abaplint.app

Documentation

Table of Contents

Home	https://github.com/heliconialabs/docs.abaplint.app
License	Creative Commons
Build	2023-12-01 07:27:41 UTC

1. Introduction

abaplint.app is a cloud based tool for helping ABAP developers develop ABAP. The app can be used for static analysis and continuous integration, providing multiple configurable rules to support any development guidelines.

1.1. Requirements

A GitHub user or organization is required for use with the abaplint.app. The latest version of abapGit is required for serializing the ABAP artifacts, abapGit works on SAP_BASIS 702 and up.

1.2. Support

Users with paid plans can contact support@abaplint.app with questions, bugs and feature requests.

2. Quick Start

Perform the following steps to get started,

1. Create GitHub User or Organization
2. Install the app from the marketplace
3. Install abapGit in the ABAP system
4. Push ABAP code to GitHub via abapGit

abaplint.app is automatically triggered for each commit, or when a pull request is opened.

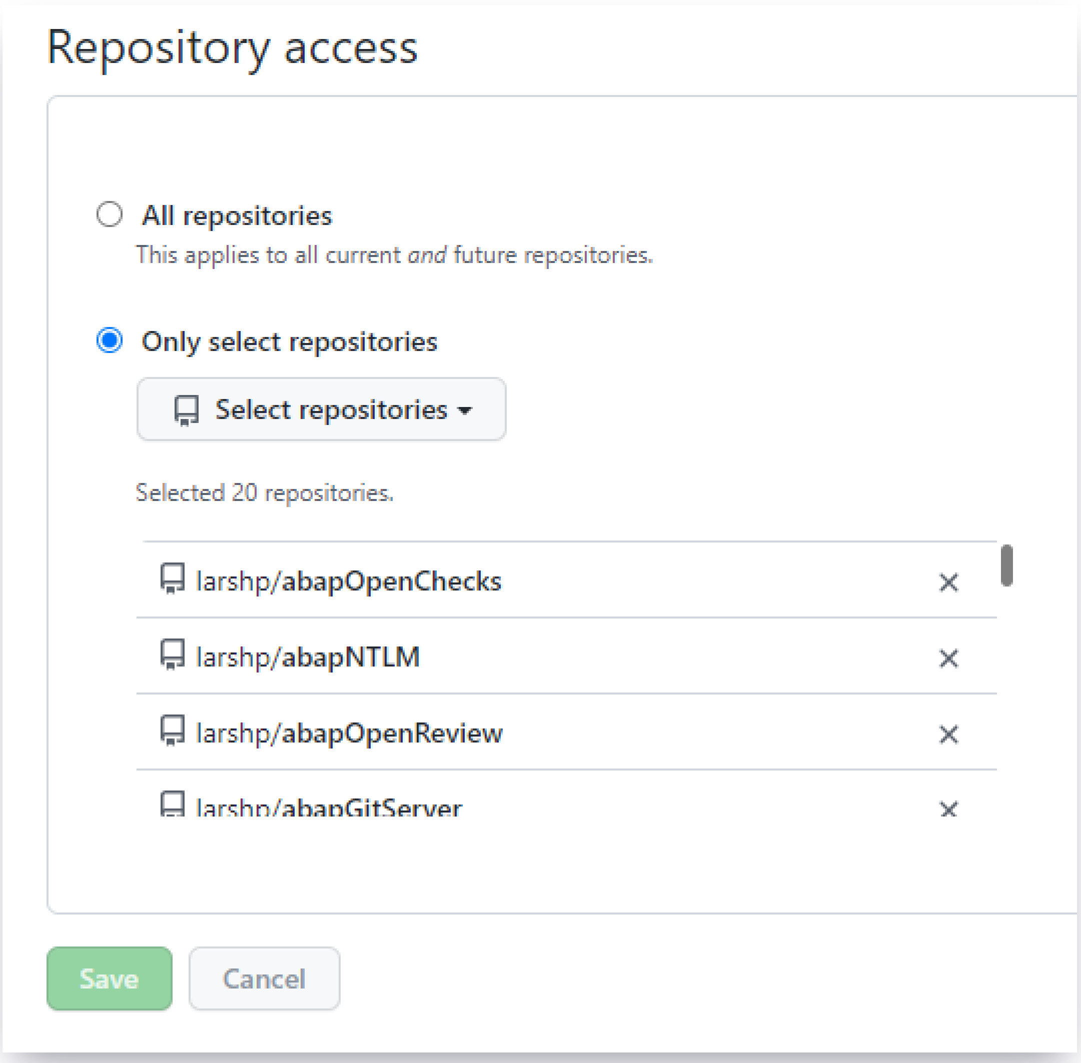
By default, abaplint will run all rules if there is no configuration found. Check the abaplint documentation for a minimal syntax checking configuration.

3. Configuration

3.1. Repository Access

The administrator of a GitHub organization controls which repositories that abaplint.app can access via https://github.com/apps/abaplint/installations/new.

Only grant access to repositories where abaplint.app will be used,



Repository access

All repositories
This applies to all current *and* future repositories.

[✓ Only select repositories]

Selected 20 repositories.

- larshp/abapOpenChecks
- larshp/abapNTLM
- larshp/abapOpenReview
- larshp/abapGitServer

Administrators can revoke or grant additional repository access at any time, note that the app permissions are limited to read and updating information in pull requests.

3.2. Rule Configuration

abaplint is configured via `/abaplint.json` in the git repository, if no configuration file is present, the default configuration is used, which have all rules enabled.

The default configuration can be found on playground.abaplint.app, or by running `abaplint -d`on command line.

If editing the configuration file via vscode abaplint extension, the editor will know the format of the json configuration file and help discovering and updating rule configurations.

rules.abaplint.org lists and documents all rules which can be configured.

3.3. App Configuration

A file with one of the following filenames can optionally be placed in the root of the git directory,

- `abaplint-app.json`
- `abaplint-app.jsonc`
- `abaplint-app.json5`

Only the first file found will be taken into account.

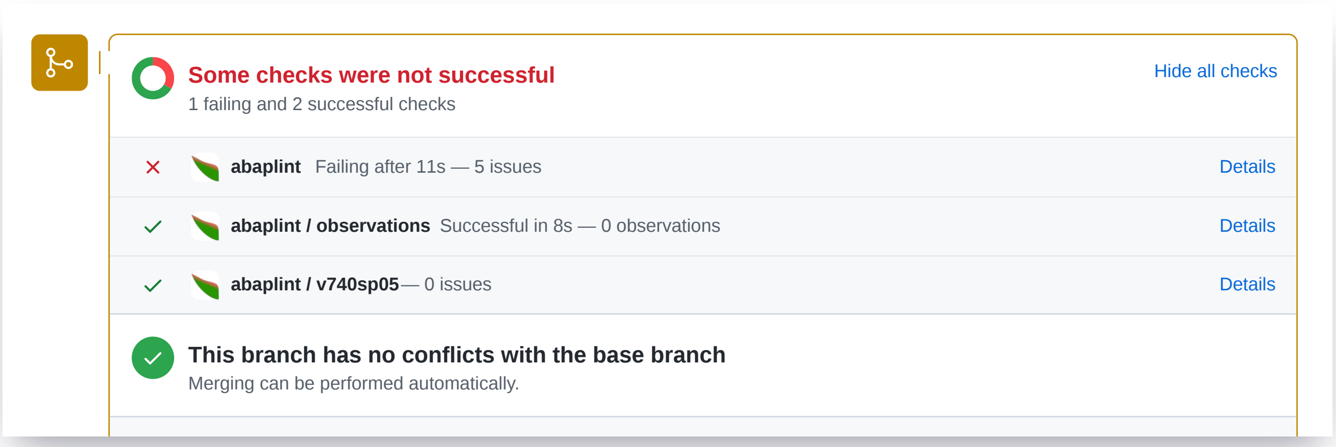
The file can be used to configure abaplint.app features. The corresponding schema can be accessed at https://schema.abaplint.app/schema.json.

3.3.1. Multiple Configurations

By adding a `abaplint-app.json` file it is possible to have abaplint.app run multiple abaplint configurations. Observations and suggestions are reported only for the first configuration. Configurations are processed in the sequence as defined in the file,

```
{
  "configurations": {
    "default": {
      "filename": "./abaplint.jsonc"
    },
    "v740sp05": {
      "filename": "./abaplintonprem.jsonc"
    }
  }
}
```

When running with the above configuration, abaplint.app also analyzes the source code using `abaplintonprem.jsonc` and issues the result as "abaplint / v740sp05",



Some checks were not successful
1 failing and 2 successful checks

Hide all checks

✗	**abaplint** Failing after 11s — 5 issues	Details
✓	**abaplint / observations** Successful in 8s — 0 observations	Details
✓	**abaplint / v740sp05**— 0 issues	Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

3.3.2. Experimental Features

Experimental features are disabled by default, but can be enabled by setting

```
{
  "experimentalFeatures": true
}
```

These features change over time and are only documented when the feature becomes generally available.

