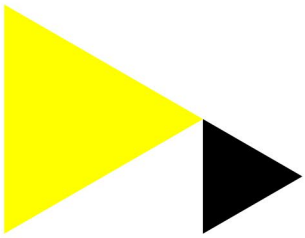
Standard for Public Code

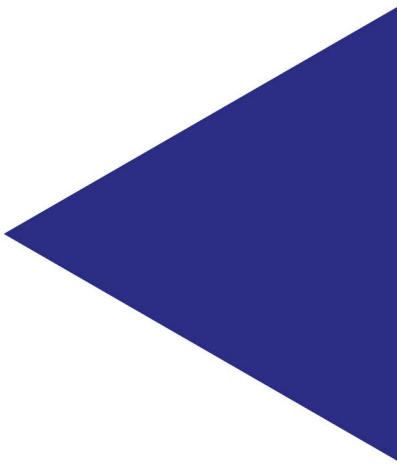
version 0.7.0

Author(s)
Cerveny, Ben; Regt, Elena Findley-de; Mullie, Claus; van Hoytema, Boris; de Waal, Martijn; Erkelens, Tamas; van der Net, Mark; Spaan, Bert; Slinger, Timo

Publication date
2023

Document Version
Final published version



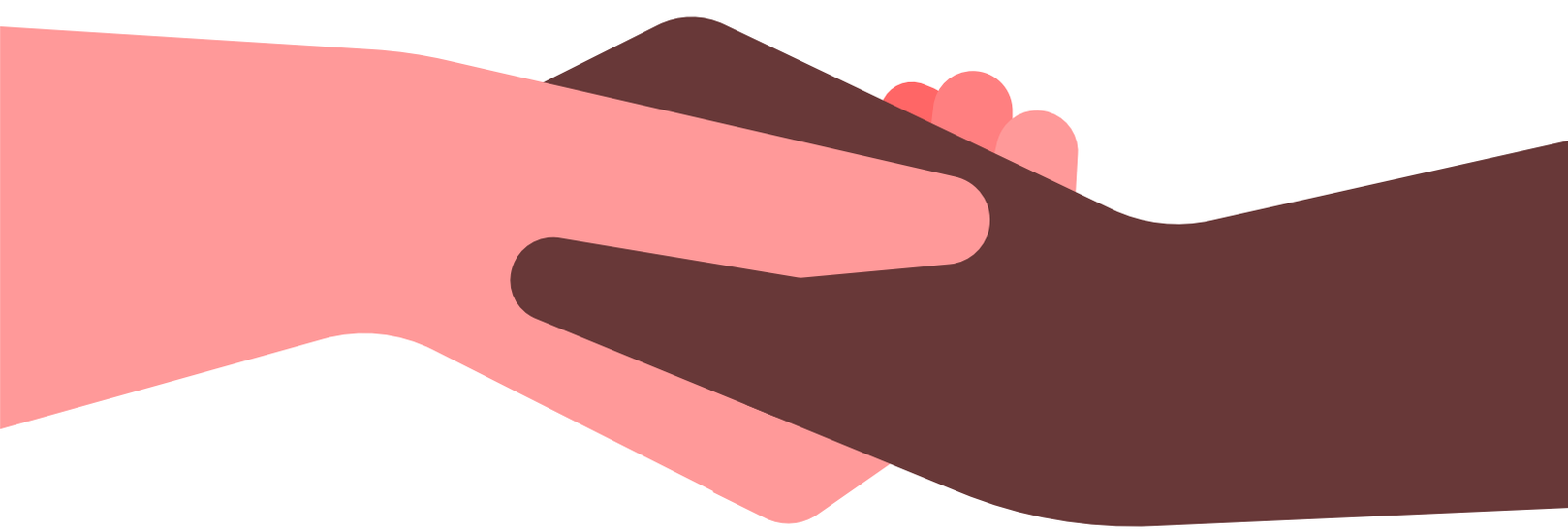


Request for contributions

Standard for Public Code

What public code is and how to implement it for:

