https://www.zellic.io/>

|Ginger Security | link:<https://gingersec.xyz/>

|===

[id="bridged_tokens"]

= Bridged tokens and addresses

The tokens that are currently bridged to Starknet, including their L1 and L2 addresses, are listed in the following ``.json`` files:

[horizontal, labelwidth="20"]

link:https://github.com/starknet-io/starknet-addresses/blob/master/bridged_tokens/mainnet.json[`mainnet.json`^]: The addresses of the tokens currently bridged to Starknet Mainnet.

link:https://github.com/starknet-io/starknet-addresses/blob/master/bridged_tokens/sepolia.json[`sepolia.json`^]: The addresses of the tokens currently bridged to Starknet Sepolia testnet.

Each token has the following parameters:

[horizontal, labelwidth="20"]

`name`:: Token name.

`symbol`:: Token symbol.

`decimals`:: Number of decimal places used to get the user representation.

`l1_token_address`:: Address of the L1 ERC-20 contract.

`l2_token_address`:: Address of the L2 ERC-20 contract.

`l1_bridge_address`:: Address of the L1 bridge contract.

`l2_bridge_address`:: Address of the L2 bridge contract.

[id="dai_token_migration"]

= Migrating DAI v0 to DAI

:description: Migrating DAI v0 on Starknet to DAI on Starknet.

:keywords: Starknet DAI, DAI doesn't work on Starknet, New DAI on Starknet, Starknet new DAI

Following link:[https://twitter.com/MakerDAO/status/1746977683190251591?](https://twitter.com/MakerDAO/status/1746977683190251591?s=20)

s=20[Maker DAO's announcement on Jan 25th], StarkWare launched a new DAI contract and bridge on Starknet.

The new DAI token and bridge are a part of StarkGate and compatible with StarkGate 2.0 features.

You can withdraw old DAI tokens (DAI v0) without any limitation. Depositing using the DAI v0 bridge are disabled. You are encouraged to migrate to the new DAI token. You can use swap services on Starknet to swap DAI v0 for DAI.

Maker DAO's DAI token on Starknet is written in Cairo0 and is not upgradeable. Without upgradability, it cannot support StarkGate's latest features, such as Smart Deposits and Withdrawal Limits, and over time it will stop being compatible with Starknet altogether (Regenesis). This means that a transition plan is necessary.

On January 25th, StarkWare launched a new set of DAI bridge and token contracts under StarkGate, written in Cairo. This new DAI token will retain the same contract 'symbol' and 'name' as the existing one. To differentiate between the two on Apps and other UIs, we refer to the old DAI as "DAI v0" and the new DAI simply as "DAI."

.Procedure

Use one of the following methods:

- * Swap your DAI v0 for DAI using an L2 swap app or aggregator within the Starknet ecosystem, such as the following:

- ** link:[https://www.layerswap.io/migration/DAI\[LayerSwap\]](https://www.layerswap.io/migration/DAI[LayerSwap])

- ** link:[https://app.ekubo.org/?](https://app.ekubo.org/?amount=1000&inputCurrency=DAI&outputCurrency=DAIv2)

- [amount=1000&inputCurrency=DAI&outputCurrency=DAIv2](https://app.ekubo.org/?amount=1000&inputCurrency=DAI&outputCurrency=DAIv2)[Ekubo]

- ** link:<https://app.avnu.fi/en?tokenFrom=0x49d36570d4e46f48e99674bd3fcc84644ddd6b96f7c741b1562b82f9e004dc7&tokenTo=0x4718f5a0fc34cc1af16a1cdee98ffb20c31f5cd61d6ab07201858f4287c938d&amount=0.001>[AVNU]

- ** link:<https://app.fibrous.finance/>[Fibrous]

- ** link:<https://app.jediswap.xyz/#/swap>[JediSwap]

- ** link:<https://app.haiko.xyz/trade/swap>[Haiko]

- ** link:<https://app.nostra.finance/>[Nostra]

- ** link:<https://app.starkdefi.com/#/swap>[StarkDeFi]

- * Use StarkGate:

- +

- . Withdraw your current DAI (DAI v0) to L1 using StarkGate

- . Re-deposit your L1 DAI using StarkGate.

- +

- StarkGate automatically issues the new DAI.

- = Command-line tools

[#starkli]

== starkli

Starkli is a fast command-line interface for interacting with the Starknet network. It supports fetching data from the Starknet network, deploying accounts, and interacting with contracts. Developed by <https://x.com/xjonathanlei>[Jonathan Lei].

Starkli also includes useful utilities for developers, such as:

- * Compute class hashes from the Cairo file that defines the class.

- * Compute a function's selector.

- * Encode messages.

- * Deploying new accounts or fetching existing accounts

- * Submitting multi-calls to your account

[discrete]

=== Relevant links

- * GitHub: link:<https://github.com/xJonathanLEI/starkli>[starkli on GitHub]

- * Documentation: link:<https://book.starkli.rs/>[Starkli Book]

* link:<https://medium.com/starknet-edu/starkli-the-new-starknet-cli-86ea914a2933>[Starkli: The New Starknet CLI], on Medium, includes information on getting started, including installation instructions.

[#sncast]

== sncast

Starknet Cast, part of the Starknet Foundry suite, is a command line tool for interacting with the Starknet network, with deep integration with [xref:tools:devtools/smart-contract-tools.adoc#starknet_foundry](#)[Starknet Foundry] projects.

* link:<https://foundry-rs.github.io/starknet-foundry/starknet/index.html>[Starknet Cast Docs]

[#dapp-frameworks]

= Dapp Frameworks

[#scaffold-stark]

== Scaffold-Stark

Built using NextJS, Starknet.js, Scarb, Starknet-React, Starknet Foundry and Typescript. Designed to make it easier for developers to create, deploy and interact with smart contracts.

[discrete]

=== Relevant links

* Website: link:<https://scaffoldstark.com/>[Scaffold-Stark website]

* Docs: link:<https://www.docs.scaffoldstark.com/>[Scaffold-Stark Docs]

* GitHub: link:<https://github.com/Quantum3-Labs/scaffold-stark-2>[starknetkit on GitHub]

[#starknet-scaffold]

== Starknet Scaffold

An open-source, up-to-date toolkit for building decentralized applications (dapps) on Starknet. Move from prototyping to production-grade apps seamlessly.

[discrete]

=== Relevant links

* Website: link:<https://www.starknetscaffold.xyz/>[Starknet-Scaffold website]

* Docs: link:<https://docs.starknetscaffold.xyz/>[Starknet-Scaffold Docs]

* GitHub: link:<https://github.com/horuslabsio/Starknet-Scaffold>[Starknet-Scaffold on GitHub]

[#devnets]

= Local Development nodes

A Starknet devnet is a local Starknet node implementations, aimed for testing and

development. A devnet behaves just like a real Starknet node, but everything is executed locally. This enables much faster and more private development of Starknet applications.

[#starknet-devnet-rs]
== starknet-devnet-rs

starknet-devnet-rs can is a Rust implementation of a local Starknet node. Developed by SpaceShard.

With starknet-devnet-rs includes many featured tailored for testing and development, which are not present on testnet/mainnet. Some of the features include:

- * Pre-deployed and pre-funded accounts
- * Forking the chain at a specific block.
- * Dumping current state (and loading in future runs)
- * Impersonating account
- * Mock L1<>L2 communication

[discrete]
=== Relevant links

- * GitHub: link:<https://github.com/0xSpaceShard/starknet-devnet-rs>[starknet-devnet-rs on GitHub]
- * Crates: link:<https://crates.io/crates/starknet-devnet>[starknet-devnet-rs on Crates]
- * Documentation: link:<https://0xspaceshard.github.io/starknet-devnet-rs/>[starknet-devnet-rs Docs]
- * Support: devnet channel on link:<https://discord.gg/starknet-community>[Starknet Discord]

[#katana]
== Katana

Katana, developed by the link:https://x.com/cartridge_gg[Dojo team], is an extremely fast devnet designed to support local development with Dojo, which is a gaming engine for Starknet. You can use Katana as a general purpose devnet as well.

[discrete]
=== Relevant links

- * GitHub: link:<https://github.com/dojoengine/dojo>[Dojo Engine on GitHub]
 - * Documentation: link:<https://book.dojoengine.org/toolchain/katana>[Katana Docs]
- [#libs-for-dapps]
= Libraries for Dapps

Decentralized applications are at the heart of Starknet. These libraries help developers build Dapps on Starknet, and connect to prominent Starknet wallets.

[#starknet-react]

== Starknet React

Starknet React is a collection of React hooks for Starknet. It is inspired by wagmi, powered by starknet.js. Developed by Apibara.

[discrete]

=== Relevant links

* GitHub: link:<https://github.com/apibara/starknet-react>[starknet-react on GitHub]

* Package: link:<https://www.npmjs.com/package/@starknet-react/core>[starknet-react on NPM]

* Documentation: link:<https://starknet-react.com/>[Starknet-React Docs]

* Beta Version Documentation: <https://v3.starknet-react.com/docs/getting-started>[Starknet-React V3 Docs]

[#get-starknet]

== Get Starknet

Starknet wallet<>Dapp connection bridge. Easy discovery and UI for Starknet wallets.

Supporting popular Starknet browser wallets

* ArgentX

* Braavos

* Metamask Snaps

* OKx

[discrete]

=== Relevant links

* GitHub: link:<https://github.com/starknet-io/get-starknet>[get-starknet on GitHub]

* Package: link:<https://www.npmjs.com/package/@starknet-io/get-starknet>[get-starknet on NPM]

[#starknetkit]

== Starknetkit

A Starknet wallet connection kit, built by Argent. Built using Starknet.js and starknet-react.

[discrete]

=== Relevant links

- * Website: link:<https://www.starknetkit.com/>[starknetkit website]
- * Docs: link:<https://www.starknetkit.com/docs/getting-started>[starknetkit Docs]
- * GitHub: link:<https://github.com/argentlabs/starknetkit>[starknetkit on GitHub]

[id="starknet_development_tools"]

= Starknet developer tools

Cairo is a turing complete language for writing provable programs. It is also the language used for writing smart contracts for Starknet. Starknet uses a different model than the Ethereum Virtual Machine (EVM), and as such, requires specialized tools for development.

image::devtools_flow.svg[Devtools Flow]

[NOTE]

=====

This list of tools is dynamic and is being updated as more tools are added. If a tool you are using is missing, please consider editing this page and creating a Pull Request.