3.3.3. Allowing no artifacts

By default, abaplint.app will report an error if no ABAP artifacts are found, this error can be disabled with:

```
{
  "noArtifactsOkay": true
}
```

3.4. Dependent GitHub Actions

abaplint.app helps with static analysis, the next step in a CI pipeline is typically to run the unit tests. Github Actions can be used to trigger any logic after the static analysis completes.

Template and working example can be found at https://github.com/heliconialabs/abaplint-app-dependent-action

Example Github Action configuration,

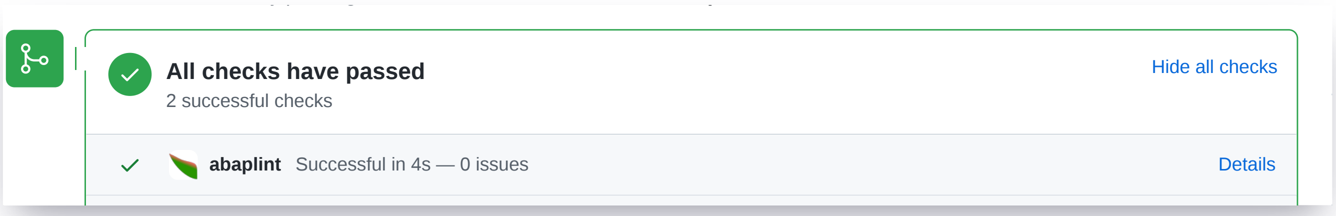
```
name: unit

on:
  pull_request

jobs:
  wait:
    runs-on: ubuntu-latest
    timeout-minutes: 5
    steps:
     - name: Wait for other checks to succeed
       uses: lewagon/wait-on-check-action@v1.1.1
       with:
         ref: ${{ github.event.pull_request.head.sha }}
         check-name: abaplint
         repo-token: ${{ secrets.GITHUB_TOKEN }}
         wait-interval: 1
  unit:
    needs: wait
    runs-on: ubuntu-latest
    steps:
    - name: Echo github context
      run: echo '${{ toJson(github.event) }}' || true
```

4. Annotations

After installation and configuration of abaplint.app, it will report findings for each commit.

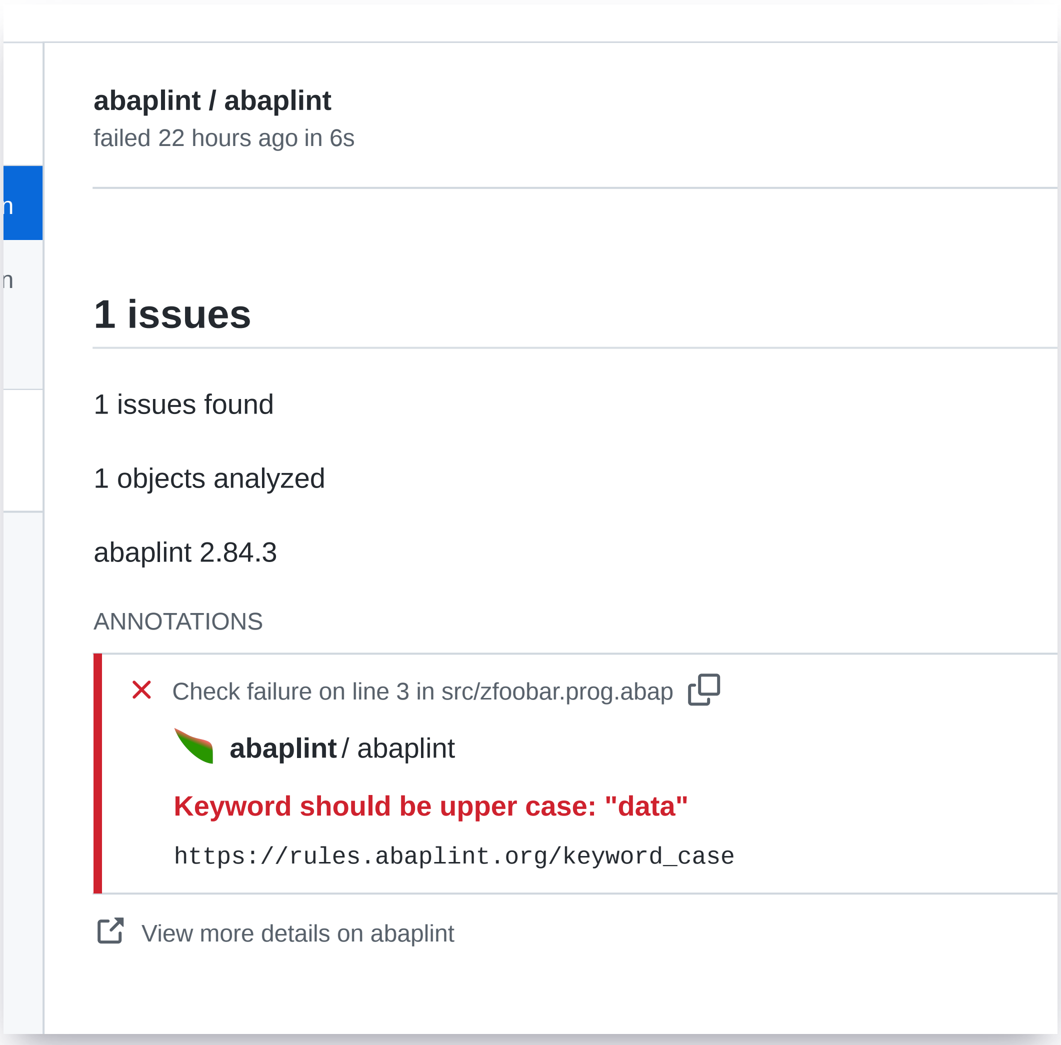


All checks have passed
2 successful checks

Hide all checks

✓ **abaplint** Successful in 4s — 0 issues Details

If issues are found, the "Details" link can be clicked, and the developer can check the findings directly in the pull request,



abaplint / abaplint
failed 22 hours ago in 6s

1 issues

1 issues found

1 objects analyzed

abaplint 2.84.3

ANNOTATIONS

Check failure on line 3 in src/zfoobar.prog.abap

abaplint / abaplint

Keyword should be upper case: "data"

```
https://rules.abaplint.org/keyword_case
```

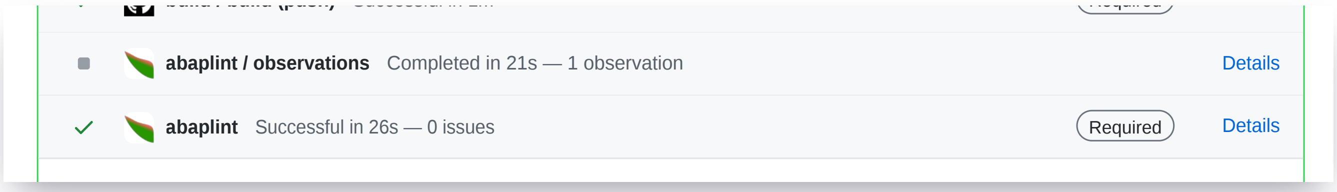
View more details on abaplint

Annotations are updated after each push, and links directly to the branch which contains the change.

Additional information regarding the specific rules can be looked up on https://rules.abaplint.org

5. Observations

Observations are run only in the context of a pull request, and reports only "success" or "neutral" run status, so it will not block merges in case required checks are setup for branch protection.



abaplint / observations Completed in 21s — 1 observation Details

✓ abaplint Successful in 26s — 0 issues Required Details

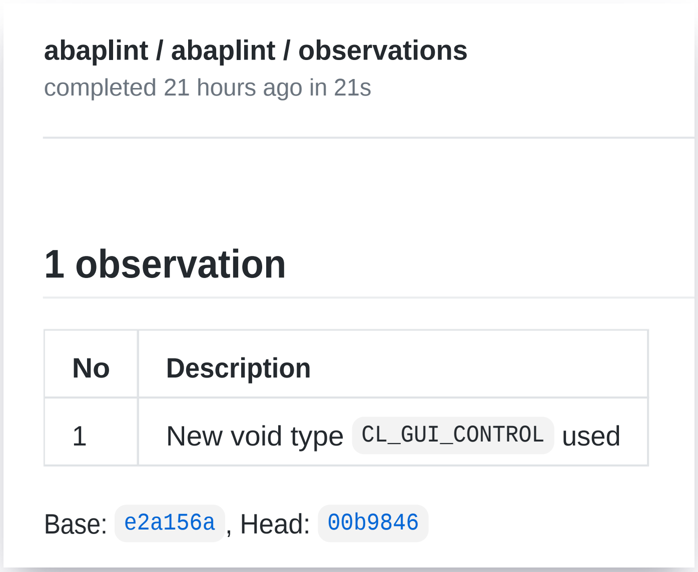
If there are zero observations the step reports success, if some are found it reports the neutral status. A neutral status is shown in the pull request, which help s the reviewer to identify and check the observations.

For running examples see pull requests for the following repositories, https://github.com/heliconialabs?q=observation

5.1. New void type used

If abaplint is configured with `errorNamespace` some types are trusted by abaplint to be correct in all uses, these types are called void types.