- **P** Public policy makers
- **M** Managers
- **D** Developers and designers



Draft
Version 0.7.0

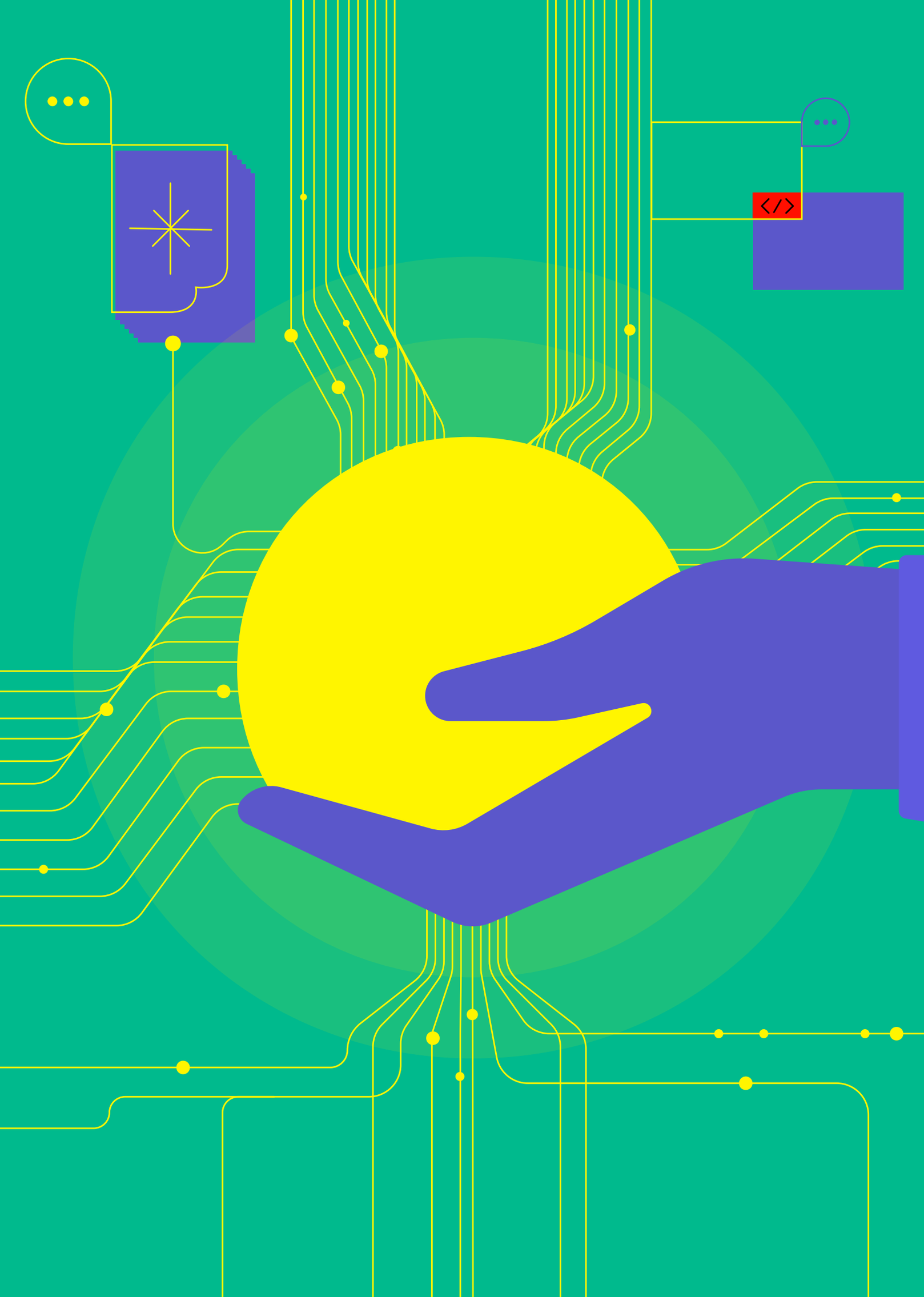
DPG

github.com/publiccodenet/standard

Authors

- Alba Roza, The Foundation for Public Code https://publiccode.net/
- Arnout Engelen
- Arnout Schuijff, The Foundation for Public Code
- Audrey Tang, digitalminister.tw https://digitalminister.tw/
- Ben Cerveny, The Foundation for Public Code
- Bert Spaan
- Boris van Hoytema, The Foundation for Public Code
- Charlotte Heikendorf
- Claus Mullie, The Foundation for Public Code
- David Barberi
- Edo Plantinga, Code For NL https://codefor.nl/
- Elena Findley-de Regt, The Foundation for Public Code
- Eric Herman, The Foundation for Public Code
- Felix Faassen, The Foundation for Public Code
- Floris Deerenberg
- Jan Ainali, The Foundation for Public Code
- Johan Groenen, Code For NL
- Marcus Klaas de Vries
- Mark van der Net, Gemeente Amsterdam https://www.amsterdam.nl/en/
- Martijn de Waal, Amsterdam University of Applied Sciences (AUAS), Faculty of Digital Media and Creative Industries, Lectorate of Play & Civic Media https://www.amsterdamuas.com/
- Matti Schneider
- Mauko Quiroga
- Maurice Hendriks, Gemeente Amsterdam
- Maurits van der Schee, Gemeente Amsterdam
- Mirjam van Tiel, The Foundation for Public Code
- Ngô Ngọc Đức Huy
- Paul Keller
- Petteri Kivimäki, Nordic Institute for Interoperability Solutions (NIIS) https://niis.org
- Sky Bristol
- Tamas Erkelens, Gemeente Amsterdam
- Timo Slinger

Table of Contents



Readers guide

The Standard describes a number of criteria. All criteria have consistent sections that make it clear how to create great public code.

References to “policy makers”, “managers”, and “developers and designers” apply to anyone performing duties associated with these roles, regardless of their job title. It is common for individuals to have duties which span multiple roles.

Below is a brief explanation of each of these sections and how they are used within the criteria of the Standard.

Introduction

This section explains what the criterion aims to achieve and why it is important for a codebase’s users and contributors.

Requirements

This section lists what needs to be done in order to comply with the standard.

The following keywords in this document are to be interpreted as described in IETF RFC 2119:

https://tools.ietf.org/html/rfc2119

- MUST
- MUST NOT
- REQUIRED
- SHALL
- SHALL NOT
- SHOULD
- SHOULD NOT
- RECOMMENDED
- MAY
- OPTIONAL

How to test

This section offers actions you can take to see if a contribution is compliant with the Standard. This is key if you want to operationalize the Standard.

We've tried to word it so that someone who is not intimately acquainted with the subject matter can still do a basic check for compliance.

P Public policy makers: what you need to do

This section tries to specifically speak to policy makers by offering them concrete actions they can perform in their role.

Public policy makers set the priorities and goals of projects and may be less technologically experienced.

M Managers: what you need to do

This section tries to specifically speak to managers by offering concrete actions they can perform in their role.

Managers are responsible for on-time project delivery, stakeholder management and continued delivery of the service. For this they are wholly reliant on both policy makers as well as developers and designers. They need to create the right culture, line up the right resources and provide the right structures to deliver great services.

D Developers and designers: what you need to do

This section tries to specifically speak to developers and designers by offering them concrete actions they can perform in their role.

Developers are usually more technically aligned and have more impact on the delivery of services than the previous groups.

Limitation of scope

The Standard for Public Code is not meant to cover individual implementations of a codebase. This means the standard does not tell implementers how to comply with their organization's local technical infrastructure or legal framework.

Also, while the Standard for Public Code refers to several standards and has considerable overlap with others, its purpose is to enable collaboration. Therefore, it does not aim to replace quality standards, like the ISO 25000 series, or those focusing on security, like the OpenSSF Best Practices Badge, but to synergize well with them.

https://github.com/coreinfrastructure/best-practices-badge

And while the purpose includes enabling collaboration, it will not guarantee that a community will spring into existence. That still requires proactiveness and ambition beyond making the codebase collaboration ready.

Glossary

Code

Any explicitly described system of rules. This includes laws, policy and ordinances, as well as source code that is used to build software. Both of these are rules, some executed by humans and others by machines.

Codebase

Any discrete package of code (both source and policy), the tests and the documentation required to implement a piece of policy or software.

This can be, for example, a document or a version-control repository.

Continuous integration

In software engineering, continuous integration (CI) is the practice of merging all developers' working copies to a development branch of a codebase as frequently as reasonable.

Different contexts

Two contexts are different if they are different public organizations or different departments for which there is not one decision maker that could make collaboration happen naturally.

General public

The public at large: end users of the code and the services based upon it.

For example, a city's residents are considered end users of a city's services and of all code that powers these services.

Open source

Open source is defined by the Open Source Initiative in their Open Source Definition.

https://opensource.org/osd-annotated

Open standard

An open standard is any standard that meets the Open Source Initiative's Open Standard Requirements.

https://opensource.org/osr

Policy

A policy is a deliberate system of principles to guide decisions and achieve rational outcomes. A policy is a statement of intent, and is implemented as a procedure or protocol. Policies are generally adopted by a governance body within an organization. Policies can assist in both subjective and objective decision making.

Public policy is the process by which governments translate their political vision into programs and actions to deliver outcomes.

At the national level, policy and legislation (the law) are usually separate. The distinction is often more blurred in local government.

In the Standard the word 'policy' refers to policy created and adopted by public organizations such as governments and municipalities.

Public code

Public code is open source software developed by public organizations, together with the policy and guidance needed for collaboration and reuse.