=====

[#sdks]

= Starknet SDKs

A Software Development Kit (SDK) is a library that abstracts the complexities of Starknet when building transactions and interacting with the blockchain, including the following:

- * Read the chain state.
- * Account creation and management.
- * Cryptography: signature verification and signing, computing hashes used by Starknet.
- * Contract declaration and deployment.
- * Contract interactions: ABI import, constructing transactions.

SDKs implement the Starknet link:<https://github.com/starkware-libs/starknet-specs>[JSON RPC specification], and are updated to support the latest API changes. There are SDKs for various languages, so you can choose the SDK according to your needs.

[cols=" , , , , " ,]

|===

| SDK name | Github | Package | Docs | Support | Maintainer

[Starknet.js](https://github.com/starknet-io/starknet.js) | [link:https://github.com/starknet-io/starknet.js](https://github.com/starknet-io/starknet.js)[starknet.js on GitHub] |
[link:https://www.npmjs.com/package/starknet](https://www.npmjs.com/package/starknet)[starknet.js on NPM] | [link:https://www.starknetjs.com/](https://www.starknetjs.com/)[starknet.js Book] | starknet.js channel on [link:https://discord.gg/starknet-community](https://discord.gg/starknet-community)[Starknet Discord] | [link:https://x.com/OxSpaceShard](https://x.com/OxSpaceShard)[SpaceShard]
[Starknet.py](https://github.com/software-mansion/starknet.py) | [link:https://github.com/software-mansion/starknet.py](https://github.com/software-mansion/starknet.py)[starknet.js on GitHub] | [link:https://pypi.org/project/starknet-py/](https://pypi.org/project/starknet-py/)[starknet.py on PyPi] | [link:https://starknetpy.rtfld.io/](https://starknetpy.rtfld.io/)[starknet.py Docs] | [link:https://t.me/starknetpy](https://t.me/starknetpy)[starknet.py on Telegram] | [link:https://x.com/swmansionxyz](https://x.com/swmansionxyz)[Software Mansion]
[Starknet-rs](https://github.com/xJonathanLEI/starknet-rs) | [link:https://github.com/xJonathanLEI/starknet-rs](https://github.com/xJonathanLEI/starknet-rs)[starknet-rs on GitHub] |
[link:https://crates.io/crates/starknet](https://crates.io/crates/starknet)[starknet-rs on Crates] | [link:https://github.com/xJonathanLEI/starknet-rs](https://github.com/xJonathanLEI/starknet-rs)[starknet-rs Docs] | [link:https://t.me/starknet_rs](https://t.me/starknet_rs)[starknet-rs on Telegram] | [link:https://x.com/xjonathanlei](https://x.com/xjonathanlei)[Jonathan Lei]
[Starknet.go](https://github.com/NethermindEth/starknet.go) | [link:https://github.com/NethermindEth/starknet.go](https://github.com/NethermindEth/starknet.go)[starknet.go on GitHub] |
[link:https://pkg.go.dev/github.com/NethermindEth/starknet.go](https://pkg.go.dev/github.com/NethermindEth/starknet.go)[starknet.go Docs] | [link:https://t.me/StarknetGo](https://t.me/StarknetGo)[starknet.go on Telegram] | [link:https://x.com/NethermindEth](https://x.com/NethermindEth)[Nethermind]
[starknet-jvm](https://github.com/software-mansion/starknet-jvm) | [link:https://github.com/software-mansion/starknet-jvm](https://github.com/software-mansion/starknet-jvm)[starknet-jvm on GitHub] | [link:https://docs.swmansion.com/starknet-jvm/](https://docs.swmansion.com/starknet-jvm/)[starknet-jvm Docs] | [link:https://x.com/swmansionxyz](https://x.com/swmansionxyz)[Software Mansion]
[starknet.swift](https://github.com/software-mansion/starknet.swift) | [link:https://github.com/software-mansion/starknet.swift](https://github.com/software-mansion/starknet.swift)[starknet.swift on GitHub] | [link:https://docs.swmansion.com/starknet.swift/documentation/starknet/](https://docs.swmansion.com/starknet.swift/documentation/starknet/)[starknet.swift Docs] | [link:https://x.com/swmansionxyz](https://x.com/swmansionxyz)[Software Mansion]
[starknet.dart](https://github.com/focustree/starknet.dart) | [link:https://github.com/focustree/starknet.dart](https://github.com/focustree/starknet.dart)[starknet.dart on GitHub] |
[link:https://pub.dev/packages/starknet](https://pub.dev/packages/starknet)[starknet.dart on Pub] | [link:https://starknetdart.dev/](https://starknetdart.dev/)[starknet.dart Docs] | [link:https://t.me/+CWezjfLIRYc0MDY0](https://t.me/+CWezjfLIRYc0MDY0)[starknet.dart on Telegram] | [link:https://x.com/focustree_app](https://x.com/focustree_app)[Focustree]
 |===

[#Security]
 = Security and Analysis tools

[#sierra-analyzer]
 == Sierra Analyzer

Sierra-Analyzer is a security toolkit for analyzing Sierra files, developed by [link:https://x.com/fuzzinglabs](https://x.com/fuzzinglabs)[FuzzingLabs].

Supported featurued include:

- * Decompile a Sierra file
- * Print the contracts Control Flow Graph
- * Run Static Analysis detectors

Links:

* link:<https://github.com/FuzzingLabs/sierra-analyzer>[sierra-analyzer on GitHub]

[#entro]

== Entro

Analyze and Decode Starknet Transactions and events.

Features:

- * Get contract class history
- * Decode contract ABI
- * Decode transaction data
- * Backfill data for faster analysis

Links:

* link:<https://github.com/NethermindEth/entro>[Entro on GitHub]

[#Contract-Development]

= Smart Contract Development

The following tools are the recommended tools for developing Starknet smart contracts. These cover compilation, package management, testing and deployment.

[#scarb]

== Scarb: The Cairo package manager

Scarb is a package manager for Cairo, but it is much more than that. It is the easiest and recommended way to build and maintain Cairo code. Think Cargo for Rust. Scarb is developed by Software Mansion.

Scarb includes the following features:

- * Initiating a new Cairo project.
- * Compiling Cairo projects.
- * Adding and removing Cairo dependencies.
- * Generating Cairo documentation.
- * Fetching and uploading packages link:<https://scarbs.xyz/>[Scarbs.xyz], the Cairo Registry.
- * Integration with the Cairo Language Server
- * It integrates with other tools in the Cairo ecosystem, such as Starknet Foundry and the Dojo gaming engine.

[discrete]

=== Relevant links

- * GitHub: link:<https://github.com/software-mansion/scarb>[Scarab on GitHub]
- * Documentation: link:<https://docs.swmansion.com/scarb/>[Scarab Docs]
- * Support: link:<https://t.me/+1pMLtrNj5NthZWJk>[Scarab on Telegram]
- * Cairo Registry: link:<https://scarbs.xyz/>[scarbs.xyz]

[#starknet_foundry]
 == Starknet Foundry

Starknet Foundry is the go-to toolchain for developing Starknet smart contracts. Similarly to its EVM counterpart, Starknet Foundry supports a plethora of features focused on testing Cairo smart contracts for Starknet.

Starknet Forge, and `snforge_std` allow the use of "cheatcodes" to test various aspects of the contracts.

For example:

- * Setting caller address
- * Manipulating the timestamp and block number
- * Forking the chain at a specific block and testing with that state
- * Fuzz testing
- * Getting accurate gas and resource reports
- * Profiling

Starknet Cast is a command line tool for interacting with the Starknet network, with deep integration with Starknet Foundry projects. With `sncast` it is possible to:

- * Declare and deploy contracts
- * Read from Starknet contracts
- * Deploy accounts
- * Interact with contracts

[discrete]
 === Relevant links

- * GitHub: link:<https://github.com/foundry-rs/starknet-foundry>[starknet-foundry on GitHub]
- * Documentation: link:<https://foundry-rs.github.io/starknet-foundry/>[starknet-foundry Docs]
- * Support: link:https://t.me/starknet_foundry_support[Starknet Foundry Support on Telegram]

[#starknet_remix_plugin]
 == The Starknet Remix plugin

Remix is a browser-based integrated development environment (IDE) for Ethereum that you can use for learning, experimenting and finding vulnerabilities in smart contracts, without installing anything. The Starknet Remix plugin lets you use Remix for testing Starknet smart contracts, so you can focus on learning Cairo and Starknet in the comfort of your browser.

Remix and the Starknet Remix plugin include the following features:

- * Integrated compiling.
- * You can deploy contracts to testnet, mainnet and the plugin's own integrated devnet.
- * You can call functions of contracts that you have already deployed, to facilitate testing and interaction.
- * The Starknet Remix Plugin is integrated with link:<https://starknet-by-example.voyager.online/>[Starknet By Example], a rich repository of practical learning content.

[discrete]

=== Relevant links

Remix Project: link:<https://remix-project.org/>[Remix Project site].

* Blogpost: link:<https://medium.com/nethermind-eth/unlocking-onboarding-to-starknet-an-overview-of-the-starknet-remix-plugin-6b0658e73521>[Unlocking Onboarding to Starknet: An Overview of the Starknet Remix Plugin].

* GitHub link:<https://github.com/NethermindEth/starknet-remix-plugin>[Starknet Remix on GitHub].

[#utilities]

= Utilities

While not under any specific category, these tools can be helpful in various stages of development.

[#usc]

== Universal Sierra Compiler

While Scarb compiles full projects, and produces both Sierra and CASM files, it is often needed to only compile a single Sierra file to CASM (for example, when getting a class from Starknet mainnet). The Universal Sierra Compiler supports all sierra versions, and can compile the the a CASM file.

[NOTE]

=====

The USC comes bundled with Starknet Foundry and does not need to be installed separately if Starknet Foundry is installed.

=====

[discrete]

=== Relevant links

* link:<https://github.com/software-mansion/universal-sierra-compiler>[Universal Sierra Compiler on GitHub]

[#rpc-request-builder]

== RPC Request Builder

The Starknet RPC Request Builder is a useful tool to generate RPC queries for Starknet, with support for basic example for JavaScript, Go and Rust.

[discrete]

=== Relevant links

* link:<https://rpc-request-builder.voyager.online/>[RPC Request Builder]

[#open-zeppelin-contract-wizard]

== Open Zeppelin Contract Wizard

The Open Zeppelin Contract Wizard is a tool that helps you create smart contracts with Open Zeppelin libraries. Easily toggle on and off features for popular smart contract patterns, and the wizard will generate the code for you.