This observation reports findings if new void types are introduced by the pull request, this helps identifying if the footprint towards eg. dependencies or the standard system increases.



abaplint / abaplint / observations
completed 21 hours ago in 21s

1 observation

No	Description
1	New void type `CL_GUI_CONTROL` used

Base: `e2a156a`, Head: `00b9846`

5.2. DB field changed

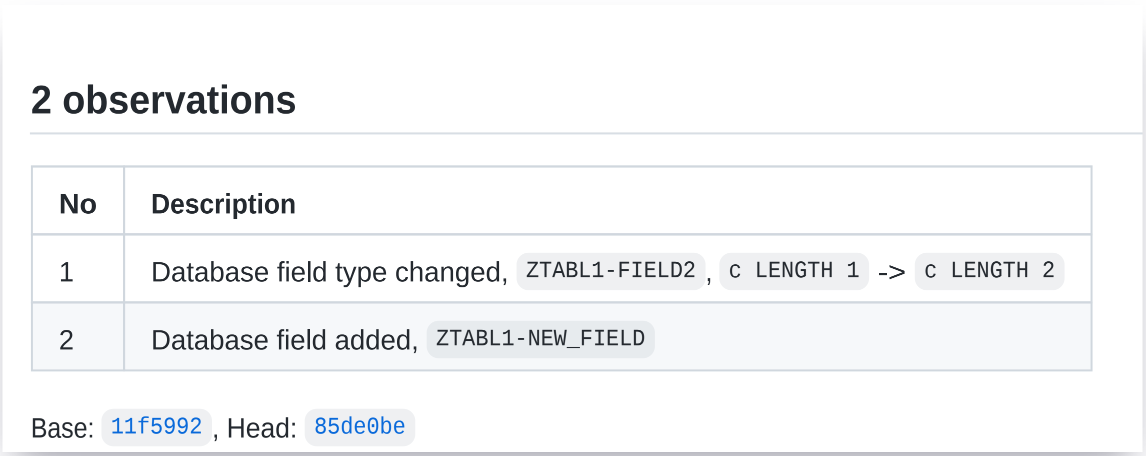
Changing structures, data elements or domains can cause transparent database tables to be changed, this observation helps making these changes visible to reviewers.

This observation is reported if:

- If the type of a field is changed
- If a field is added
- If a field is removed

If a key field is added

- If a key field is removed



2 observations

No	Description
1	Database field type changed, `ZTABL1-FIELD2`, `C LENGTH 1` -> `C LENGTH 2`
2	Database field added, `ZTABL1-NEW_FIELD`

Base: `11f5992`, Head: `85de0be`

5.3. Target Rules

Rules configured as `targetRules` will show up as observations, see the chapter on Target Rules.

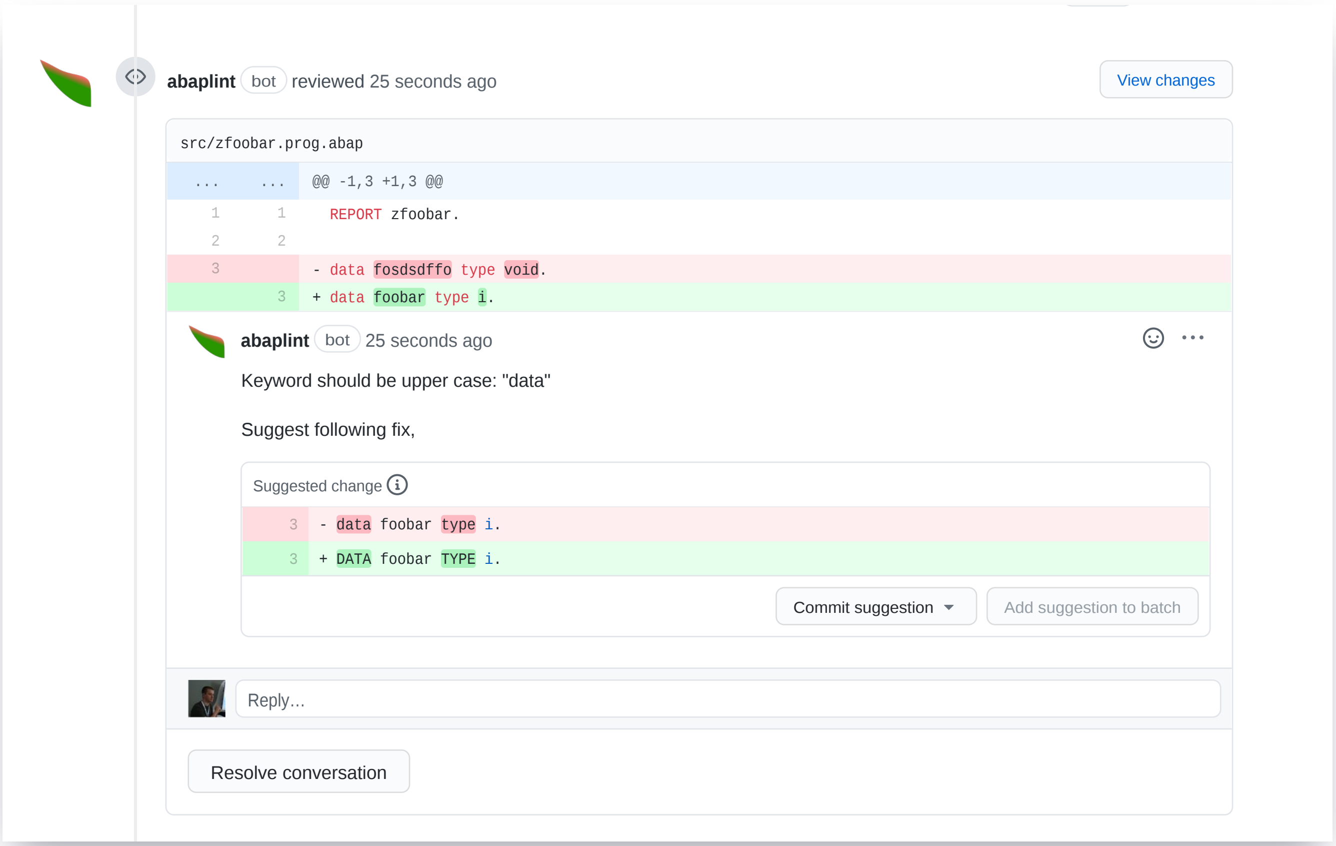
5.4. Parallel File Changes

If identical file names are changed in more than one open pull request is reported it will trigger an observation.

Git will do its best to merge the changes, however it can be troublesome to deploy these changes to the landscape simultaneously.

6. Suggestions

Suggestions by abaplint.app appear in pull requests, rules which have quick-fixes must be enabled, and only one suggestion appear at a time.



abaplint bot reviewed 25 seconds ago

src/zfoobar.prog.abap

```
@@ -1,3 +1,3 @@
  REPORT zfoobar.

- data fosdsdffo type void.
+ data foobar type i.
```

abaplint bot 25 seconds ago

Keyword should be upper case: "data"

Suggest following fix,

Suggested change

```
- data foobar type i.
+ DATA foobar TYPE i.
```

7. Target Rules

Target rules are configured in the normal `abaplint.json` file, these rules does not set failed status, but instead allows pull requests to be merged. The target rules findings shows up as observations.

This is useful for moving the code in a better direction when its convenient. The developer and reviewer will both see the observations in the pull request status, and decide if to fix or defer. Deferring is useful for critical/complex changes, but many scenarios also allow for gradually performing fixes to move to a better target.