Public code is both computer source code (such as software and algorithms) and public policy executed in a public context, by humans or machines.

Public code serves the public interest, is open, legible, accountable, accessible and sustainable.

By developing public code independently from but still implementable in the local context for which it was developed, as well as documenting the development process openly, public code can provide a building block for others to:

- re-implement in their local context
- take as a starting point to continue development
- use as a basis for learning

To facilitate reuse, public code is either released into the public domain or licensed with an open license that permits others to view and reuse the work freely and to produce derivative works.

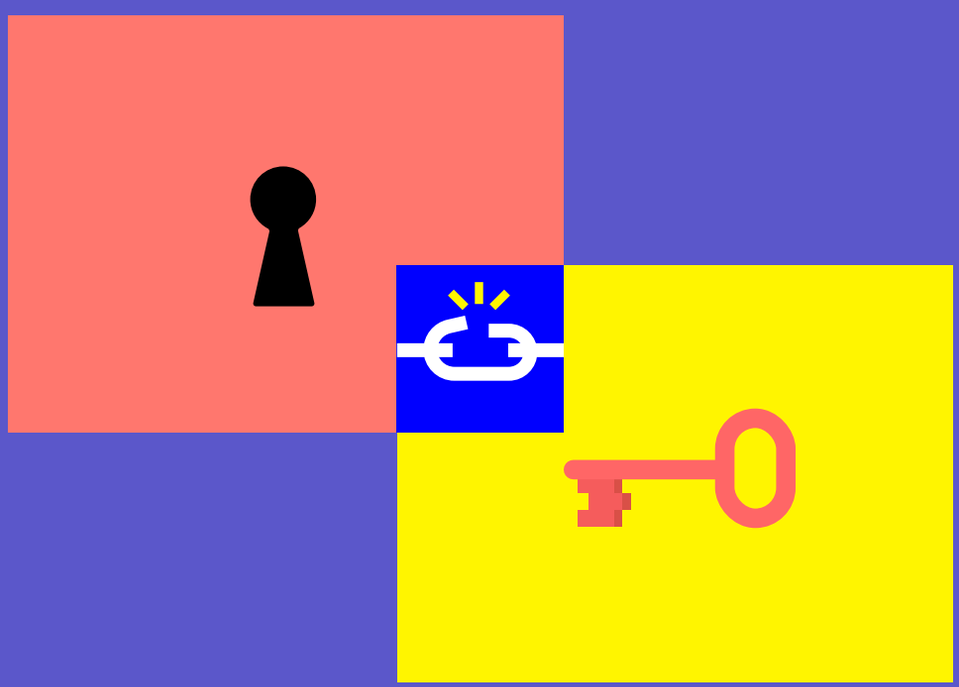
Repository

A repository is a storage location used by version control tools for files and metadata of a codebase. Repositories allow multiple contributors to work on the same set of files. Repositories are able to store multiple versions of sets of files.

Version control

Version control is the management of changes to source code and the files associated with it. Changes are usually identified by a code, termed the *revision number* (or similar). Each revision is associated with the time it was made and the person making the change, thus making it easier to retrace the evolution of the code. Revision control systems can be used to compare different versions with each other and to see how content has changed over time.

Criteria



Code in the open

Coding in the open improves transparency, increases code quality, makes the code easier to audit, and enables collaboration.

Together, this creates more opportunities for citizens to understand how software and policy impact their interactions with a public organization.

Requirements

- All source code for any policy in use (unless used for fraud detection) MUST be published and publicly accessible.
- All source code for any software in use (unless used for fraud detection) MUST be published and publicly accessible.
- The codebase MUST NOT contain sensitive information regarding users, their organization or third parties.
- Any source code not currently in use (such as new versions, proposals or older versions) SHOULD be published.
- Documenting which source code or policy underpins any specific interaction the general public may have with an organization is OPTIONAL.

How to test

- Confirm that the source for each version currently in use is published on the internet where it can be seen from outside the original contributing organization and without the need for any form of authentication or authorization.
- Confirm that the codebase files and commit history do not include sensitive information.
- Check for the publication of source code not currently in use.

P Public policy makers: what you need to do

- Develop policies in the open.
- Prioritize open and transparent policies.

M Managers: what you need to do

- Develop a culture that embraces openness, learning and feedback.
- Collaborate with external vendors and freelancers by working in the open.

D Developers and designers: what you need to do

- As a reviewer, for each commit, verify that content does not include sensitive information such as configurations, usernames or passwords, public keys or other real credentials used in production systems.
- Clearly split data and code, in order to meet the requirement about sensitive information above.

Further reading

- Coding in the open by the UK Government Digital Service. https://gds.blog.gov.uk/2012/10/12/coding-in-the-open/
- When code should be open or closed by the UK Government Digital Service. https://www.gov.uk/government/publications/open-source-guidance/when-code-should-be-open-or-closed
- Security considerations when coding in the open by the UK Government Digital Service. https://www.gov.uk/government/publications/open-source-guidance/security-considerations-when-coding-in-the-open
- Deploying software regularly by the UK Government Digital Service. https://www.gov.uk/service-manual/technology/deploying-software-regularly
- How GDS uses GitHub by the UK Government Digital Service. https://gdstechnology.blog.gov.uk/2014/01/27/how-we-use-github/

Bundle policy and source code

Access to both source code and policy documentation provides building blocks for anyone to implement the codebase in their local context or contribute to the further development of the codebase.

Understanding the domain and policies within that domain is fundamental to understanding what problems a codebase is trying to solve and how it sets out to solve them.

To be able to evaluate whether to implement a codebase in a new context, an organization needs to understand what process changes it must choose to make or how to contribute additional configurability to the existing solution in order to adapt it to the new context.

Requirements

- The codebase MUST include the policy that the source code is based on.
- If a policy is based on source code, that source code MUST be included in the codebase, unless used for fraud detection.
- Policy SHOULD be provided in machine readable and unambiguous formats.
- Continuous integration tests SHOULD validate that the source code and the policy are executed coherently.

How to test

- Confirm with a civil servant that all policy that the source code is based on is included.
- Confirm with a civil servant that all source code that the policy is based on is included.
- Check if policy can be interpreted by a machine.

- Check the continuous integration tests for coherent execution of source code and policy pass.