[discrete]

=== Relevant links

* link:<https://wizard.openzeppelin.com/cairo>[Open Zeppelin Contract Wizard]

[#cairo-profiler]

== Cairo Profiler

Cairo-profiler can be used to create profiles of Cairo executions from execution traces. These can be analyzed and displayed to show Flame Graphs, and other useful information.

[NOTE]

====

Cairo-profiler is currently integrated into Starknet Foundry, but can be used as a standalone tool.

====

[discrete]

=== Relevant links

* link:<https://github.com/software-mansion/cairo-profiler>[Cairo Profiler on GitHub]

[#cairo-playground]

== Cairo Playground

If want to dive deep into the Cairo VM, and experiment writing Cairo online, and don't want to deploy a smart contract on Starknet, the Cairo Playground is a great way to do so.

[discrete]

=== Relevant links

* link:<https://www.cairo-lang.org/cairovm/>[Cairo Playground]

[#starknet-devnet-js]

== Starknet Devnet JS

A JavaScript package, abstracting the Starknet Devnet API, making it easier to interact with starknet-devnet-rs.

This simplifies writing end-to-end tests using Devnet, including tests for L1<>L2 communications.

Notable features:

- * Spawn a new Devnet instance without installing it
- * Wrapping RPC calls to Devnet
- * Abstracting complex L1<>L2 communication setup with a local L1 node (e.g. Anvil)

[discrete]

=== Relevant links

* link:<https://github.com/0xSpaceShard/starknet-devnet-js>[starknet-devnet-js on GitHub]

[#vs_code_cairo_extension]

= The Visual Studio Code Cairo extension

An extension for the Microsoft VSCode IDE that provides assistance when writing Cairo smart contracts, by using the Cairo Language Server. It integrates with Scarb, and works best when Scarb is installed via `asdf`.

Features include:

- * Live diagnostic highlighting for compile errors
- * Quick fixes with suggestions
- * Go to definition
- * Code formatting
- * Code completion for imports

[discrete]

=== Relevant links

* link:<https://marketplace.visualstudio.com/items?itemName=starkware.cairo1>[Cairo 1.0]

- Visual Studio Marketplace]

* link:<https://github.com/starkware-libs/cairo/tree/main/vscode-cairo>[vscode-cairo on GitHub]

[id="important_addresses"]

= Starknet contract and sequencer addresses

include::ROOT:partial\$snippet_important_goerli_removed.adoc[]

== Starknet on Mainnet

[horizontal, labelwidth="15",role="stripes-odd"]

The Starknet Core Contract:: link:<https://etherscan.io/address/0xc662c410C0ECf747543f5bA90660f6ABeBD9C8c4>[`0xc662c410C0ECf747543f5bA90660f6ABeBD9C8c4`^]

Verifier address:: link:<https://etherscan.io/address/0x47312450B3Ac8b5b8e247a6bB6d523e7605bDb60>[`0x47312450B3Ac8b5b8e247a6bB6d523e7605bDb60`^]

Sequencer base URL for API routing:: <https://alpha-mainnet.starknet.io>

== Starknet version on Sepolia testnet

[horizontal, labelwidth="15",role="stripes-odd"]

The Starknet Core Contract:: link:<https://sepolia.etherscan.io/address/0xE2Bb56ee936fd6433DC0F6e7e3b8365C906AA057>[`0xE2Bb56ee936fd6433DC0F6e7e3b8365C906AA057`^]

Verifier address:: link:<https://sepolia.etherscan.io/address/0x07ec0D28e50322Eb0C159B9090ecF3aeA8346DFe>[`0x07ec0D28e50322Eb0C159B9090ecF3aeA8346DFe`^]

Sequencer base URL for API routing:: <https://alpha-sepolia.starknet.io>

== Starknet fee tokens

The Starknet fee tokens are STRK and ETH.

[horizontal, labelwidth="15",role="stripes-odd"]

L2 STRK address (Mainnet and testnet):: link:<https://starkscan.co/contract/0x04718f5a0fc34cc1af16a1cdee98ffb20c31f5cd61d6ab07201858f4287c938d>[`0x04718f5a0fc34cc1af16a1cdee98ffb20c31f5cd61d6ab07201858f4287c938d`]

L2 ETH address (Mainnet and testnet):: link:<https://starkscan.co/contract/0x049d36570d4e46f48e99674bd3fcc84644ddd6b96f7c741b1562b82f9e004dc7>[`0x049d36570d4e46f48e99674bd3fcc84644ddd6b96f7c741b1562b82f9e004dc7`]

== Starknet voting token

The Starknet voting token is vSTRK. For information on vSTRK, see link:[https://governance.starknet.io/learn/vstrk_overview\[_vSTRK overview_\]](https://governance.starknet.io/learn/vstrk_overview[_vSTRK overview_]) on the Starknet Governance Hub.

[horizontal, labelwidth="15",role="stripes-odd"]

Mainnet address:: link:[https://starkscan.co/contract/0x0782f0ddca11d9950bc3220e35ac82cf868778edb67a5e58b39838544bc4cd0f\[0x0782f0ddca11d9950bc3220e35ac82cf868778edb67a5e58b39838544bc4cd0f\]](https://starkscan.co/contract/0x0782f0ddca11d9950bc3220e35ac82cf868778edb67a5e58b39838544bc4cd0f[0x0782f0ddca11d9950bc3220e35ac82cf868778edb67a5e58b39838544bc4cd0f])

Sepolia testnet address:: link:[https://starkscan.co/contract/0x035c332b8de00874e702b4831c84b22281fb3246f714475496d74e644f35d492\[0x035c332b8de00874e702b4831c84b22281fb3246f714475496d74e644f35d492\]](https://starkscan.co/contract/0x035c332b8de00874e702b4831c84b22281fb3246f714475496d74e644f35d492[0x035c332b8de00874e702b4831c84b22281fb3246f714475496d74e644f35d492])

[id="limits_and_triggers"]

= Current limits

Starknet currently has a number of limits in place in order to keep the network stable and optimized for the best performance.

Blockifier-related constants and limits are defined, for each Starknet version starting from v0.13.0, in a JSON file called `versioned_constants` in this link:[https://github.com/starkware-libs/sequencer/tree/main/crates/blockifier/resources\[directory\]](https://github.com/starkware-libs/sequencer/tree/main/crates/blockifier/resources[directory]).

[NOTE]

====

These are subject to revisions and change on a regular basis

====

include::ROOT:partial\$snippet_important_goerli_removed.adoc[]

.Starknet's current limits

[%header, stripes=even]

[%autowidth.stretch]

|====

|Entity | Description | Sepolia | Mainnet

|Block time | The maximum amount of time within which a pending block is closed, if no other limit is met. | 30 seconds | 30 seconds

|Block limit (Cairo steps)| The maximum number of Cairo steps that can be completed within each block to ensure block production times remain consistent and predictable. | 40,000,000 | 40,000,000

|Block limit (gas)| Certain Starknet operations, such as sending messages between L1 and L2, consume Ethereum gas. The current L1 state update mechanism involves an Ethereum transaction for each Starknet block.

The gas limit for Starknet blocks is therefore inherited from the gas limit for Ethereum blocks.

|5,000,000 | 5,000,000

|Max transaction size (Cairo steps)|The maximum number of computational steps, measured in Cairo steps, that a transaction can contain when processed on the Starknet network.

This limit is important for ensuring the efficient execution of transactions and preventing potential congestion.

| 10,000,000 | 10,000,000

|Max number of events per transaction|The maximum number of events that a transaction can emit during its execution.

| 1,000 | 1,000

|Max number of data felts per event|The maximum number of felts that an event can contain in its `data` array.

| 300 | 300

|Max number of key felts per event|The maximum number of felts that an event can contain in its `keys` array.

| 50 | 50

|Max Cairo steps for `validate`| The maximum number of computational steps, measured in Cairo steps, for a `validate` function. | 1,000,000 | 1,000,000

|Max contract bytecode size (Number of felts in the program)| The maximum size of the bytecode or program that a smart contract can have on Starknet.

Bytecode is the low-level code that comprises smart contracts. Limiting this size helps manage the complexity of contracts and the overall efficiency of the network.

| 81,290 | 81,290

|Max contract class size|The maximum size for a contract class within Starknet.

Contract classes are a fundamental building block for smart contracts, and limiting their size can have implications for the network's scalability and security.

| 4,089,446 bytes

| 4,089,446 bytes

|IP address limits (read/write)| In order to reduce network spam, Starknet limits the amount of contract reads and writes that a single IP

address can make. | 200 per min per IP address| 200 per min per IP address

| Signature length (felts) | | 4,000 | 4,000

| Calldata length (felts) | | 4,000 | 4,000

|===

= Block explorers

A `_block explorer_`, or blockchain explorer, enables you to see transactions, blockchain metrics and other information.

The following block explorers provide information on Starknet.

.Starknet block explorers, in alphabetical order

[cols="1,2",stripes=even]

[%autowidth.stretch]

|===

| Block explorer name | URL

| Starkscan | link:<https://starkscan.co>[<https://starkscan.co>^]

| ViewBlock | link:<https://viewblock.io/starknet>[<https://viewblock.io/starknet>^]

| Voyager | link:<https://voyager.online>[<https://voyager.online>^]

| oklink | link:<https://www.oklink.com/starknet>[<https://www.oklink.com/starknet>^]

| NFTScan (NFT explorer) | link:<https://starknet.nftscan.com/>[<https://starknet.nftscan.com/>^]

|===

[id="starknet_book"]

= About the Starknet Book

The [link:https://book.starknet.io](https://book.starknet.io)[Starknet Book] serves as a comprehensive guide to understanding Starknet, Cairo, and introduces you to the Starknet ecosystem.

The Starknet Book caters to various objectives and interests. Mix and match these chapters to customize your learning experience based on your unique interests and requirements. Whether you're exploring smart contract development, frontend integration, or learning about the core architecture, The Starknet Book is your trusted companion on the journey of deepening your understanding of Starknet.

== Table of Contents

[cols="1,3"]

|===

| Chapter | Description

| Chapter 1

| *Introduction to Starknet and Cairo*

How to quickly get started with Starknet and Cairo development.

| Chapter 2

| *Starknet tooling*

An overview of the available tooling for Starknet.

| Chapter 3

| *Starknet architecture*

Learn how Starknet works under the hood. Learn how sequencer, prover and nodes

interact.

| chapter 4

| *Account abstraction*

Account abstraction is one of the main usability features of Starknet. Learn how this works and how you can utilize it.

```
|===  
[id="contributing-to-docs-doc-guidelines"]  
= Documentation guidelines  
// include::_attributes/common-attributes.adoc  
:toc: macro
```

// These guidelines are based on the guidelines for OpenShift documentation. Some sections of the original documentation are commented out. In the future, once it is determined they are not relevant to Starknet docs, those sections might be removed.