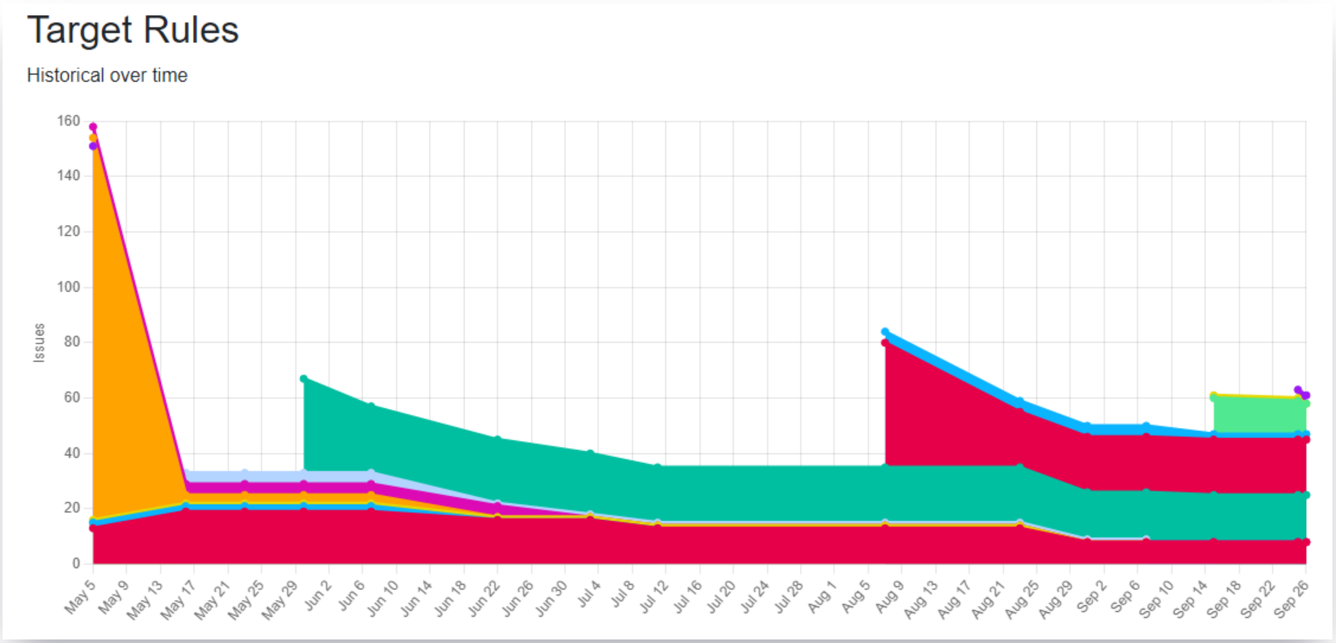
7.1. Configuration

The target rules can be defined after the normal `rules`, using the vscode abaplint extension will help with validation and intellisense.

```
{
  ...
  "rules": {
    ...
  },
  "targetRules": {
    "double_space": true,
  },
}
```

7.2. Insights

The team can always check up on the direction the code is moving. Perhaps moving some rules to required, or adding additional, or even removing if the code is not moving as first intended.



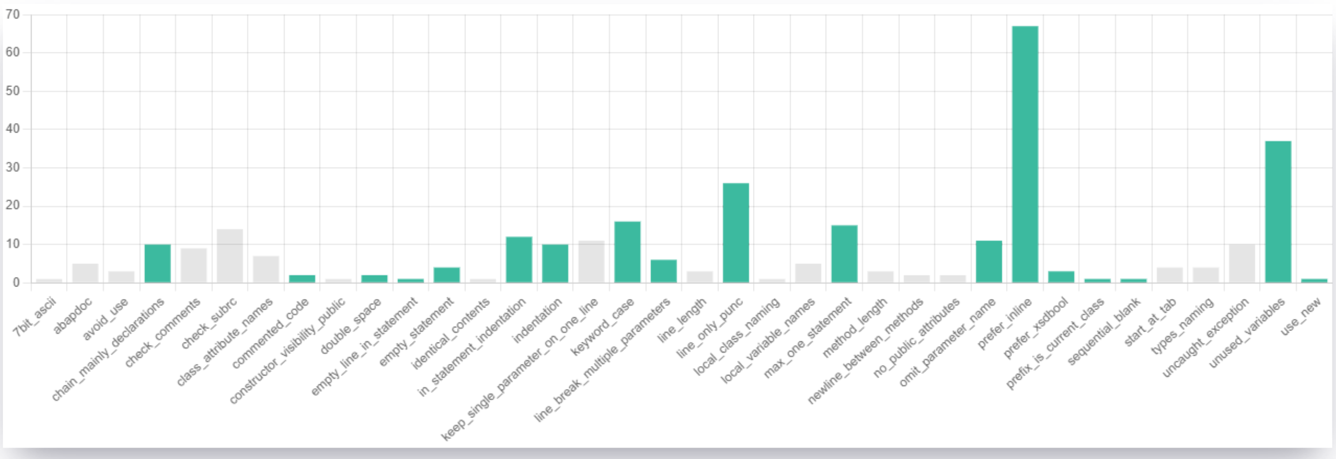
Spikes will happen when enabling additional target rules.

8. Insights

Various code insights are automatically generated daily by abaplint.app, insights can be accessed via web. Insights for public projects are public, users must login to access insights for private projects. All insights are generated only for the default branch.

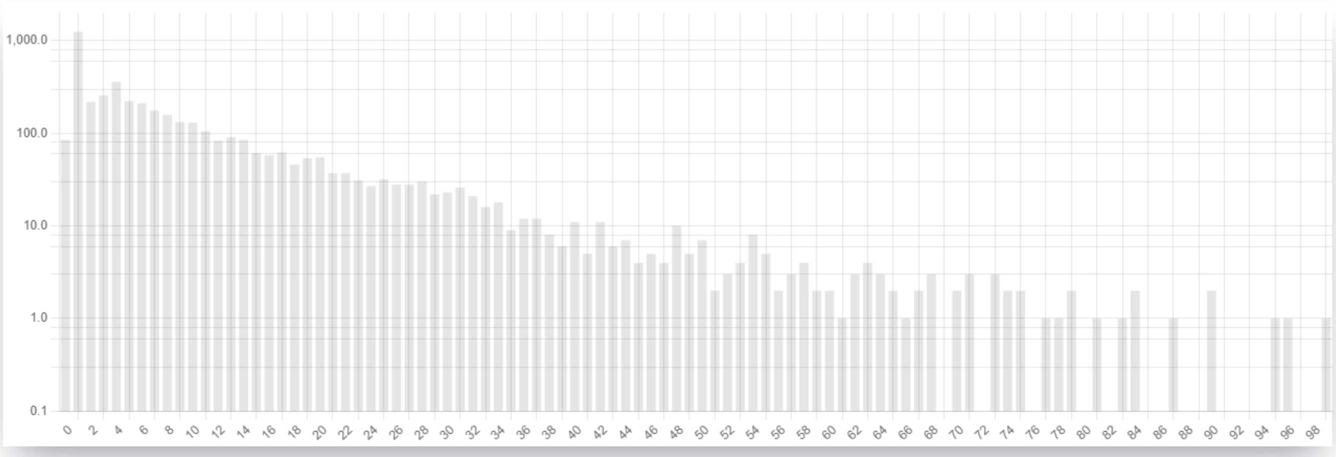
8.1. Issues

abaplint is run for the source code, if no configuration is present, the default configuration is used. Issues are listed, plus a graph gives an overview, issues with quick-fixes are shown in green,



8.2. Method Length

Method length gives an overview of the method implementation lengths across the source code. Note that the scale is logarithmic,

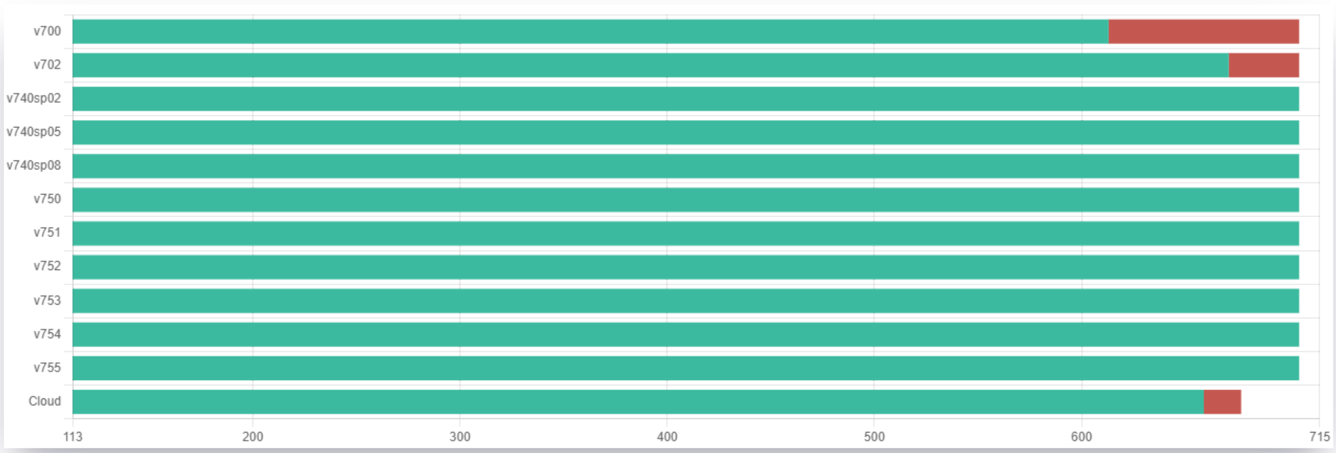


8.3. Method Complexity

Overview of method complexity measured as cyclomatic complexity.