P Public policy makers: what you need to do

- Collaborate with developers and designers to make sure there is no mismatch between policy code and source code.
- Provide the relevant policy texts for inclusion in the repository; if the text is not available in English, also provide an English summary. Be sure to include standards that your organization has chosen to adhere to and any organizational processes which impact the development or the deployment context of the codebase for your organization.
- Provide references and links to texts which support the policies.
- Document policy in formats that are unambiguous and machine-readable such as Business Process Model and Notation, Decision Model and Notation and Case Management Model Notation.

https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation
https://en.wikipedia.org/wiki/CMMN
https://en.wikipedia.org/wiki/Decision_Model_and_Notation

- Track policy with the same version control and documentation used to track source code.
- Check in regularly to understand how the non-policy code in the codebase has changed and whether it still matches the intentions of the policy.
- Include relevant policies which impact the community, codebase, and development, including legal obligations like the General Data Protection Regulation or the EU Web Accessibility Directive, or human rights policies, like a public organization's commitment to equal opportunity.

https://eur-lex.europa.eu/eli/reg/2016/679/oj

https://ec.europa.eu/digital-single-market/en/web-accessibility

M Managers: what you need to do

- Keep policy makers, developers and designers involved and connected throughout the whole development process.
- Make sure policy makers, developers and designers are working to the same objectives.

D Developers and designers: what you need to do

- Become familiar with and be able to use the process modelling notation that the policy makers in your organization use.
- Work together with policy makers to make sure there is no mismatch between policy code and source code.
- Give feedback on how to make policy documentation more clear.

Further reading

- BPMN Quick Guide by Trisotech. https://www.bpmnquickguide.com/view-bpmn-quick-guide/

Create reusable and portable code

Creating reusable and portable code enables policy makers, developers and designers to reuse what has been developed, test it, improve it and contribute those improvements back, leading to better quality, cheaper maintenance and higher reliability.

Thoughtfully and purposefully designing a codebase for reusability allows for the mission, vision and scope of the codebase to be shared by multiple parties. Codebases developed and used by multiple parties are more likely to benefit from a self-sustaining community.

Organizing a codebase such that it is composed of well documented modules improves reusability and maintainability. A module is easier to reuse in another context if its purpose is clearly documented.

Code which does not rely on the situation-specific infrastructure of any contributor, vendor or deployment can be tested by any other contributor.

Requirements

- The codebase MUST be developed to be reusable in different contexts.
- The codebase MUST be independent from any secret, undisclosed, proprietary or non-open licensed code or services for execution and understanding.
- The codebase SHOULD be in use by multiple parties.
- The roadmap SHOULD be influenced by the needs of multiple parties.
- The development of the codebase SHOULD be a collaboration between multiple parties.
- Configuration SHOULD be used to make code adapt to context specific needs.
- The codebase SHOULD be localizable.

- Code and its documentation SHOULD NOT contain situation-specific information.
- Codebase modules SHOULD be documented in such a way as to enable reuse in codebases in other contexts.
- The code SHOULD NOT require services or platforms available from only a single vendor.

How to test

- Confirm with someone in a similar role at another organization if they can use the codebase and what that would entail.
- Confirm that the codebase can run without using any proprietary or non open-licensed code or services.
- Check that the codebase is in use by multiple parties or in multiple contexts.
- Check that the codebase contributors are from multiple parties.
- Check that the codebase files and commit history do not include situation-specific data.
- Check that the code can be deployed and run without services or platforms available from a single vendor.

P Public policy makers: what you need to do

- Document your policy with enough clarity and detail that it can be understood outside of its original context.
- Make sure your organization is listed as a known user by the codebase.
- Identify other organizations for your teams to collaborate with.

M Managers: what you need to do

- Make sure that stakeholders and business owners understand that reusability is an explicit codebase goal as this reduces technical debt and provides sustainability for the codebase.
- Make sure that your teams are collaborating with other teams.

D Developers and designers: what you need to do

Source should be designed:

- for reuse by other users and organizations regardless of locale,
- to solve a general problem instead of a specific one,
- in logically meaningful and isolated modules,
- so that someone in a similar organization facing a similar problem would be able to use (parts of) the solution.

Make sure that the codebase documentation describes the build-time and runtime dependencies. If your context requires deploying to proprietary platforms or using proprietary components, make sure that collaborators can develop, use, test, and deploy without them.

For each commit, reviewers verify that content does not include situation-specific data such as hostnames, personal and organizational data, or tokens and passwords.

Further reading

- Making source code open and reusable by the UK Government Digital Service.
  https://www.gov.uk/service-manual/technology/making-source-code-open-and-reusable
- Localization vs. Internationalization by the World Wide Web Consortium.
  https://www.w3.org/International/questions/qa-i18n

Welcome contributors

The atmosphere in a codebase community helps users decide to use one codebase over another. Welcoming anyone as a contributor enables the community to grow and sustain itself over time. A community where contributors have clear ways to influence codebase and community goals and progress is less likely to split and end up in diverging communities. Newcomers need to understand and trust the codebase community's governance.

Requirements

- The codebase MUST allow anyone to submit suggestions for changes to the codebase.
- The codebase MUST include contribution guidelines explaining what kinds of contributions are welcome and how contributors can get involved, for example in a `CONTRIBUTING` file.
- The codebase MUST document the governance of the codebase, contributions and its community, for example in a `GOVERNANCE` file.
- The contribution guidelines SHOULD document who is expected to cover the costs of reviewing contributions.
- The codebase SHOULD advertise the committed engagement of involved organizations in the development and maintenance.
- The codebase SHOULD have a publicly available roadmap.
- The codebase SHOULD publish codebase activity statistics.
- Including a code of conduct for contributors in the codebase is OPTIONAL.

How to test

- Confirm that it is possible to submit suggestions for changes to the codebase.

- Confirm there are contribution guidelines.
- Confirm that the codebase governance is clearly explained, including how to influence codebase governance.
- Check that contributing guidelines cover who is expected to cover the costs of reviewing contributions.
- Check for a list of involved organizations.
- Check for a roadmap.
- Check for published activity statistics.
- Check for a code of conduct.

P Public policy makers: what you need to do

- Add a list to the codebase of any other resources that policy experts, non-governmental organizations and academics would find useful for understanding or reusing your policy.
- Consider adding contact details so that other policy makers considering collaboration can ask you for advice.

M Managers: what you need to do

- Make sure that the documentation of the governance includes the current process for how to make changes to the governance.
- If the community has some consensus about how the governance should change, then include those ideas stated as ambitions in the documentation.
- If needed, make sure you have allocated budget for the contributions review process as agreed by the codebase community.
- Make sure the documentation explains how each organization is involved in the codebase, what resources it has available for it and for how long.
- Support your experienced policy makers, developers and designers to stay part of the community for as long as possible.

D Developers and designers: what you need to do

- Respond promptly to requests.

- Keep your managers informed of the time and resources you require to support other contributors.
- Communicate clearly to contributors what they need to do make sure their contribution can be integrated.