== Overview

All Starknet documentation should conform to the same rules and guidelines, as detailed below.

toc::[]

[#basic writing guidelines]
== Basic writing guidelines

- * Avoid the passive voice, use the active voice. Only use the passive voice as a last resort. Most of the time you can write with the active voice.
- * Make it easy to scan
- ** Use helpful titles and headings
- ** Lists
- ** Tables
- * By default, use the second person imperative `_You_`, not third person, `_The user_`. + Talk `_to_` your reader, not `_about_` them.
- * Be consistent, use the same word to refer to a thing in different sentences.
- * Use the present tense, not the future. For example:
+
> To create a Signer you ++++++*will*++++++ need the private key.
- * Don't use slang.
- * Don't refer to programmatic names as words.
- * Don't use Latin (e.g. and i.e.).
- * Avoid quotation marks except for actual quotes.

- * Avoid ambiguous words, like `_may_`. Use `_might_` or `_can_`, depending on your intent, instead of `_may_`.
- * Avoid complex words, use simple words where possible.
- * Avoid using parentheses, except as labels.

== Adding files to the collection

When you create new files, you must add them to the ``components/Starknet/modules/<module>/nav*.adoc`` file so that the build system can render them in the table of contents.

For possible values of ``<module>``, see the directories under ``/components/Starknet/modules/``.

If you don't include the file in the ``nav*.adoc`` file, Antora includes the file in the built output, but it is not accessible via the TOC.

== Structuring information

The documentation guidelines for Starknet build on top of the [link:https://redhat-documentation.github.io/modular-docs/\[_Red Hat modular docs reference guide_\]](https://redhat-documentation.github.io/modular-docs/[_Red Hat modular docs reference guide_]). However, because Antora does not natively account for assemblies, Starknet documentation does not use assemblies, so when reading this guide, focus on the information related to modularizing content based on the three major information types:

- * `_Conceptual_`
- * `_Procedural_`
- * `_Reference_`

== Style guidance

The guidelines on this page are primarily concerned with the modular structure and AsciiDoc / Antora requirements for building Starknet documentation. For style guidance, use these style guides in the following order:

. [xref:starknet_docs_style_guide.adoc\[_Starknet documentation style guide_\]](#)

Reference this guide first. It provides guidance that is specific to Starknet documentation.

. [link:https://redhat-documentation.github.io/supplementary-style-guide\[_Red Hat supplementary style guide for product documentation_\]](https://redhat-documentation.github.io/supplementary-style-guide[_Red Hat supplementary style guide for product documentation_]). This guide overrides certain guidance from the [_Google developer documentation style guide_](#).

. [link:https://developers.google.com/style\[_Google developer documentation style guide_\]](https://developers.google.com/style[_Google developer documentation style guide_])

If you cannot find helpful information or if you must deviate from the guidance in any of

these guides, open an issue in the <https://github.com/starknet-io/starknet-docs-style-guide/issues> [Starknet documentation style guide repo]. The stakeholders can then discuss and determine if and how to address the issue.

== General text editor guidelines

- * Set your editor to strip trailing whitespace.
- * The end of each file should have an empty line.
- // * Do *not* hard wrap lines at 80 characters (or at any other length).
- // +
- // It is not necessary to update existing content to unwrap lines, but you can remove existing hard wrapping from any lines that you are currently working in.

```
[id="topic-file-metadata"]
```

== Topic metadata

Every topic should be placed in a logical directory under ``/components/StarkNet/modules`` with the following metadata at the top of the file:

```
----
[id="<topic-anchor>_{context}"]           <1>
= Topic title                             <2>
----
```

<1> A topic anchor with ``_{context}`` that must be lowercase and must match the topic's file name.

<2> Human readable title. To ensure consistency in the results of the `leveloffset` values in include statements, you must use a level one heading (=) for the topic title.

Example:

```
----

[id="cli-basic-commands_{context}"]
= Basic CLI commands
----
```

== Conditional text

Starknet documentation uses AsciiDoc's ``ifdef/endif`` macro to conditionalize and reuse content, down to the single-line level.

For information on conditionalization in AsciiDoc, see link:<https://docs.asciidoctor.org/asciidoc/latest/directives/conditionals> [Conditionals] in the `_AsciiDoc Language Documentation_`.

For example, if the same file should appear in `_Document A_` and `_Document B_`, with only minor differences:

This first line is unconditionalized, and will appear in both `_Document A_` and `_Document B_`.

```
\ifdef::document_A[]
```

This line will only appear for `_Document A_`.

```
\endif::document_A[]
```

```
ifdef::document_B
```

This line will only appear for `_Document B_`.

```
\endif::document_B[]
```

```
ifndef::document_B
```

This line will not appear for `_Document B_`.

```
\endif::[]
```

[NOTE]

=====

While the ``ifdef/endif`` blocks have no size limit, do not use them to conditionalize an entire file. If an entire file is specific to only some distributions, specify them in the ``nav.adoc`` file.

=====

```
[id="snippet-file-metadata"]
```

```
== Text snippet file metadata
```

Every text snippet should be placed in the ``partials`` folder for the topic in which they are used, and should contain the following metadata at the top:

```
[source,adoc]
```

```
// Text snippet included in the following files: <1>
```

```
//
```

```
// * list of files where this text snippet is included
```

<1> List of topics in which this text snippet is included.

[NOTE]

=====

An anchor ID and human readable title are not required metadata. This type of component is text only and not intended to be published or cross referenced on its own. See `<<writing-text-snippets>>`.

====

.Example:

[source,adoc]

```
// Text snippet included in the following files:  
//  
// * getting_started/pages/account_setup.adoc  
// * getting_started/pages/deploying_contracts.adoc
```

[NOTE]

====

Starknet accounts are smart contracts. As such, creating one involves sending a transaction, and takes a bit longer than creating an EOA on other networks. You can learn more in [https://docs.starknet.io/documentation/architecture_and_concepts/Account_Abstraction/introduction/\[What is an account?\].](https://docs.starknet.io/documentation/architecture_and_concepts/Account_Abstraction/introduction/[What is an account?].)

====

[id="attribute-files"]

== Attribute files

[quote,AsciiDoc Language Documentation]

Document attributes are effectively document-scoped variables for the AsciiDoc language. The AsciiDoc language defines a set of built-in attributes, and also allows the author (or extensions) to define additional document attributes, which may replace built-in attributes when permitted.

For detailed information on attributes in AsciiDocs, see link:[https://docs.asciidoctor.org/asciidoc/latest/attributes/document-attributes/\[Document Attributes\]](https://docs.asciidoctor.org/asciidoc/latest/attributes/document-attributes/[Document Attributes]) in the _AsciiDoc Language Documentation_.

If an attribute is used in multiple files, it is helpful to place those attributes in a single attributes file, and use an ``include`` statement to import those attributes where relevant. The attribute file is a normal AsciiDoc file.

All attribute files must be placed in the ``partials`` directory for the primary topic that uses them. Reference an attributes file using the following syntax:

```
include::partial$attributes/<file_name>.adoc[]
```

For example:

```
include::partial$attributes/attributes.adoc[]
```

If files in more than one topic reference the same attribute file, use the following syntax, or suggest a new strategy in a Github issue:

. Create a symlink to the attributes file in the `partials` directory of the parent module for the file that includes the attributes file.

For example: Consider the following files:

- * Attributes file: `/components/Starknet/modules/ROOT/partial\$attributes.adoc`
- * Content file: `/components/Starknet/modules/useful_info/pages/audit.adoc`

To include `attributes.adoc` in `audit.adoc`:

```
include::$ROOT:partial$attributes.adoc[]
```

== File names

Try to shorten the file name as much as possible _without_ abbreviating important terms that might cause confusion. For example, the `managing-authorization-policies.adoc` file name would be appropriate for a topic entitled _Managing Authorization Policies_.

== Directory names

If you create a directory with a multiple-word name, separate each word with an underscore, for example `backup_and_restore`.

Do not create or rename a top-level directory in the repository and topic map without checking with the docs team first.

```
// [TIP]
// ====
// To create the symbolic links:
//
// . Navigate to the directory that you need to add the links in.
// . Use the following command to create a symbolic link:
// +
// ----
// $ ln -s <target_directory> <link_name>
// ----
// +
// For example, if you are creating the links in a directory that is two levels deep, such
// as `cli_reference/cli`, use the following commands:
```

```
// +
// ----
// $ ln -s ../../images/ images
// $ ln -s ../../modules/ modules
// $ ln -s ../../snippets/ snippets
// $ ln -s ../../_attributes/ attributes
// ----
// +
// Be sure to adjust the number of levels to back up (`../`) depending on how deep your
// directory is.
//
// If you accidentally create an incorrect link, you can remove that link by using `unlink
// <link_name>`.
// =====
```

== Discrete headings

If you have a section heading that you do not want to appear in the TOC, for example, if you think that some section is not worth showing up or if there are already too many nested levels, you can use a discrete heading:

<https://docs.asciidoctor.org/asciidoc/latest/blocks/discrete-headings/>

To use a discrete heading, just add `[discrete]` to the line before your unique ID. For example:

```
----
[discrete]
[id="managing-authorization-policies_{context}"]
== Managing authorization policies
----
```

== Anchoring titles and section headings

All titles and section headings must have an anchor ID. The anchor ID must be similar to the title or section heading.

You must add the `{context}` variable to the end of each anchor ID in topic files. When called, the `{context}` variable is resolved into the value declared in the `:context:` attribute in the corresponding section of the document. This enables cross-referencing to topic IDs in context when a topic is included in multiple locations.

[NOTE]

=====

The `{context}` variable must be preceded by an underscore (`_`) when declared in an anchor ID.

====

The following is an example of an anchor ID for a topic file title:

```
----  
[id="sending-notifications-to-external-systems_{context}"]  
= Sending notifications to external systems  
----
```

The following is an example of an anchor ID for a second level (``==`) heading:

```
----  
[id="deployment-scaling-benefits_{context}"]  
== Deployment and scaling benefits  
----
```

== Writing concepts

A `_concept_` contains information to support the tasks that users want to do and must not include task information like commands or numbered steps.

Avoid using gerunds in concept titles. "About <concept>" is a common concept topic title.