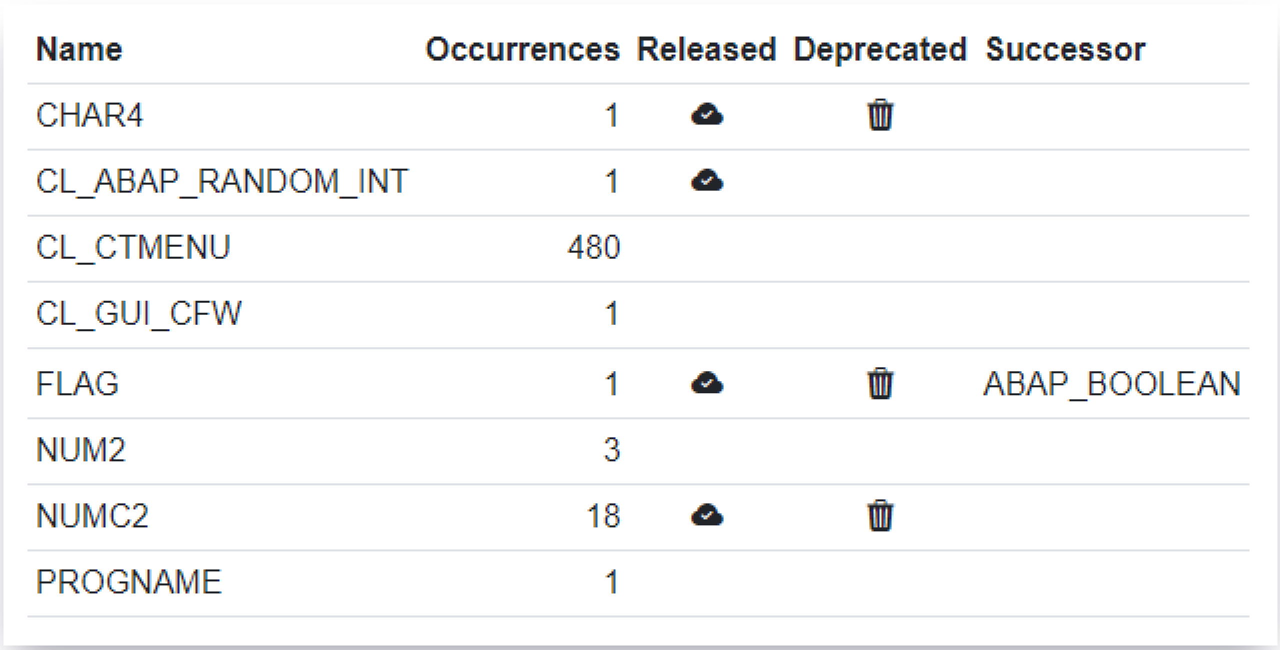
8.4. Statement Compatibility

Statement parsing id performed against different language versions, this gives an overview of effort required to downport or upport the codebase.



8.5. Void Types

Lists void types as per the `errorNamespace` configuration, along with release, deprecation and successor information.



Name	Occurrences	Released	Deprecated	Successor
CHAR4	1	☁	🗑	
CL_ABAP_RANDOM_INT	1	☁		
CL_CTMENU	480			
CL_GUI_CFW	1			
FLAG	1	☁	🗑	ABAP_BOOLEAN
NUM2	3			
NUMC2	18	☁	🗑	
PROGNAME	1			

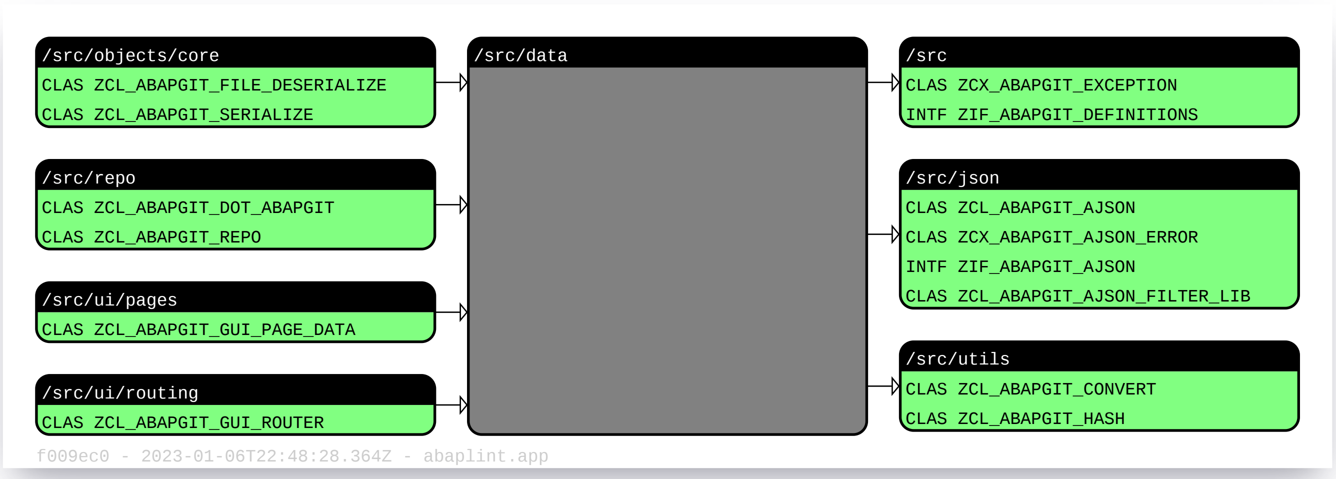
8.6. Dependencies

Lists where dependencies are used in the codebase. Dependencies are configured in the `abaplint.json` file

8.7. Package Coupling

Given a folder(package) the diagram lists objects and packages using something in the package on the left side. The right side lists what is being used from the package.

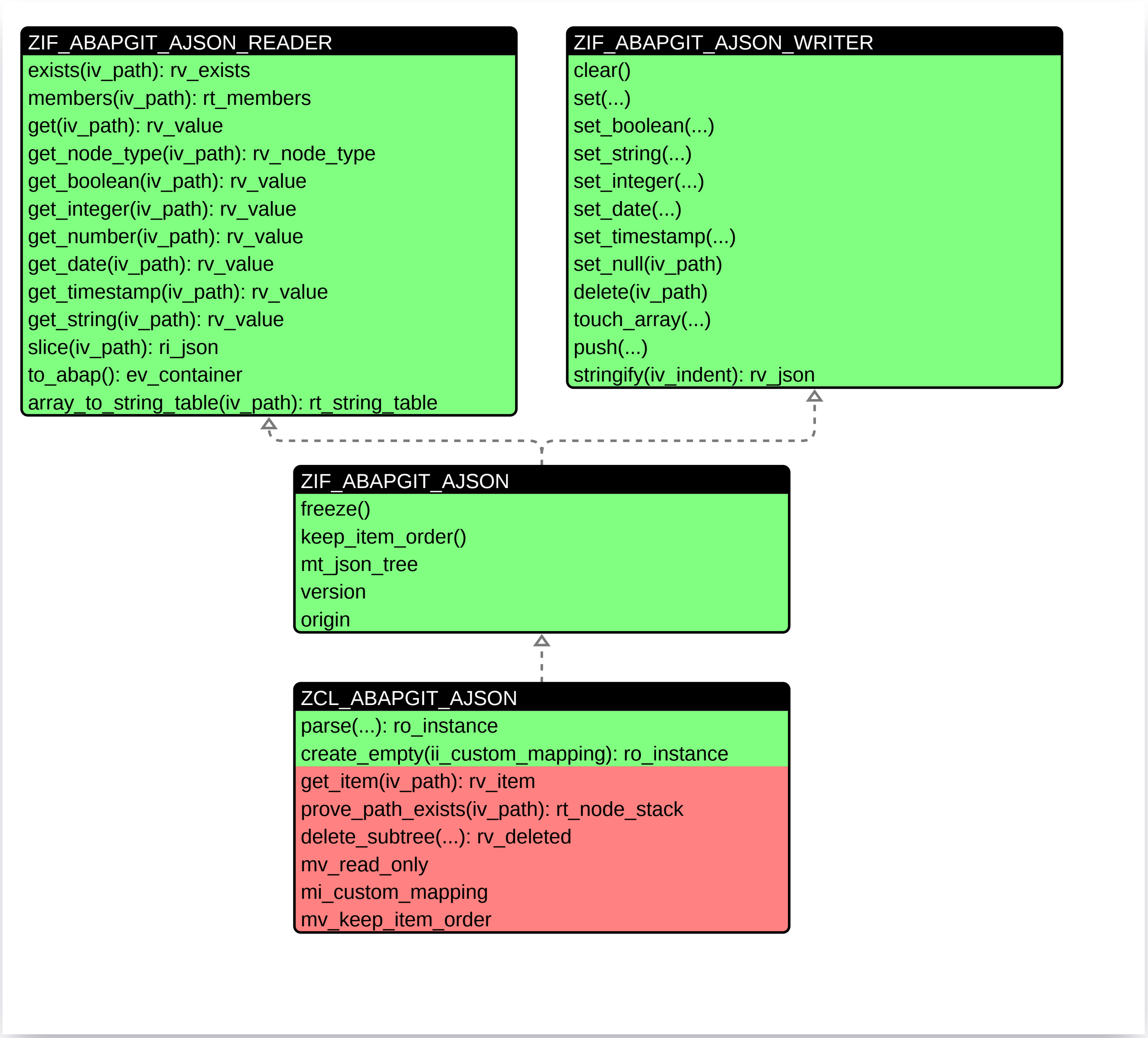
Test class includes and dependencies are ignored. Dependencies into sub-folders are shown with blue background.