Further reading

- Building welcoming communities by Open Source Guides. https://opensource.guide/building-community/
- The Open Source Contributor Funnel by Mike McQuaid. https://mikemcquaid.com/2018/08/14/the-open-source-contributor-funnel-why-people-dont-contribute-to-your-open-source-project/
- Leadership and governance for growing open source community projects, by Open Source Guides. https://opensource.guide/leadership-and-governance/
- Building online communities by Pieter Hintjens (long read!). http://hintjens.com/blog:117

Make contributing easy

To develop better, more reliable and feature rich software, users need to be able to fix problems, add features, and address security issues of the shared codebase.

A shared digital infrastructure makes it easier to make collaborative contributions. The less effort it takes to make contributions that are accepted by the codebase, the more likely users are to become contributors.

Requirements

- The codebase MUST have a public issue tracker that accepts suggestions from anyone.
- The documentation MUST link to both the public issue tracker and submitted codebase changes, for example in a `README` file.
- The codebase MUST have communication channels for users and developers, for example email lists.
- There MUST be a way to report security issues for responsible disclosure over a closed channel.
- The documentation MUST include instructions for how to report potentially security sensitive issues.

How to test

- Confirm that there is a public issue tracker.
- Confirm that the codebase contains links to the public issue tracker and submitted codebase changes.
- Confirm that it is possible to participate in a discussion with other users and developers about the software using channels described in the codebase.
- Confirm that there is a closed channel for reporting security issues.
- Confirm that there are instructions for privately reporting security issues.

P Public policy makers: what you need to do

- Track policy issues in the codebase, so that a relevant external policy expert can volunteer help.

M Managers: what you need to do

- Track management issues in the codebase, so that external managers with relevant experience can volunteer help.
- Support your experienced policy makers, developers and designers to keep contributing to the codebase for as long as possible.

D Developers and designers: what you need to do

- Just like for reviews, make sure to respond to requests promptly.
- Keep your managers informed of the time and resources you require to support other contributors.
- Make sure that appropriate communication channels for asking maintainers and stakeholders questions are easy to locate, for instance in the README.
- Make sure that appropriate contact details are included in the metadata, for instance in the publiccode.yml file.

Further reading

- How to inspire exceptional contributions to your open-source project by Joel Hans.
- The benefits of coding in the open by the UK Government Digital Service.

https://dev.to/joelhans/how-to-inspire-exceptional-contributions-to-your-open-source-project-1ebf

https://gds.blog.gov.uk/2017/09/04/the-benefits-of-coding-in-the-open/

Maintain version control

Version control means keeping track of changes to the code and other files of the codebase over time. This allows you to maintain structured documentation of the history of the codebase. This is essential for collaboration at scale, as it enables developers to work on changes in parallel and helps future developers to understand the reasons for changes.

Requirements

- All files in the codebase MUST be version controlled.
- All decisions MUST be documented in commit messages.
- Every commit message MUST link to discussions and issues wherever possible.
- The codebase SHOULD be maintained in a distributed version control system.
- Contribution guidelines SHOULD require contributors to group relevant changes in commits.
- Maintainers SHOULD mark released versions of the codebase, for example using revision tags or textual labels.
- Contribution guidelines SHOULD encourage file formats where the changes within the files can be easily viewed and understood in the version control system.
- It is OPTIONAL for contributors to sign their commits and provide an email address, so that future contributors are able to contact past contributors with questions about their work.

How to test

- Confirm that the codebase is kept in version control using software such as Git.

- Review the commit history, confirming that all commit messages explain why the change was made.
- Review the commit history, confirming that where possible all commit messages include the discussion about the change was or where to find it (with a URL).
- Check if the version control system is distributed.
- Review the commit history, check if grouping of relevant changes in accordance with the contributing guidelines.
- Check that it is possible to access a specific version of the codebase, for example through a revision tag or a textual label.
- Check that the file formats used in the codebase are text formats where possible.

P Public policy makers: what you need to do

- If a new version of the codebase is created because of a policy change, make sure it's clear in the documentation:
  - what the policy change is,
  - how it's changed the codebase.

For example, adding a new category of applicant to a codebase that manages granting permits would be considered a policy change.

M Managers: what you need to do

- Support policy makers, developers and designers to be clear about what improvements they're making to the codebase. Making improvements isn't a public relations risk.

D Developers and designers: what you need to do

- Make sure that all files required to understand the code, build and deploy are in the version control system.
- Write clear commit messages so that it is easy to understand why the commit was made.

- Mark different versions so that it is easy to access a specific version, for example using revision tags or textual labels.
- Write clear commit messages so that versions can be usefully compared.
- Work with policy makers to describe how the source code was updated after a policy change.

Further reading

- Producing OSS: Version Control Vocabulary by Karl Fogel. https://producingoss.com/en/vc.html#vc-vocabulary
- Maintaining version control in coding by the UK Government Digital Service. https://www.gov.uk/service-manual/technology/maintaining-version-control-in-coding
- GitHub Skills by GitHub for learning how to use GitHub or refresh your skills. https://skills.github.com/
- Git Cheat Sheet by GitHub, a list with the most common used git commands. https://education.github.com/git-cheat-sheet-education.pdf

Require review of contributions

Peer-review of contributions is essential for code quality, reducing security risks and operational risks.

Requiring thorough review of contributions encourages a culture of making sure every contribution is of high quality, completeness and value. Code review increases the chance of discovering and fixing potential bugs or mistakes before they are added to the codebase. Knowing that all code was reviewed discourages a culture of blaming individuals, and encourages a culture of focusing on solutions.

A policy of prompt reviews assures contributors of a guaranteed time for feedback or collaborative improvement, which increases both rate of delivery and contributor engagement.

Requirements

- All contributions that are accepted or committed to release versions of the codebase MUST be reviewed by another contributor.
- Reviews MUST include source, policy, tests and documentation.
- Reviewers MUST provide feedback on all decisions to not accept a contribution.
- The review process SHOULD confirm that a contribution conforms to the standards, architecture and decisions set out in the codebase in order to pass review.
- Reviews SHOULD include running both the code and the tests of the codebase.
- Contributions SHOULD be reviewed by someone in a different context than the contributor.
- Version control systems SHOULD NOT accept non-reviewed contributions in release versions.

- Reviews SHOULD happen within two business days.
- Performing reviews by multiple reviewers is OPTIONAL.

How to test

- Confirm that every commit in the history has been reviewed by a different contributor.
- Confirm that reviews include source, policy, tests and documentation.
- Confirm that rejected contributions were appropriately explained.
- Check if guidelines for reviewers include instructions to review for conformance to standards, architecture and codebase guidelines.
- Check with reviewers if they run the code and tests during review.
- Check with reviewers if commits have been reviewed by a different contributor in a different context.
- Check for use of branch protection in the version control system.
- Check the times between contribution submission and review.

P Public policy makers: what you need to do

- Institute a ‘four eyes’ policy where everything, not just code, is reviewed.
- Use a version control system or methodology that enables review and feedback.