For more information about creating concept topics, see the link:<https://redhat-documentation.github.io/modular-docs/#creating-concept-modules>[_Red Hat modular docs reference guide_] and the link:https://raw.githubusercontent.com/redhat-documentation/modular-docs/master/modular-docs-manual/files/TEMPLATE_CONCEPT_concept-explanation.adoc[concept template].

== Writing procedural topics

A `_procedure_` contains the steps that users follow to complete a process or task. Procedures contain ordered steps and explicit commands.

Use a gerund in the procedure title, such as "Creating".

For more information about writing procedural topics, see the link:<https://redhat-documentation.github.io/modular-docs/#creating-procedure-modules>[_Red Hat modular docs reference guide_] and the link:https://raw.githubusercontent.com/redhat-documentation/modular-docs/master/modular-docs-manual/files/TEMPLATE_PROCEDURE_doing-one-procedure.adoc[procedure template].

```
[id="writing-text-snippets"]
```

== Writing text snippets

A `_text snippet_` is an optional component that lets you reuse content in multiple topics. Text snippets are not a substitute for topics but instead are a more granular form of

content reuse.

While a topic is content that a reader can understand on its own (like an article) or as part of a larger body of work (like a guide), a text snippet is not self-contained and is not intended to be published or cross referenced on its own.

Examples include the following:

- * Admonitions that appear in multiple locations.
- * An introductory paragraph that appears in multiple locations.
- * The same series of steps that appear in multiple procedural topics.
- * A deprecation statement that appears in multiple sets of release notes.

Example:

You could write the following paragraph once and include it in each location that explains how to install a cluster using the installer-provisioned default values:

[source,adoc]

In {product-title} version {product-version}, you can install a cluster on {cloud-provider-first} ({cloud-provider}) that uses the default configuration options.

For more information about creating text snippets, see the link:<https://redhat-documentation.github.io/modular-docs/#using-text-snippets>[_Red Hat modular docs reference guide_].

```
// == IP addresses
```

```
//
```

```
// You can include IPv4 addresses from test clusters in examples in the documentation, as long as they are private. Private IPv4 addresses fall into one of the following ranges:
```

```
//
```

```
// * 10.0.0.0 to 10.255.255.255 (class A address block 10.0.0.0/8)
```

```
// * 172.16.0.0 to 172.31.255.255 (class B address block 172.16.0.0/12)
```

```
// * 192.168.0.0 to 192.168.255.255 (class C address block 192.168.0.0/16)
```

```
//
```

```
// Replace all public IP addresses with an address from the following blocks. These address blocks are reserved for documentation:
```

```
//
```

```
// * 192.0.2.0 to 192.0.2.255 (TEST-NET-1 address block 192.0.2.0/24)
```

```
// * 198.51.100.0 to 198.51.100.255 (TEST-NET-2 address block 198.51.100.0/24)
```

```
// * 203.0.113.0 to 203.0.113.255 (TEST-NET-3 address block 203.0.113.0/24)
```

```
//
```

```
// [NOTE]
```

```
// =====
```

// There might be advanced networking examples that require specific IP addresses, or cloud provider-specific examples that require a public IP address. Contact a subject matter expert if you need assistance with replacing IP addresses.

// =====

=== Example URLs

To provide an example URL path that you do not want to render as a hyperlink, use this format:

```
....
`\https://www.example.com`
....
```

== Embedding a local source code file

You can embed local source code files in AsciiDoc topics.

Use the ``include`` directive to target the local file.

To use a local source code file, add it to the ``/<module>/attachments/`` directory, and include it in your module. For example:

```
[source,yaml]
----
\include::attachment$install-config.yml[]
----
```

[NOTE]

=====

Do not include `link:https://docs.asciidoctor.org/asciidoc/latest/directives/include-lines/[lines by content ranges]`. This approach can lead to content errors when the included file is subsequently updated.

=====

[discrete]

=== Using AsciiDoc callouts in the source code

You can use AsciiDoc callouts in the source code file.

Comment out the callout in the YAML file to ensure that file can still be parsed as valid YAML.

Asciidoctor recognizes the commented callout and renders it correctly in the output.

For example:

```
[source,yaml]
----
apiVersion: v1 # <1>
```

```
// == Indicating Technology Preview features
//
// To indicate that a feature is in Technology Preview, include the `snippets/technology-
// preview.adoc` file in the feature's assembly or module to keep the supportability
// wording consistent across Technology Preview features. Provide a value for the
// `:FeatureName:` variable before you include this module.
//
// [source,text]
// ----
// :FeatureName: The XYZ plug-in
// \include::snippets/technology-preview.adoc[]
// ----
//
// == Indicating deprecated features
//
// To indicate that a feature is deprecated, include the `modules/deprecated-
// feature.adoc` file in the feature's assembly, or to each relevant assembly such as for a
// deprecated Operator, to keep the supportability wording consistent across deprecated
// features. Provide a value for the `:FeatureName:` variable before you include this
// module.
```

== Verification of your content

All documentation changes must be verified by a subject matter expert before merging. This includes executing all procedure changes and confirming expected results. There are exceptions for typo-level changes, formatting-only changes, and other negotiated documentation sets and distributions.

// If a documentation change is due to a bug report or Jira issue, the bug/issue should be put on ON_QA when you have a PR ready. After QE approval is given (either in the bug/issue or in the PR), the QE associate should move the bug/issue status to VERIFIED, at which point the associated PR can be merged. It is also ok for the assigned writer to change the status of the bug/issue to VERIFIED if approval for the changes has been provided in another forum (slack, PR, or email). The writer should indicate that the QE team approved the change as a comment in the bug/issue.

== Images

=== Image format

Use `*.png` format images.

=== Block images

To include a block image (an image on its own line):

1. Put the image file in the ``modules/<module>/images`` folder.

// +

// Ensure that the folder containing your assembly contains an ``images`` symbolic link to the top-level ``images/`` directory, otherwise the image will not be found when building the docs.

2. In the ``.adoc`` content, use this format to link to the image:

+

`image::<module>:<image_filename>[<alt_text>]`

+

Notice the double ``::`` instead of a single ``:``, as seen in inline image usage.

+

.Example

[source,adoc]

`image::documentation:architecture_and_concepts:l1l2.png[L1 to L2 messaging]`

+

The image file, ``l1l2.png``, is in ``modules/architecture_and_concepts/images/``.

=== Inline images

Use this formatting:

`image:<module>:<image_filename>[<alt_text>]`

Note the single ``:`` instead of a double ``::``, as seen in block image usage.

For example:

`image:documentation:architecture_and_concepts:manage-columns.png[title="Manage Columns icon"]`

== Formatting

For all of the system blocks including table delimiters, use four characters. For example:


```
....
|=== for tables
---- for code blocks
....
```

[NOTE]

=====

You can use backticks or other markup in the title for a block, such as a code block ``Example`` or a table ``Description`` title.

=====

=== Code blocks, command syntax, and example output

Code blocks generally show examples of command syntax, example screen output, and configuration files.

The main distinction between showing command syntax and a command example is that a command syntax shows readers how to use the command without real values. An example command, however, shows the command with actual values with an example output of that command, where applicable.

For example:

```
....
Run the following command to initialize an account:
```

```
[source,terminal]
```

```
----
starknet new_account --account <account_name>
----
```

.Example output

```
[source,terminal]
```

```
----
Account address:
0x04e93e1fb507d23b398f0a09f5873d3a7769b0e7ed40dbbe8fe7a2e8ea831006
Public key:
0x07a328511fa8552cd61aaaa89076fe40c3566f4594f29324aa754d41d7c7c55e
Move the appropriate amount of funds to the account, and then deploy the account
by invoking the 'starknet deploy_account' command.
```

NOTE: This is a modified version of the OpenZeppelin account contract. The signature is computed differently.

....

This renders as:

```
> Run the following command to initialize an account:
>
> ----
>starknet new_account --account <account_name>
> ----
>
> .Example output
> ----
> Account address:
0x04e93e1fb507d23b398f0a09f5873d3a7769b0e7ed40dbbe8fe7a2e8ea831006
> Public key:
0x07a328511fa8552cd61aaaa89076fe40c3566f4594f29324aa754d41d7c7c55e
> Move the appropriate amount of funds to the account, and then deploy the account
> by invoking the 'starknet deploy_account' command.
>
> NOTE: This is a modified version of the OpenZeppelin account contract. The
signature is computed
differently.
> ----
```

The following guidelines go into more detail about specific requirements and recommendations when using code blocks:

* If a step in a procedure is to run a command, make sure that the step text includes an explicit instruction to "run" or "enter" the command. In most cases, use one of the following patterns to introduce the code block:

```
** <Step description> by running the following command:
** <Step description> by entering the following command:
** <Step description>, run the following command:
** <Step description>, enter the following command:
```

* Any example of command line input must begin with a prompt, as follows:
** A terminal prompt for a normal user should begin with a dollar sign (`\$`) prompt:

```
+
[source,terminal]
----
$ <regular_user_permission_command_line_input>
----
```

** A terminal prompt for a superuser should begin with a hash symbol (`#`) prompt:
+
[source,terminal]

<superuser_permission_command_line_input>

** A terminal prompt for a command in a non-standard shell, such as a Docker shell, should use the prompt of that shell. For example:

+

[source,terminal]

root@17617744386d:/app# ./player.py

* Avoid using markup in a code block. If you must use any markup in code blocks, see the AsciiDoctor documentation on source blocks and substitutions:

** link:<https://docs.asciidoctor.org/asciidoc/latest/verbatim/source-blocks/>[Source Code Blocks]

** link:<https://docs.asciidoctor.org/asciidoc/latest/subs/>[Substitutions]

+

[CAUTION!]

=====

It can take some trial and error to figure out the correct source block macro to use for the exact markup you want to use.

=====

* For all code blocks, you must include an empty line above a code block (unless that line is introducing block metadata, such as ``[source,terminal]`` for syntax highlighting).

+

Acceptable:

+

....

Lorem ipsum

\$ lorem.sh

....

+

Not acceptable:

+

....

Lorem ipsum

\$ lorem.sh

....

+

Without the line spaces, the content is likely to be not parsed correctly.

* Use `[source,terminal]` for CLI commands, and any other commands that you enter in the terminal, to enable syntax highlighting. Any `[source]` metadata must go on the line directly before the code block. For example:

```
+
....
[source,terminal]
----
$ oc get nodes
----
```

+
If you are also showing a code block for the output of the command, use `[source,terminal]` for that code block as well.