/src/objects/core
CLAS ZCL_ABAPGIT_FILE_DESERIALIZE
CLAS ZCL_ABAPGIT_SERIALIZE
/src/repo
CLAS ZCL_ABAPGIT_DOT_ABAPGIT
CLAS ZCL_ABAPGIT_REPO
/src/ui/pages
CLAS ZCL_ABAPGIT_GUI_PAGE_DATA
/src/ui/routing
CLAS ZCL_ABAPGIT_GUI_ROUTER
/src/data
/src
CLAS ZCX_ABAPGIT_EXCEPTION
INTF ZIF_ABAPGIT_DEFINITIONS
/src/json
CLAS ZCL_ABAPGIT_AJSON
CLAS ZCX_ABAPGIT_AJSON_ERROR
INTF ZIF_ABAPGIT_AJSON
CLAS ZCL_ABAPGIT_AJSON_FILTER_LIB
/src/utils
CLAS ZCL_ABAPGIT_CONVERT
CLAS ZCL_ABAPGIT_HASH
f009ec0 - 2023-01-06T22:48:28.364Z - abaplint.app

8.8. UML Class Diagrams

Provides auto generated UML diagrams for all global classes and interfaces, example:



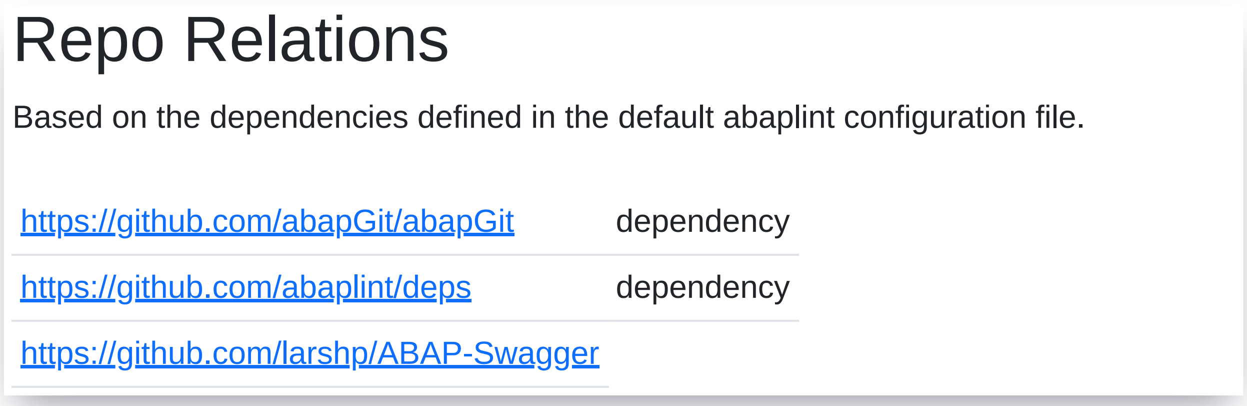
ZIF_ABAPGIT_AJSON_READER
exists(iv_path): rv_exists
members(iv_path): rt_members
get(iv_path): rv_value
get_node_type(iv_path): rv_node_type
get_boolean(iv_path): rv_value
get_integer(iv_path): rv_value
get_number(iv_path): rv_value
get_date(iv_path): rv_value
get_timestamp(iv_path): rv_value
get_string(iv_path): rv_value
slice(iv_path): ri_json
to_abap(): ev_container
array_to_string_table(iv_path): rt_string_table
ZIF_ABAPGIT_AJSON_WRITER
clear()
set(...)
set_boolean(...)
set_string(...)
set_integer(...)
set_date(...)
set_timestamp(...)
set_null(iv_path)
delete(iv_path)
touch_array(...)
push(...)
stringify(iv_indent): rv_json
ZIF_ABAPGIT_AJSON
freeze()
keep_item_order()
mt_json_tree
version
origin
ZCL_ABAPGIT_AJSON
parse(...): ro_instance
create_empty(ii_custom_mapping): ro_instance
get_item(iv_path): rv_item
prove_path_exists(iv_path): rt_node_stack
delete_subtree(...): rv_deleted
mv_read_only
mi_custom_mapping
mv_keep_item_order

8.9. Disabled Rules

If rules are disabled in the configuration, abaplint.app will try to enable to rule and provide the issue count and how much is quick-fixable.

8.10. Repository Relations

Dependencies from the abaplint configuration file is listed,



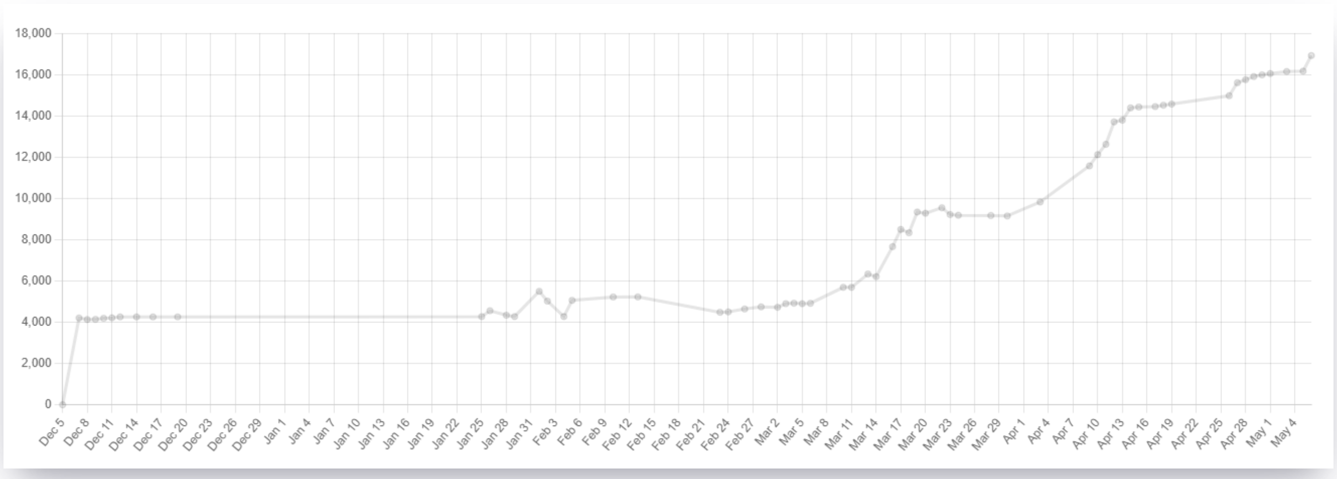
Repo Relations

Based on the dependencies defined in the default abaplint configuration file.

https://github.com/abapGit/abapGit	dependency
https://github.com/abaplint/deps	dependency
https://github.com/larshp/ABAP-Swagger	