M Managers: what you need to do

- Make delivering great code a shared objective.
- Make sure writing and reviewing contributions to source, policy, documentation and tests are considered equally valuable.
- Create a culture where all contributions are welcome and everyone is empowered to review them.
- Make sure no contributor is ever alone in contributing to a codebase.

D Developers and designers: what you need to do

- Ask other contributors on the codebase to review your work, in your organization or outside of it.
- Try to respond to others' requests for code review promptly, initially providing feedback about the concept of the change.

Further reading

- How to review code the GDS way by the UK Government Digital Service. https://gds-way.cloudapps.digital/manuals/code-review-guidelines.html#content
- Branch protection on GitHub and GitLab. https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/defining-the-mergeability-of-pull-requests/about-protected-branches https://about.gitlab.com/blog/2014/11/26/keeping-your-code-protected/
- The Gentle Art of Patch Review by Sage Sharp. https://sage.thesharps.us/2014/09/01/the-gentle-art-of-patch-review/
- Measuring Engagement by Mozilla. https://docs.google.com/presentation/d/1hsJLv1ieSqtXBzd5YZusY-mB8e1VJzaeOmh8Q4VeMio/edit#slide=id.g43d857af8_0177

Document codebase objectives

Clearly documenting codebase objectives communicates the purpose of the codebase. It helps stakeholders and contributors scope the development of the codebase. The objectives also provide an easy way for people to decide whether this codebase, or one of its modules, is interesting for them now or in the future.

Requirements

- The codebase MUST contain documentation of its objectives, like a mission and goal statement, that is understandable by developers and designers so that they can use or contribute to the codebase.
- Codebase documentation SHOULD clearly describe the connections between policy objectives and codebase objectives.
- Documenting the objectives of the codebase for the general public is OPTIONAL.

How to test

- Confirm that the codebase documentation includes the codebase objectives, mission or goal.
- Check for descriptions of connections between policy objectives and codebase objectives.

P Public policy makers: what you need to do

- Add the policy objectives to the codebase documentation, for example in the `README`.
- Make sure that all your codebase objectives have links or references to supporting policy documents added to meet the Bundle policy and source code criterion.

M Managers: what you need to do

- Add the organizational and business objectives to the codebase documentation, for example in the `README`.

D Developers and designers: what you need to do

- Add the technology and design objectives to the codebase documentation, for example in the `README`.

Further reading

- How to write project objectives by Marek Hajduczenia. http://grouper.ieee.org/groups/802/3/RTPGE/public/may12/hajduczenia_01_0512.pdf

Document the code

Well documented code helps people to understand what the code does and how to use it. Documentation is essential for people to start using the codebase and contributing to it more quickly.

Requirements

- All of the functionality of the codebase, policy as well as source, MUST be described in language clearly understandable for those that understand the purpose of the code.
- The documentation of the codebase MUST contain a description of how to install and run the source code.
- The documentation of the codebase MUST contain examples demonstrating the key functionality.
- The documentation of the codebase SHOULD contain a high level description that is clearly understandable for a wide audience of stakeholders, like the general public and journalists.
- The documentation of the codebase SHOULD contain a section describing how to install and run a standalone version of the source code, including, if necessary, a test dataset.
- The documentation of the codebase SHOULD contain examples for all functionality.
- The documentation SHOULD describe the key components or modules of the codebase and their relationships, for example as a high level architectural diagram.
- There SHOULD be continuous integration tests for the quality of the documentation.
- Including examples that make users want to immediately start using the codebase in the documentation of the codebase is OPTIONAL.

How to test

- Confirm that other stakeholders, professionals from other public organizations and the general public find the documentation clear and understandable.
- Confirm that the documentation describes how to install and run the source code.
- Confirm that the documentation includes examples of the key functionality.
- Check with members of the general public and journalists if they can understand the high level description.
- Check that the instructions for how to install and run a standalone version of the source code result in a running system.
- Check that all functionality documented contains an example.
- Check that the documentation includes a high level architectural diagram or similar.
- Check that the documentation quality is part of integration testing, for example documentation is generated from code, and links and images are tested.

P Public policy makers: what you need to do

- Check in regularly to understand how the non-policy code in the codebase has changed.
- Give feedback on how to make non-policy documentation more clear.

M Managers: what you need to do

- Try to use the codebase, so your feedback can improve how clearly the policy and code are documented. For example, is the current documentation sufficient to persuade a manager at another public organization to use this codebase?
- Make sure you understand both the policy and source code as well as the documentation.

D Developers and designers: what you need to do

- Check in regularly to understand how the non-source code in the codebase has changed.
- Give feedback on how to make non-source documentation more clear.

Further reading

- Documentation guide by Write the Docs. https://www.writethedocs.org/guide/

Use plain English

English is the *de facto* language of collaboration in software development.

Public sector information needs to be accessible to all its constituents. Plain and simple language makes the codebase and what it does easier to understand for a wider variety of people.

Translations further increase the possible reach of a codebase. Language that is easy to understand lowers the cost of creating and maintaining translations.

Requirements

- All codebase documentation MUST be in English.
- All code MUST be in English, except where policy is machine interpreted as code.
- All bundled policy not available in English MUST have an accompanying summary in English.
- Any translation MUST be up to date with the English version and vice versa.
- There SHOULD be no acronyms, abbreviations, puns or legal/non-English/domain specific terms in the codebase without an explanation preceding it or a link to an explanation.
- Documentation SHOULD aim for a lower secondary education reading level, as recommended by the Web Content Accessibility Guidelines 2.

https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=315#readable

- Providing a translation of any code, documentation or tests is OPTIONAL.

How to test

- Confirm that codebase documentation is available in English.
- Confirm that source code is in English, or confirm any non-English is policy or terms with preceding explanations.

- Confirm any non-English policy has an accompanying summary in English.
- Confirm that translations and the English version have the same content.
- Check that no unexplained acronyms, abbreviations, puns or legal/non-English/domain specific terms are in the documentation.
- Check the spelling, grammar and readability of the documentation.

P Public policy makers: what you need to do

- Frequently test with other managers, developers and designers in the process if they understand what you are delivering and how you document it.

M Managers: what you need to do

- Try to limit the use of acronyms, abbreviations, puns or legal/non-English/domain specific terms in internal communications in and between teams and stakeholders. Add any such terms to a glossary and link to it from the places they are being used.
- Be critical of documentation and descriptions in proposals and changes. If you don't understand something, others will probably also struggle with it.

D Developers and designers: what you need to do

- Frequently test with policy makers and managers if they understand what you are delivering and how you document it.
- Ask someone outside of your context if they understand the content (for example, a developer working on a different codebase).