* Use source tags for the programming language used in the code block to enable syntax highlighting. For example:

```
** `[source,cairo]`
** `[source,python]`
** `[source,javascript]`
** `[source,json]`
```

// * Do not use more than one command per code block. For example, the following must be split up into three separate code blocks:

```
// +
// ....
// To create templates you can modify, run the following commands:
//
// [source,terminal]
// ----
// $ oc adm create-login-template > login.html
// ----
//
// [source,terminal]
// ----
// $ oc adm create-provider-selection-template > providers.html
// ----
//
// [source,terminal]
// ----
// $ oc adm create-error-template > errors.html
// ----
// ....
```

* If your command contains multiple lines and uses callout annotations, you must comment out the callout(s) in the codeblock, as shown in the following example:

+

....

To scale based on the percent of CPU utilization, create a `HorizontalPodAutoscaler` object for an existing object:

[source,terminal]

```
$ oc autoscale <object_type>/<name> V/ <1>
--min <number> V/ <2>
--max <number> V/ <3>
--cpu-percent=<percent> <4>
```


<1> Specify the type and name of the object to autoscale.

<2> Optional: Specify the minimum number of replicas when scaling down.

<3> Specify the maximum number of replicas when scaling up.

<4> Specify the target average CPU utilization over all the pods, represented as a percent of requested CPU.

....

* Separate a command and its related example output into individual code blocks. This enables a reader to easily copy the command using the *Copy* button  on docs.starknet.io.

+

In addition, prepend the code block for the output with the title `.Example output` to make it consistently clear across the docs when this is being represented. A lead-in sentence explaining the example output is optional. For example:

+

....

Run the `starknet new_account` command to initialize an account:

[source,terminal]

```
$ starknet new_account --account <account_name>
```

The output verifies that a new account was initialized:

.Example output

[source,terminal]

Account address:

0x04e93e1fb507d23b398f0a09f5873d3a7769b0e7ed40dbbe8fe7a2e8ea831006

Public key:

0x07a328511fa8552cd61aaaa89076fe40c3566f4594f29324aa754d41d7c7c55e

Move the appropriate amount of funds to the account, and then deploy the account by invoking the 'starknet deploy_account' command.

NOTE: This is a modified version of the OpenZeppelin account contract. The signature is computed differently.

....

* To mark up command syntax, use the code block and wrap any replaceable values in angle brackets (`<>`) with the required command parameter, using underscores (`_`) between words as necessary for legibility. For example:

+

....

To deploy the account you initialized, now run the following command:

[source,terminal]

```
$ starknet deploy_account --account=<account_name>
```

....

+

This renders as:

+

--

> To deploy the account you initialized, now run the following command:

> ----

```
> $ starknet deploy_account --account=<account_name>
```

> ----

--

* When referring to a path to a location that the user has selected or created, treat the part of the path that the user chose as a replaceable value. For example:

+

....

Create a secret that contains the certificate and key in the namespace:

[source,terminal]

```
$ oc create secret tls <certificate> --cert=<path_to_certificate>/cert.crt
```

....

+

This renders as:

```

+
--
> Create a secret that contains the certificate and key in the namespace:
>
> ----
> $ oc create secret tls <certificate> --cert=<path_to_certificate>/cert.crt
> ----
--
* If you must provide additional information on what a line of a code block
represents, you can use callouts (`<1>`, `<2>`, etc.) to provide that information.
+
Use this format when embedding callouts into the code block:
+
[subs=-callouts]
....
----
code example 1 <1>
code example 2 <2>
----
<1> A note about the first example value.
<2> A note about the second example value.
....

* If you must provide additional information on what a line of a code block
represents and the use of callouts is impractical, you can use a description list
to provide information about the variables in the code block. Using callouts
might be impractical if a code block contains too many conditional statements to
easily use numbered callouts or if the same note applies to multiple lines of the
codeblock.
+
[source,adoc]
....
----
code <variable_1>
code <variable_2>
----
+
where:

[horizontal]
<variable_1>:: Specifies the explanation of the first variable.
<variable_2>:: Specifies the explanation of the first variable.
....
+
Be sure to introduce the description list with "where:" and start each variable
description with "Specifies."

```

* For long lines of code that you want to break up among multiple lines, use a backslash to show the line break. For example:

+

```
$ oc get endpoints --all-namespaces --template \
  '{{ range .items }}{{ .metadata.namespace }}:{{ .metadata.name }} \
  {{ range .subsets }}{{ range .addresses }}{{ .ip }} \
  {{ end }}{{ end }}{{ "\n" }}{{ end }}' | awk '/ 172\.\.30\./ { print $1 }'
```

* For snippets or sections of a file, use an ellipsis (`...` or `# ...` for YAML) to show that the file continues before or after the quoted block.

+

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
# ...
```

+

Do not use `[...]`, `<<snip>`, or any other variant.

=== Inline code or commands

Do NOT show full commands or command syntax inline within a sentence. The next section covers how to show commands and command syntax.

The only use case for inline commands would be general commands and operations, without replaceables and command options. In this case use back ticks to indicate an inline command. For example:

....

Use the `GET` operation to do x.

....

This renders as:

> Use the `GET` operation to do x.

=== System messages

System messages include error, warning, confirmation, and information messages.

If a message is short enough to include inline, enclose it in back ticks:

....

Previously, image builds and pushes would fail with the `error reading blob from source` error message because the builder logic would compute the contents of new layers twice.

....

This renders as:

> Previously, image builds and pushes would fail with the `error reading blob from source` error message because the builder logic would compute the contents of new layers twice.

If a message is too long to include inline, put it inside a code block with `[source,text]` metadata:

....

Previously, the AWS Terraform provider that the installation program used occasionally caused a race condition with the S3 bucket, and the cluster installation failed with the following error message:

[source,text]

When applying changes to module.bootstrap.aws_s3_bucket.ignition, provider level=error msg="\"aws\" produced an unexpected new value for was present, but now absent.

Now, the installation program uses different AWS Terraform provider code, which now robustly handles S3 eventual consistency, and the installer-provisioned AWS cluster installation does not fail with that error message.

....

This renders as:

> Previously, the AWS Terraform provider that the installation program used occasionally caused a race condition with the S3 bucket, and the cluster installation failed with the following error message:

>

> ----

> When applying changes to module.bootstrap.aws_s3_bucket.ignition, provider level=error msg="\"aws\" produced an unexpected new value for was present, but now absent.

> ----

>

> Now, the installation program uses different AWS Terraform provider code, which now

robustly handles S3 eventual consistency, and the installer-provisioned AWS cluster installation does not fail with that error message.

NOTE: Always refer to a message with the type of message it is, followed by the word `_message_`. For example, refer to an error message as an `_error message_`, and not simply as an `_error_`.

=== Lists

Write numbered lists as shown in this example:

```
....
. Item 1 (2 spaces between the period and the first character)

. Item 2

. Item 3
....
```

This renders as:

```
> . Item 1
> . Item 2
> . Item 3
```

If you must add any text, admonitions, or code blocks you have to add the ``+`` below the line to indicate continuation. For example:

```
....
. Item 1
+
----
some code block
----

. Item 2

. Item 3
....
```

This renders as:

```
> . Item 1
> +
> ----
> some code block
> ----
```

> . Item 2
> . Item 3

=== Footnotes

Avoid footnotes when possible.

If you reference a footnote from only a single location, use the following syntax:

```
.Footnote
....
footnote:[This is the footnote text.]
....
```

If you reference a footnote from multiple locations, set an attribute with the footnote text. As a consequence, this will duplicate the footnote text at bottom of the page.

```
.Footnote with text set by an attribute
....
:note-text: This is a footnote.
```

This text has a footnote qualifier attached footnote:[{note-text}].

But this other text uses the same qualifier elsewhere footnote:[{note-text}].
....

```
[id="collapsible-content"]
=== Collapsible content
```

You can collapse sections of content by using the `collapsible` option, which converts the Asciidoctor markup to HTML `details` and `summary` sections. The `collapsible` option is used at the writer's discretion and is appropriate for considerably long code blocks, lists, or other such content that significantly increases the length of a topic.

[NOTE]

```
=====
You must set a title for the `summary` section. If a title is not set, the default title is
"Details."
=====
```

Collapsible content is formatted as shown:

```
....
.Title of the `summary` dropdown
[%collapsible]
=====
This is content within the `details` section.
```

====

....

This renders as a dropdown with collapsed content:

.Title of the `Summary` dropdown

[%collapsible]

====

This is content within the `Details` section.

====

If your collapsible content includes an admonition such as a note or warning, you must nest the admonition:

....

.Collapsible content that includes an admonition

[%collapsible]

====

This content includes an admonition.

[source,terminal]

\$ oc whoami

[NOTE]

=====

Nest admonitions when using the `collapsible` option.

=====

====

....

This renders as:

.Collapsible content that includes an admonition

[%collapsible]

====

This content includes an admonition.

[source,terminal]

\$ oc whoami

[NOTE]

=====

Nest admonitions when using the `collapsible` option.

```
=====  
=====
```

=== Quick reference

.User accounts and info

[option="header"]

|=====

|Markup in command syntax |Description |Substitute value in Example block

|`<username>`

|Name of user account

|user@example.com

|`<password>`

|User password

|password

|=====

.Projects and applications

[option="header"]

|=====

|Markup in command syntax |Description |Substitute value in Example block

|`<project>`

|Name of project

|myproject

|`<app>`

|Name of an application

|myapp

|=====

=== Additional resources sections

The following guidelines apply to all "Additional resources" sections:

- * Avoid including paragraphs in the section. Use an unordered list.
- * The links and xrefs in the unordered list must contain human-readable text between the square brackets.
- * Each item in the unordered list must contain a minimum of text besides the link or xref.
- * Use `.Additional resources` formatting for an Additional resources section when it applies to a section within a topic. For example:

+

[source,adoc]

.Additional resources

* Use `` formatting for the section heading (`` Additional resources`) when it applies to the entire topic. For example:

+

[id="additional-resources_configuring-alert-notifications"]
== Additional resources

* link:some-url.com[Human readable label]
* xref:some_xref[Human readable label]
* xref:some_other_xref[Human readable label]

== Admonitions (Notes, Tips, Cautions...)

Format admonitions, such as notes and warnings, as follows:

....

[ADMONITION]

=====

Text for admonition

=====

....

For a list of available admonition types, see link:<https://redhat-documentation.github.io/supplementary-style-guide/#admonitions>[Admonitions] in the _Red Hat supplementary style guide for product documentation_.

[#peer_review]

== Peer reviews

* Easy fixes: For simple fixes such as fixing typos, any writer with merge permissions can go ahead and merge those fixes without any review.

* New features: When documenting new features or significant changes to existing functionality, you must have a peer review prior to merging.

* Outdated content that should be updated as quickly as possible: You should get a peer review, but if the assigned reviewer does not review it within ~48 hours, then you can merge it with technical updates and follow up with the peer review as soon as possible.

[id="setting_up_environment"]
= Setting up your environment

:icons:

:toc: macro

:toc-title:
:toclevels: 1
:linkattrs:
:description: How to set up your environment to contribute

toc::[]

== Create a GitHub account

Before you can contribute to Starknet documentation, you must
<https://www.github.com/join>[sign up for a GitHub account].

== Set up authentication

When you have your account set up, follow the instructions to
<https://help.github.com/articles/generating-ssh-keys/>[generate and set up SSH
keys on GitHub] for proper authentication between your workstation and GitHub.

Confirm authentication is working correctly with the following command:

```
----  
$ ssh -T git@github.com  
----
```

== Fork and clone the Starknet documentation repository

For significant contributions, set up the Starknet documentation repository on your
workstation so that you can create PRs and contribute.
You only have to perform these steps during the initial setup.

=== Procedure

. Fork the <https://github.com/starknet-io/starknet-docs> repository to your
GitHub account from the GitHub UI. You can do this by clicking on **Fork** in the
upper right-hand corner.

. In the terminal on your workstation, change into the directory where you want
to clone the forked repository.

. Clone the forked repository onto your workstation with the following
command, replacing `<user_name>` with your actual GitHub username.