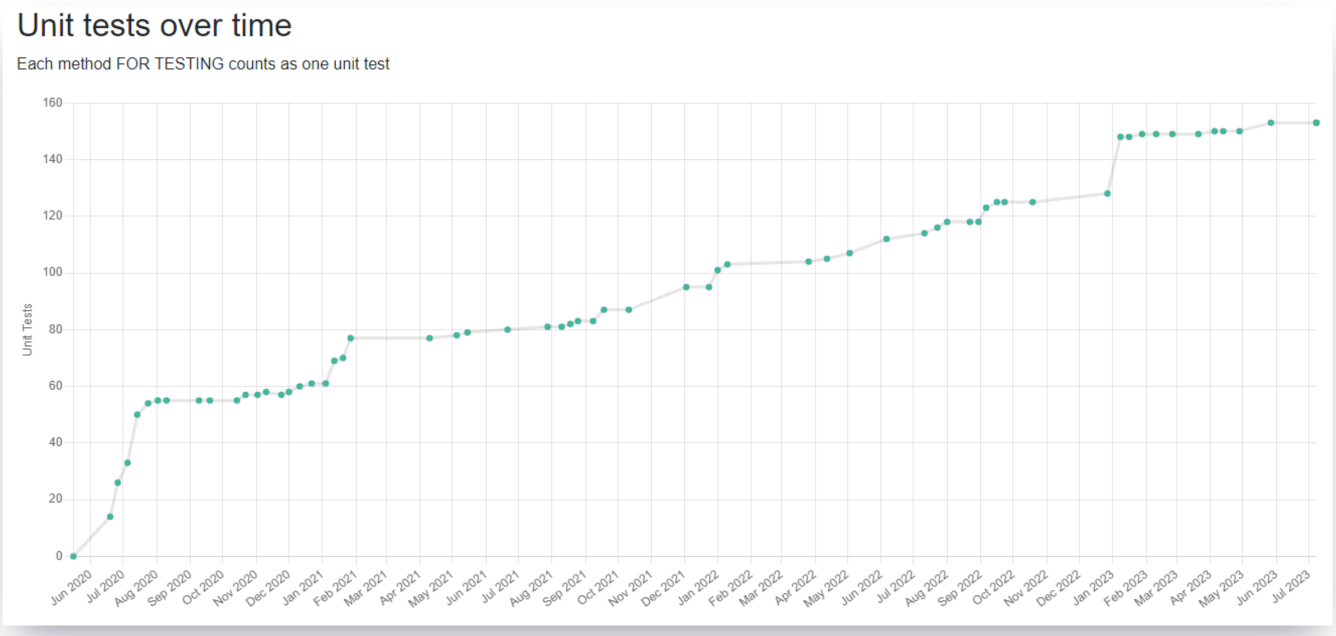
8.11. Lines Over Time

Show codebase size over time,



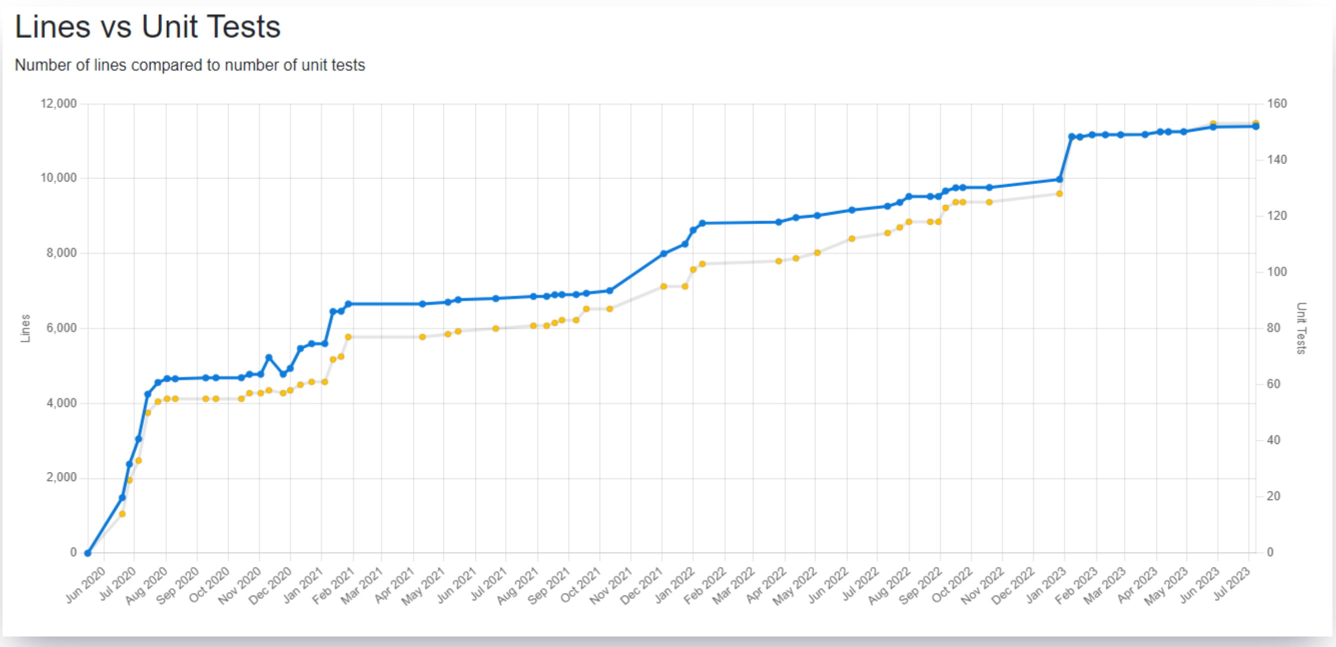
8.12. Unit Tests

Show number of unit tests over time,



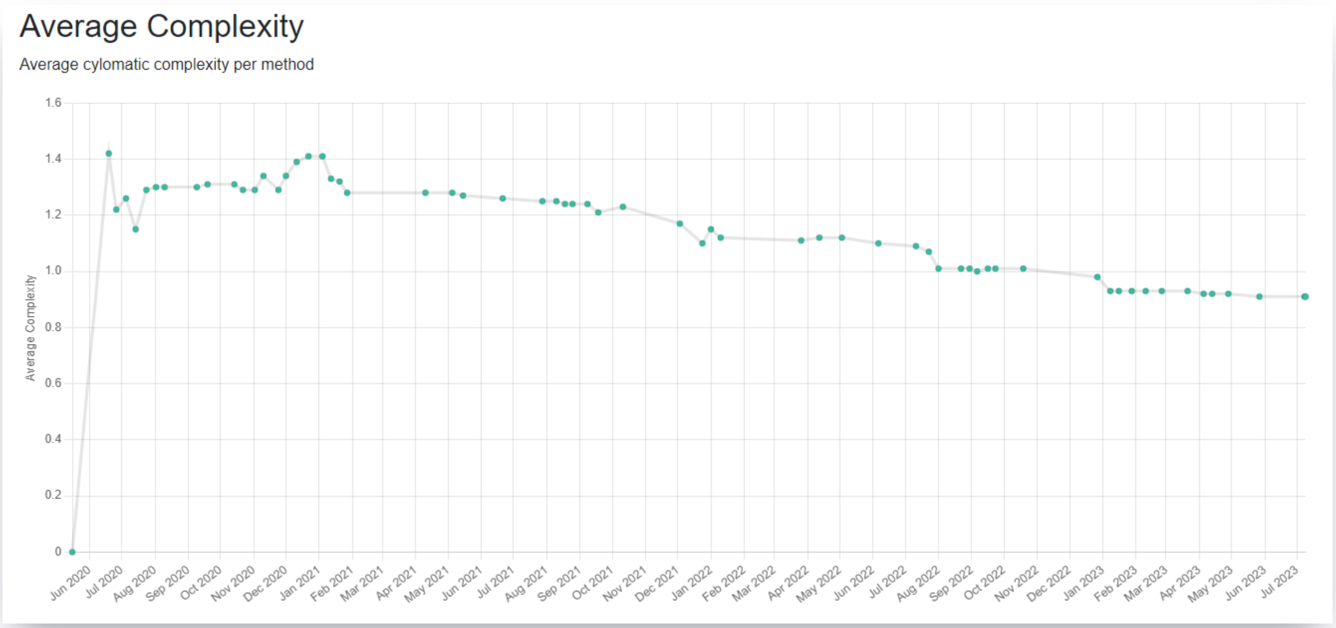
8.13. Lines vs Unit Tests

Show number of lines compared to number of unit tests over time,



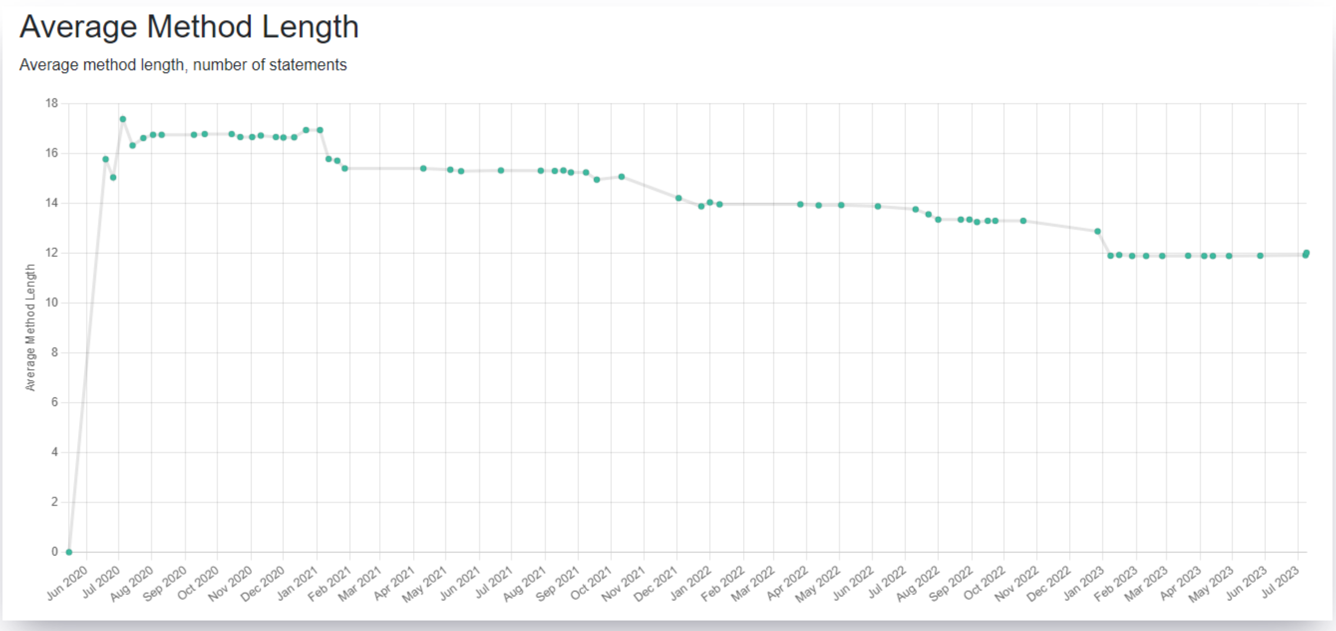
8.14. Average Complexity

Show average cyclomatic complexity over time



8.15. Average Method Length

Show average method length over time, measured in number of statements



8.16. Target Rules

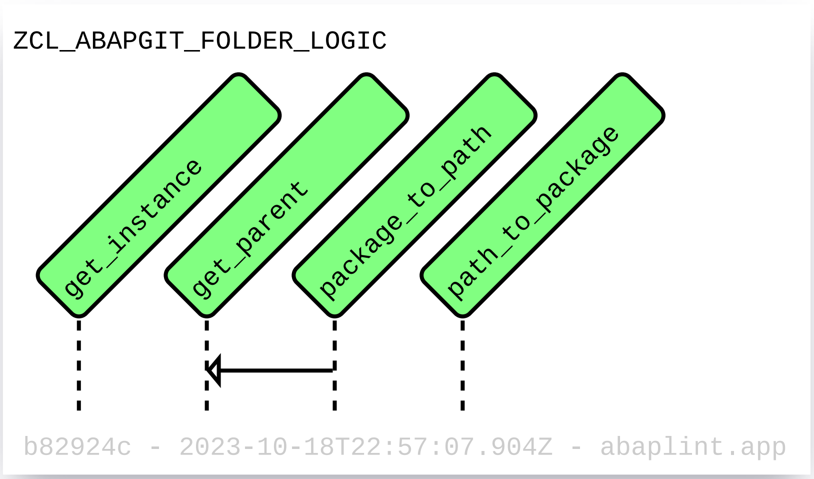
See the Target Rules chapter

8.17. Class Length

Find the latest classes in the codebase, these are good candidates for refactoring.

8.18. Intra Class Call Graph

Display relations between methods internally in a class. The diagram does not reflect sequence, only if a method is called from another method. Only static method references are displayed.



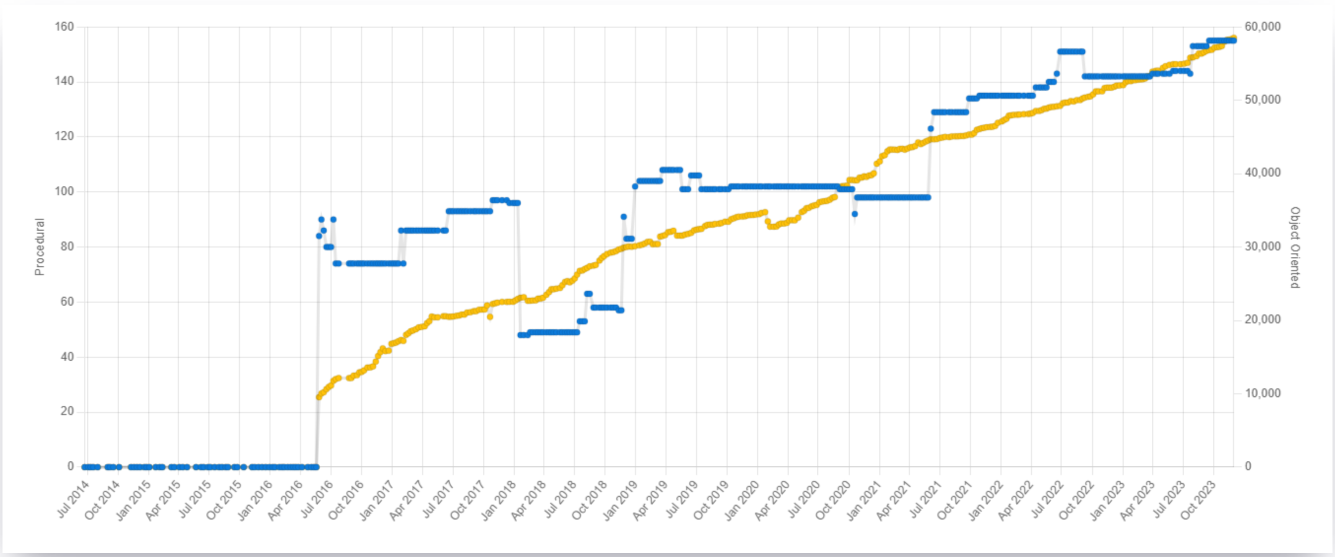
8.19. Object Types

Overview of how many objects of each type exists in the repository. Cloud indicates the object type is possible in ABAP Cloud, not that the actual object contents works in ABAP Cloud.

Type	Count	Cloud
CLAS	406	☁
INTF	87	☁
W3MI	9	
PROG	3	
FUGR	1	☁
TRAN	1	

8.20. Procedural vs OO

Number of statements in FORM + FUNCTION compared to number of statements in METHOD implementations



9. Cross Check

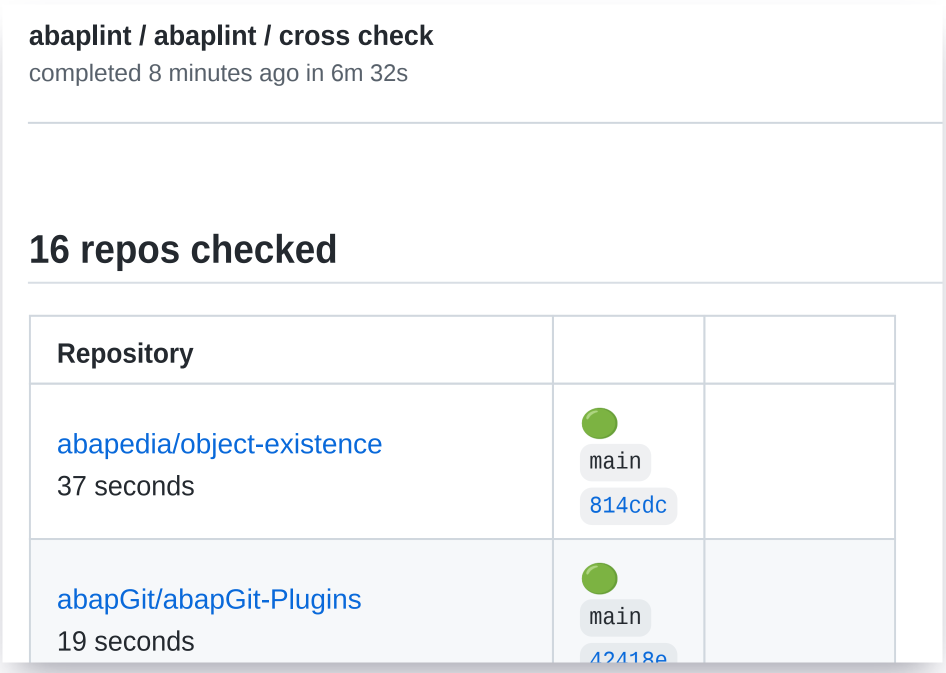
Some applications consists of multiple components maintained by different teams. This abaplint.app feature allows to discover syntax errors across repositories before changes are merged, so that fixes can be prepared in due time.

All repository relations and branches are found and checked one by one. If the checked already have errors, it is skipped.