Further reading

- Meeting the Web Content Accessibility Guidelines 2.1, Guideline 3.1 Readable by W3C makes text content readable and understandable. https://www.w3.org/TR/WCAG21/#readable
- TheEuropean Union accessibility directive by the European Commission, is an example of regulation requiring high accessibility. https://ec.europa.eu/digital-single-market/en/web-accessibility
- Definition of plain language by United States General Services Administration. https://www.plainlanguage.gov/about/definitions/

Use open standards

Open standards guarantee access to the knowledge required to use and contribute to the codebase. They enable interoperability between systems and reduce the risk of vendor lock-in. Open standards which are unambiguous allow for independent development of either side of data exchange.

Requirements

- For features of the codebase that facilitate the exchange of data the codebase MUST use an open standard that meets the Open Source Initiative Open Standard Requirements.
- Any non-open standards used MUST be recorded clearly as such in the documentation.
- Any standard chosen for use within the codebase MUST be listed in the documentation with a link to where it is available.
- Any non-open standards chosen for use within the codebase MUST NOT hinder collaboration and reuse.
- If no existing open standard is available, effort SHOULD be put into developing one.
- Open standards that are machine testable SHOULD be preferred over open standards that are not.
- Non-open standards that are machine testable SHOULD be preferred over non-open standards that are not.

https://opensource.org/osr

How to test

- Confirm that data exchange follows an OSI approved open standard.
- Confirm that any non-open standards used are clearly documented as such.
- Confirm that documentation includes a list of the standards followed within the codebase, each with a working link, or a statement that no standards were chosen.

P Public policy makers: what you need to do

- Mandate use of open standards everywhere possible.
- Prohibit procurement of technology that does not use open standards.

M Managers: what you need to do

- Consider including open standard compliance assessment in code reviews.

D Developers and designers: what you need to do

- Add continuous integration tests for compliance with the standards.
- Review the commits and other repository resources for references to standards and cross-check those with the standards listed.

Further reading

- Open Standards principles, policy paper of the UK Cabinet Office. https://www.gov.uk/government/publications/open-standards-principles/open-standards-principles

Use continuous integration

Asynchronous collaboration is enabled by developers merging their work to a shared branch frequently, verified by automated tests. The more frequent the merging and the smaller the contribution, the easier it is to resolve merge conflicts.

Automated testing of all functionality provides confidence that contributions are working as intended and have not introduced errors, and allows reviewers to focus on the structure and approach of the contribution. The more focused the test, the easier it is to clearly identify and understand errors as they arise.

Documenting a codebase's continuous integration workflow helps contributors understand the expectations of contributions. Continuous integration allows for an easier monitoring of the state of the codebase.

Requirements

- All functionality in the source code MUST have automated tests.
- Contributions MUST pass all automated tests before they are admitted into the codebase.
- The codebase MUST have guidelines explaining how to structure contributions.
- The codebase MUST have active contributors who can review contributions.
- Automated test results for contributions SHOULD be public.
- The codebase guidelines SHOULD state that each contribution should focus on a single issue.
- Source code test and documentation coverage SHOULD be monitored.
- Testing policy and documentation for consistency with the source and vice versa is OPTIONAL.

- Testing policy and documentation for style and broken links is OPTIONAL.
- Testing the code by using examples in the documentation is OPTIONAL.

How to test

- Confirm that there are tests present.
- Confirm that code coverage tools check that coverage is at 100% of the code.
- Confirm that contributions are only admitted into the codebase after all of the tests are passed.
- Confirm that contribution guidelines explain how to structure contributions.
- Confirm that there are contributions from within the last three months.
- Check that test results are viewable.
- Check if code coverage data is published.

P Public policy makers: what you need to do

- Involve managers as well as developers and designers as early in the process as possible and keep them engaged throughout development of your policy.
- Make sure there are also automated tests set up for policy documentation.
- Fix policy documentation promptly if it fails a test.
- Make sure the code reflects any changes to the policy (see Maintain version control).

M Managers: what you need to do

- Make sure to test with real end users as quickly and often as possible.
- Plan the work to integrate small parts very often instead of large parts less frequently.
- Procure consultancy services that deliver incrementally aligned with the plan.
- After a large failure, encourage publication of incident reports and public discussion of what was learned.

D Developers and designers: what you need to do

- Help managers structure the work plan such that it can be integrated as small increments.
- Help contributors limit the scope of their contributions and feature requests to be as small as reasonable.
- Help managers and policy makers test their contributions, for example by testing their contributions for broken links or style.
- Structure code written to handle conditions which are difficult to create in a test environment in such a way that the conditions can be simulated during testing. Forms of resource exhaustion such as running out of storage space and memory allocation failure are typical examples of difficult to create conditions.
- Tune the test code coverage tools to avoid false alarms resulting from inlining or other optimizations.
- Deploy often.
- Integrate your work at least once a day.

Further reading

- What is continuous integration by Martin Fowler. https://www.martinfowler.com/articles/continuousIntegration.html
- Use continuous delivery by the UK Government Digital Service. https://gds-way.cloudapps.digital/standards/continuous-delivery.html
- Quality assurance: testing your service regularly by the UK Government Digital Service. https://www.gov.uk/service-manual/technology/quality-assurance-testing-your-service-regularly

Publish with an open license

An open and well known license makes it possible for anyone to see the code in order to understand how it works, to use it freely and to contribute to the codebase. This enables a vendor ecosystem to emerge around the codebase.

Clearly indicating the license for each file within a codebase facilitates correct reuse and attribution of parts of a codebase.

Requirements

- All code and documentation MUST be licensed such that it may be freely reusable, changeable and redistributable.
- Software source code MUST be licensed under an OSI-approved or FSF Free/Libre license.

https://spdx.org/licenses/

- All code MUST be published with a license file.
- Contributors MUST NOT be required to transfer copyright of their contributions to the codebase.
- All source code files in the codebase SHOULD include a copyright notice and a license header that are machine-readable.
- Having multiple licenses for different types of code and documentation is OPTIONAL.

How to test

- Confirm that the codebase is clearly licensed.
- Confirm that the license for the source code is on the OSI-approved or FSF Free/Libre license list and the license for documentation conforms to the Open Definition.

https://spdx.org/licenses/

https://opendefinition.org/licenses/

- Confirm that the licenses used in the codebase are included as files.

- Confirm that contribution guidelines and repository configuration do not require transfer of copyright.
- Check for machine-readable license checking in the codebase continuous integration tests.

P Public policy makers: what you need to do

- Develop policy that requires code to be open source.
- Develop policy that disincentivizes non-open source code and technology in procurement.

M Managers: what you need to do

- Only work with open source vendors that deliver their code by publishing it under an open source license.
- Beware that even though Creative Commons licenses are great for documentation, licenses that stipulate Non-Commercial or No Derivatives are NOT freely reusable, changeable and redistributable and don't fulfill these requirements.

https://creativecommons.org/licenses/

D Developers and designers: what you need to do

- Add a new `license` file to every new codebase created.
- Add a copyright notice and a license header to every new source code file created.
- When code is being reused by the codebase, make sure that it has a license that is compatible with the license or licenses of the codebase.