```
+  
----  
$ git clone git@github.com:<user_name>/starknet-docs.git  
----
```

. Change into the directory for the local repository you just cloned.

+

```
----  
$ cd starknet-docs  
----
```

. Add an upstream pointer back to Starknet's remote repository, in this case `_starknet-docs_`.

+

```
----  
$ git remote add upstream git@github.com:starknet-io/starknet-docs.git  
----
```

This ensures that you are tracking the remote repository so you can keep your local repository synchronized.

[#previewing_the_website_on_your_local_machine]
== Previewing the website on your local machine

You don't need to install Antora to preview the website. This repo includes an embedded Antora installation, which is referred to in the Antora documentation as a `_local installation_`.

=== Prerequisites

You must have an active Node.js LTS release on your machine. See the `_Antora documentation_` for your OS:

- * <https://docs.antora.org/antora/latest/install/linux-requirements/#node>[Linux]
- * <https://docs.antora.org/antora/latest/install/windows-requirements/#node>[Windows]
- * <https://docs.antora.org/antora/latest/install/macos-requirements/#node>[macOS]

[id="http_server"]
[TIP]

=====

Install the optional link: <https://docs.antora.org/antora/latest/preview-site/#run-a-local-server-optional>[http server Antora provides] so you can view the output in a local web server

=====

[#git_related_info]
== Git-related info

- * When creating a new PR, tag one or more of the following as reviewers: ``stoobie``, ``ArielElp``, ``LucasLvy``.
- * When creating a new PR, you need to merge to the ``main`` branch.

[id="overview"]

= Starknet documentation supplementary style guide

:toc:

:title: Starknet documentation supplementary style guide

:description: Style guidance for writing Starknet technical documentation

This guide provides style guidelines and preferred term usage for [link:https://starknet.io/](https://starknet.io/) [the Starknet website], including [link:http://docs.starknet.io/docs.starknet.io](http://docs.starknet.io/docs.starknet.io). Use it as a supplement to the following style guides:

[#how_to_use_this_guide]

== How to use this guide

When looking for style guidance and writing guidance, use these guides in the following order:

[horizontal]

Starknet documentation supplementary style guide:: Reference this guide first. It provides guidance that is specific to Starknet documentation. This guidance either supplements or, in specific cases, overrides the other style guides.

[link:https://redhat-documentation.github.io/supplementary-style-guide/](https://redhat-documentation.github.io/supplementary-style-guide/)[_Red Hat supplementary style guide for product documentation_]:: Provides additional guidance or overrides guidance in the _Google developer documentation style guide_.

[link:https://developers.google.com/style/](https://developers.google.com/style/)[_Google developer documentation style guide_]:: The primary source of style guidance for Starknet documentation.

If you cannot find helpful information or if you must deviate from the guidance in any of these guides, <https://github.com/starknet-io/starknet-docs-style-guide/issues> [open an issue] for this style guide. The stakeholders can then discuss and determine how to address the issue.

[[_related_guides]]

== Related guides

The following reference guides for technical writers provide technical information on AsciiDoc, the markup language we use to author the Starknet technical documentation:

[horizontal]

[link:https://docs.asciidoctor.org/asciidoc/latest/](https://docs.asciidoctor.org/asciidoc/latest/)[_AsciiDoc Language Documentation_]:: documentation for the AsciiDoc language as it's implemented in AsciiDoctor.

[link:https://redhat-documentation.github.io/asciidoc-markup-conventions/](https://redhat-documentation.github.io/asciidoc-markup-conventions/)[_AsciiDoc Mark-up Quick Reference_]:: Guidance specific to writing in AsciiDoc. Includes links to complete documentation for AsciiDoc and AsciiDoctor.

[#titles_and_section_headings]

== Titles and section headings

=== Guidelines for writing headings

- * Use sentence case in all titles and section headings. See <http://www.titlecase.com/> or <https://convertcase.net/> for a conversion tool.
- * Be as descriptive as possible with the title or section headings without making them unnecessarily long.
- * For procedural topics, use a gerund form in headings, such as:
 - ** Creating
 - ** Managing
 - ** Using
- * For conceptual topic titles, see [https://developers.google.com/style/headings\[Headings and titles\]](https://developers.google.com/style/headings[Headings and titles]) in the `_Google developer documentation style guide_`.
- * For Reference topic titles, use a title beginning with a noun that describes the content, such as:
 - ** `_Compiler command reference_`
 - ** The name of a command/programming element, such as ``_starknet get_block_``.
 - ** `_System calls_`
- ** For more examples, see link:[https://redhat-documentation.github.io/modular-docs/#modular-docs-reference-examples\[Reference module examples\]](https://redhat-documentation.github.io/modular-docs/#modular-docs-reference-examples[Reference module examples]) in the `_Modular documentation reference guide_`.

=== Guidelines for headings as structural elements

- * Use only one level 1 heading (`=`) in any file.
 - * For subsequent heading levels, avoid using a single heading for each level. For example, if you have one heading 2 you should add another heading 2.
- +
- Headings generally separate ideas on a page, so usually a heading indicates a new idea, which logically presupposed a heading for the first idea. Sometimes implementing this guideline requires restructuring the page.

.Example of what to do

- * H1: Applying to join the bar
- * H2: Pass law school
- * H3: Study all day
- * H3: Study all night
- * H2: Pass the bar exam
- * H3: Learn all the laws
- * H3: Register for the exam

.Example of what `_not_` to do

- * H1: Applying to join the bar
- * H2: Pass the bar exam //Notice this is the only H2

- * H3: Pass law school //Notice this is the only H3
- * H4: Study all day
- * H4: Study all night
- * H4: Learn all the laws
- * H4: Register for the exam

== Links to external websites

If you want to link to a third-party website:

- * Do not create a hyperlink.
- * Use the site top level URL as text.
- * Provide the title of the page to search for on the site.

.Example

[source,adoc]

For more information, see `_A specific page_` at `\http://www.example.com/`.

A hyperlink to a page on a third-party website is convenient and user-friendly `_as long as the link works_`. The problem is that a third-party site can move pages without notification, in which case that user-friendly link can become a user-unfriendly broken link, and broken links also impact our search engine rankings.

[#glossary]

== Glossary

Deprecated:: Refers to a feature or capability that is still supported, but support will be removed in a future release of Starknet.

Future fixes or enhancements are unlikely. If necessary, an alternative is available.

Fri:: The smallest unit of the Starknet native token, STRK, equal to 10^{-18} STRK.

G-fri:: 1,000,000,000 fries.

Removed:: Refers to a feature or capability that has been entirely removed.

Unsupported:: Refers to a feature or capability that is no longer supported.

[#word_list]

== Word list

If a term doesn't appear here, refer to the following guides, in order:

. link:<https://redhat-documentation.github.io/supplementary-style-guide/#glossary-terms-conventions>[Glossary of terms and conventions] in the `_Red Hat supplementary style guide for product documentation_`.

. link:<https://developers.google.com/style/word-list>[Word list] in the _Google developer documentation style guide_.

[#E]

=== E

EIP-_ num _, ERC-20::

Correct form: EIP-_ num _

+

Example: EIP-4844

+

Incorrect forms: EIP4844, EIP 4844

+

Reasoning: This is the convention used on ethereum.org.

[#F]

=== F

fri (10^{18} STRK)::

Correct forms

+

--

* *Singular:* Fri

* *Plural:* Fries

--

+

Usage rule

+

Normal word casing, so use _Fri_ at the beginning of a sentence, and _fri_ after the first word of a sentence.

+

Examples

+

* Alice holds 5 million fries.

* Fri is the smallest unit of STRK.

[#G]

=== G

G-fri (1,000,000,000 fries)::

Correct forms

+

--

* *Singular:* G-fri

* *Plural:* G-fries

--

+

Usage rule

+

Normal word casing, so use _G-fri_ at the beginning of a sentence, and _g-fri_ after the first word of a sentence.

+

Examples

* Alice holds 5 million g-fries.

* G-fri is a unit that is equal to one billion fries.

[#O]

=== O

offchain::

Correct form: offchain

+

Incorrect forms: off-chain, off chain

+

Reasoning: Align with trending industry usage.

onchain::

Correct form: onchain

+

Incorrect forms: on-chain, on chain

+

Reasoning: Align with trending industry usage.

[#S]

=== S

Starknet Core Contract::

Correct form: Starknet Core Contract

+

Incorrect forms: Starknet core contract, Starknet Core contract

[#T]

=== T

transaction::

Correct form: transaction, transactions

+

Avoid: tx, txs

+

**Reasoning:* Although this abbreviation is well known in the industry, we avoid abbreviations. Abbreviations present a barrier to first-time readers and can also interfere with localization.