Public repositories consider only public dependencies, private will take repositories in the same organization into account.

The cross check feature is enabled by default, but only reported if there are any related repositories, the feature can be disabled in `abaplint-app.json` if needed.

If there are no errors found, success is reported as a status check on the pull request. In case of new errors, a neutral status is reported, click the link from the status check to see the overview of the results,



abaplint / abaplint / cross check
completed 8 minutes ago in 6m 32s

16 repos checked

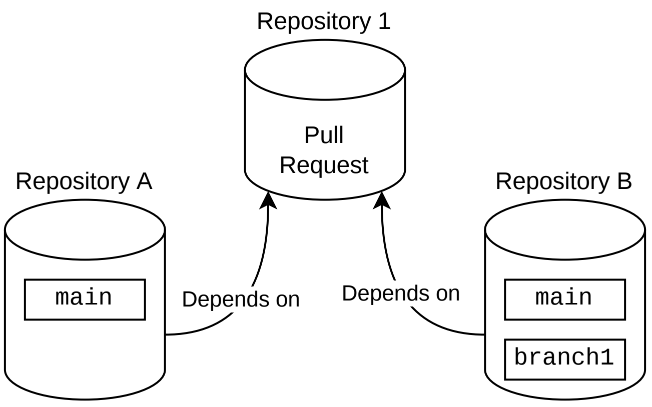
Repository		

abapedia/object-existence
37 seconds
	🟢
	`main`
	`814cdc`	

abapGit/abapGit-Plugins
19 seconds
	🟢
	`main`
	`42418e`	

9.1. Example

A pull request is opened in "Repository 1", which have two dependent repositories: A and B,



The pull request will cross check all combinations,

- Repository A, branch `main` against the pull request
- Repository B, branch `main` against the pull request
- Repository B, branch `branch1` against the pull request

10. Additional Information

10.1. Licensing

Use in private/internal repositories requires a license purchased via GitHub Marketplace.

Only active users pushing code requires licenses. Users are not named users ie. active users can change between each month with changes.

The author email address from each commit is used to count users. Use `noreply` email addresses to protect the identity of users, and align the email addresses used.

10.2. Privacy

See Privacy Policy

- To perform code analysis, the source is temporarily cloned using a temporary token
- Analysis is performed on Digital Ocean infrastructure in Germany
- Aggregated data about the source-code is stored