```
const algoliasearch = require("algoliasearch");
const fs = require("fs");
const path = require("path");
const simpleGit = require("simple-git");
const AsciiDoctor = require("asciidoctor");
const asciidoctor = AsciiDoctor();
const { convert } = require("html-to-text");
const git = simpleGit(__dirname);
```

```
function resolvePath(...args) {
  return path.resolve(__dirname, ...args);
}
```

```
if (process.argv.length < 5) {
  throw new Error("Failed to find proper settings for Algolia indexing");
}
const algoliaAppId = process.argv[2];
const algoliaApiKey = process.argv[3];
const algoliaIndexNamePrefix = process.argv[4];
```

```
if (!algoliaAppId || !algoliaApiKey || !algoliaIndexNamePrefix) {
  throw new Error("Please setup Algolia settings in GitHub Action Secrets");
}
```

```
const initBranch = async () => {
  const status = await git.status();
  let currentBranch = status.tracking;
  currentBranch = currentBranch.split("/")[1];
  console.log(`Indexing articles on branch "${currentBranch}" ...`);
  startIndexing(currentBranch);
};
```

```
initBranch();
```

```
const startIndexing = (currentBranch) => {
  const client = algoliasearch(algoliaAppId, algoliaApiKey);
  const algoliaIndex = client.initIndex(
    `${algoliaIndexNamePrefix}-${currentBranch}`,
  );
  const commonPath = resolvePath("../", "components");
```

```
fs.readdir(commonPath, "utf8", (err, data) => {
  if (err) {
```

```

    console.error(err, "err");
    return;
  }
  data.forEach((item) => {
    const nextPath = path.join(commonPath, item, "modules");
    fs.readdir(nextPath, "utf8", (errNext, dataNext) => {
      if (errNext) {
        console.error(errNext, "err_next");
        return;
      }
      dataNext.forEach((listItem) => {
        const pagePathFold = path.join(nextPath, listItem, "pages");
        fs.readdir(pagePathFold, "utf8", (pageErr, pageData) => {
          if (pageErr) {
            console.error(pageErr, "pageErr");
            return;
          }
          pageData.forEach((target) => {
            const targetPath = path.join(pagePathFold, target);
            const stat = fs.lstatSync(targetPath);
            if (stat.isDirectory()) {
              fs.readdir(targetPath, "utf-8", (fileErr, fileData) => {
                if (fileErr) {
                  console.log(fileErr, "fileErr");
                  return;
                }
                fileData.forEach((targetFile) => {
                  const targetFilePath = path.join(targetPath, targetFile);
                  beforeUpload(targetFilePath, targetFile);
                });
              });
            }
            beforeUpload(targetPath, target);
          });
        });
      });
    });
  });
});

```

```

function beforeUpload(targetPath, target) {
  const stat = fs.lstatSync(targetPath);
  if (stat.isFile() && path.extname(target) === ".adoc") {
    fs.readFile(targetPath, "utf-8", (err, data) => {
      if (err) {
        console.error(err, "beforeUpload");
        return;
      }
    });
  }
}

```

```

    }
    const titleFromAscii =
      data
        .split("\n")
        .find((str) => str.slice(0, 2) === "= ")
        ?.split("=")[1] ?? "";
    const titleFromMK =
      data
        .split("\n")
        .find((str) => str.slice(0, 2) === "# ")
        ?.split("#")[1] ?? "";
    const title = (titleFromAscii || titleFromMK).trim();

    const html = asciidoctor.convertFile(targetPath, {
      to_file: false,
      standalone: true,
    });
    const text = convert(html, {
      wordwrap: 130,
    });
    const recode = {
      objectID: targetPath.substring(targetPath.indexOf("modules") + 7),
      title: title,
      content: text,
    };
    uploadFile(recode, targetPath);
  });
}
}

function uploadFile(file, targetPath) {
  const url = targetPath
    .split("modules")[1]
    .replace("/pages", "")
    .replace(".adoc", "")
    .replace("ROOT/index", "")
    .replace("index", "");
  const record = {
    url: "https://docs.starknet.io/documentation" + url,
    ...file,
  };
  algoliaIndex.saveObject(record).wait();
  console.log("Done indexing ==>", url);
  console.log("Saved record title ==>", record.title);
  console.log("Saved record url ==>", record.url);
}

```



```

});
};
"use strict";

/* Copyright (c) 2022 OpenDevise, Inc.
 *
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License Version 2.0. If a copy of this license was not distributed
 * with this file, you can obtain one at http://mozilla.org/MPL/2.0/.
 */
const { promises: fsp } = require("fs");
const ospath = require("path");

/**
 * Rewrites local content sources to support the use of linked worktrees.
 *
 * @author Dan Allen <dan@opendevise.com>
 */
module.exports.register = function () {
  this.once("playbookBuilt", async ({ playbook }) => {
    const expandPath = this.require("@antora/expand-path-helper");
    for (const contentSource of playbook.content.sources) {
      const { url, branches } = contentSource;
      if (url.charAt() !== ".") continue;
      const absdir = expandPath(url, { dot: playbook.dir });
      const gitfile = ospath.join(absdir, ".git");
      if (
        await fsp.stat(gitfile).then(
          (stat) => !stat.isDirectory(),
          () => false,
        )
      ) {
        const worktreeGitdir = await fsp
          .readFile(gitfile, "utf8")
          .then((contents) => contents.trimRight().substr(8));
        const worktreeBranch = await fsp
          .readFile(ospath.join(worktreeGitdir, "HEAD"), "utf8")
          .then((contents) =>
            contents.trimRight().replace(/^ref: (?:(refs\/heads\/)?/, ""),
          );
        const reldir = ospath.relative(
          playbook.dir,
          await fsp
            .readFile(ospath.join(worktreeGitdir, "commondir"), "utf8")
            .then((contents) => {
              const gitdir = ospath.join(worktreeGitdir, contents.trimRight());

```

```

        return ospath.basename(gitdir) === ".git"
        ? ospath.dirname(gitdir)
        : gitdir;
    }},
  );
  contentSource.url = reldir ? `${ospath.sep}${reldir}` : ".";
  if (!branches) continue;
  contentSource.branches = (
    branches.constructor === Array ? branches : [branches]
  ).map((pattern) => pattern.replaceAll("HEAD", worktreeBranch));
}
}
});
};
// The tokenization used by lunr by default doesn't work well with code.
// We extend the set of characters considered separators to include
// parentheses and commas.

module.exports.register = () => {
  const lunr = require("lunr");
  lunr.tokenizer.separator = /\s\-\(\),\+\;/;
};
{
  "name": "starknet-docs",
  "version": "0.1.514",
  "private": true,
  "scripts": {
    "pre-release": "standard-version --prerelease --skip.changelog --releaseCommitMessageFormat 'chore(prerelease): {{currentTag}}'",
    "release": "standard-version",
    "release:dry": "standard-version --dry-run",
    "generate": "antora playbook.yml"
  },
  "devDependencies": {
    "@antora/cli": "3.1.1",
    "@antora/site-generator": "3.1.1"
  },
  "dependencies": {
    "@antora/lunr-extension": "^1.0.0-alpha.8",
    "@asciidoctor/tabs": "^1.0.0-beta.3",
    "algoliasearch": "^4.14.2",
    "asciidoctor": "^2.2.6",
    "html-to-text": "^8.2.1",
    "lunr": "^2.3.9",
    "simple-git": "^3.15.0",
    "standard-version": "^9.3.2"
  }
}

```

```

    }
  }
  #!/bin/bash
  http-server . -c-1

  /* Styling for JSON container */
  #jsonContainer {
    font-family: monospace;
    margin: 20px;
  }

  /* Styling for JSON keys */
  .json-key {
    font-weight: bold;
    cursor: pointer;
  }

  /* Styling for collapsible sections */
  .json-section {
    margin-left: 20px;
    padding-left: 10px;
    border-left: 1px solid #ccc;
    cursor: pointer;
  }

  .json-content {
    display: none;
  }

  .json-section.collapsed .json-content {
    display: none;
  }

  .json-section.collapsed .json-toggle {
    transform: rotate(-45deg);
  }
  function createJsonElement(data, indent) {
    const type = typeof data;

    if (Array.isArray(data)) {
      const element = document.createElement("div");
      element.classList.add("json-array");

      if (indent > 0) {
        element.classList.add("json-indent");
      }
    }
  }

```

```

data.forEach((item) => {
  const itemElement = createJsonElement(item, indent + 1);
  element.appendChild(itemElement);
});

return element;
} else if (type === "object" && data !== null) {
  const element = document.createElement("div");
  element.classList.add("json-object");

  if (indent > 0) {
    element.classList.add("json-indent");
  }

  Object.keys(data).forEach((key) => {
    const value = data[key];

    if (key === "name") {
      const keyElement = document.createElement("span");
      keyElement.classList.add("json-key");
      keyElement.textContent = `${key}: `;

      const valueElement = createJsonElement(value, indent + 1);

      const sectionElement = document.createElement("div");
      sectionElement.classList.add("json-section");
      sectionElement.appendChild(keyElement);
      sectionElement.appendChild(valueElement);

      sectionElement.addEventListener("click", () => {
        sectionElement.classList.toggle("collapsed");
      });

      element.appendChild(sectionElement);
    } else {
      const nestedElement = createJsonElement(value, indent + 1);
      element.appendChild(nestedElement);
    }
  });

  return element;
} else {
  const element = document.createElement("span");
  element.classList.add("json-value");

```

```

    if (type === "string") {
        element.classList.add("token", "string");
        element.textContent = `"${data}"`;
    } else if (type === "number") {
        element.classList.add("token", "number");
        element.textContent = data.toString();
    } else if (type === "boolean") {
        element.classList.add("token", "boolean");
        element.textContent = data.toString();
    } else if (type === "null") {
        element.classList.add("token", "null");
        element.textContent = "null";
    }

    return element;
}

function fetchAndDisplayJSON() {
    var url =
        "https://raw.githubusercontent.com/starkware-libs/starknet-specs/master/api/
starknet_api_openrpc.json?token=ASNPSF625O5JPOCD2SYH5D3BO3AAM&uiSchem
a%5BappBar%5D%5Bui:input%5D=false&uiSchema%5BappBar%5D%5Bui:darkMode
%5D=true&uiSchema%5BappBar%5D%5Bui:examplesDropdown%5D=false";

    fetch(url)
        .then((response) => {
            if (!response.ok) {
                throw new Error("Error fetching JSON: " + response.status);
            }
            return response.json();
        })
        .then((jsonData) => {
            var jsonContainer = document.getElementById("jsonContainer");
            var methodsData = jsonData.methods || {};

            jsonContainer.innerHTML = "";
            jsonContainer.appendChild(createJsonElement(methodsData, 0));
        })
        .catch((error) => {
            console.error(error);
        });
}

// Call the fetchAndDisplayJSON function when the page has loaded
document.addEventListener("DOMContentLoaded", fetchAndDisplayJSON);

```