Further reading

- Open source definition by the Open Source Initiative, all open source licenses meet this definition. https://opensource.org/osd
- Animated video introduction to Creative Commons by Creative Commons Aotearoa New Zealand. https://creativecommons.org/about/videos/creative-commons-kiwi
- REUSE Initiative specification by Free Software Foundation Europe for unambiguous, human-readable and machine-readable copyright and licensing information. https://reuse.software/spec/
- SPDX License List by the Linux Foundation with standardized, machine-readable abbreviations for most licenses. https://spdx.org/licenses/

Make the codebase findable

The more findable a codebase is, the more potential new collaborators will find it. Just publishing a codebase and hoping it is found does not work, instead proactiveness is needed.

A metadata description file increases discoverability. Well-written metadata containing a unique and persistent identifier, such as a Wikidata item or FSF software directory listing (thus being part of the semantic web), makes the codebase easier for people to refer, cite, disambiguate and discover through third party tools.

Requirements

- The name of the codebase SHOULD be descriptive and free from acronyms, abbreviations, puns or organizational branding.
- The codebase SHOULD have a short description that helps someone understand what the codebase is for or what it does.
- Maintainers SHOULD submit the codebase to relevant software catalogs.
- The codebase SHOULD have a website which describes the problem the codebase solves using the preferred jargon of different potential users of the codebase (including technologists, policy experts and managers).
- The codebase SHOULD be findable using a search engine by codebase name.
- The codebase SHOULD be findable using a search engine by describing the problem it solves in natural language.
- The codebase SHOULD have a unique and persistent identifier where the entry mentions the major contributors, repository location and website.

- The codebase SHOULD include a machine-readable metadata description, for example in a publiccode.yml file.

https://github.com/publiccodeyml/publiccode.yml

- A dedicated domain name for the codebase is OPTIONAL.
- Regular presentations at conferences by the community are OPTIONAL.

How to test

- Check that the codebase name is free of acronyms, abbreviations, puns or organizational branding.
- Check that the codebase repository has a short description of the codebase.
- Check for the codebase listing in relevant software catalogs.
- Check for a codebase website which describes the problem the codebase solves.
- Check that the codebase appears in the results on more than one major search engine when searching by the codebase name.
- Check that the codebase appears in the results on more than one major search engine when searching by using natural language, for instance, using the short description.
- Check unique and persistent identifier entries for mention of the major contributors.
- Check unique and persistent identifier entries for the repository location.
- Check unique and persistent identifier entries for the codebase website.
- Check for a machine-readable metadata description file.

P Public policy makers: what you need to do

- Contribute a description of the policy area or problem this codebase acts on or operates.
- Test your problem description with peers outside of your context who aren't familiar with the codebase.

- Present on how the codebase implements the policy at relevant conferences.

M Managers: what you need to do

- Search trademark databases to avoid confusion or infringement before deciding the name.
- Use the short description wherever the codebase is referenced, for instance, as social media account descriptions.
- Budget for content design and Search Engine Optimization skills in the team.
- Make sure people involved in the project present at relevant conferences.

D Developers and designers: what you need to do

- Search engine optimization, for instance adding a sitemap.

https://www.sitemaps.org/protocol.html

- Use the short description wherever the codebase is referenced, for instance, as the repository description.
- Test your problem description with peers outside of your context who aren't familiar with the codebase.
- Suggest conferences to present at and present at them.

Further reading

- Introduction to Wikidata by the Wikidata community.

https://www.wikidata.org/wiki/Wikidata:Introduction

- FSF software directory listing by the Free Software Foundation.

https://directory.fsf.org/wiki/Main_Page

- The FAIR Guiding Principles for scientific data management and stewardship by the GO FAIR International Support and Coordination Office provide a nice list of attributes that make (meta)data more machine actionable (and hence more findable). Some of these apply directly to codebases, while others may provoke exploration into what the codebase equivalent would be.

https://www.go-fair.org/fair-principles/

Use a coherent style

Following a consistent and coherent style enables contributors in different environments to work together. Unifying vocabularies reduces friction in communication between contributors.

Requirements

- The codebase MUST use a coding or writing style guide, either the codebase community's own or an existing one referred to in the codebase.
- Contributions SHOULD pass automated tests on style.
- The style guide SHOULD include expectations for inline comments and documentation for non-trivial sections.
- Including expectations for understandable English in the style guide is OPTIONAL.

How to test

- Confirm that contributions are in line with the style guides specified in the documentation.
- Check for the presence of automated tests on style.

P Public policy makers: what you need to do

- Create, follow and continually improve on a style guide for policy and documentation as well as document this in the codebase, for example in the `CONTRIBUTING` or `README`.

M Managers: what you need to do

- Include written language, source, test and policy standards in your organizational definition of quality.

D Developers and designers: what you need to do

If the codebase does not already have engineering guidelines or other contributor guidance, start by adding documentation to the repository describing whatever is being done now, for example in the `CONTRIBUTING` or `README`. An important purpose of the file is to communicate design preferences, naming conventions, and other aspects machines can't easily check. Guidance should include what would be expected from code contributions in order for them to be merged by the maintainers, including source, tests and documentation. Continually improve upon and expand this documentation with the aim of evolving this documentation into engineering guidelines.

Additionally:

- Use a linter.
- Add linter configurations to the codebase.

Further reading

- Programming style on Wikipedia. https://en.wikipedia.org/wiki/Programming_style

Document codebase maturity

Clearly signalling a codebase's maturity helps others decide whether to use and contribute to it. A codebase version's maturity includes the maturity of its dependencies. Understanding how a codebase has evolved is key to understanding the codebase and how to contribute to it.

Requirements

- The codebase MUST be versioned.
- The codebase MUST prominently document whether or not there are versions of the codebase that are ready to use.
- Codebase versions that are ready to use MUST only depend on versions of other codebases that are also ready to use.
- The codebase SHOULD contain a log of changes from version to version, for example in the `CHANGELOG`.
- The method for assigning version identifiers SHOULD be documented.
- It is OPTIONAL to use semantic versioning.

How to test

- Confirm that the codebase has a strategy for versioning which is documented.
- Confirm that it is obvious to policy makers, managers, developers and designers whether the codebase has versions that are ready to use.
- Confirm that ready to use versions of the codebase do not depend on any versions of other codebases that are not ready to use.
- Check that the versioning scheme of the codebase is documented and followed.

- Check that there is a log of changes.

P Public policy makers: what you need to do

- When developing policy, understand that any code developed needs to be tested and improved before it can be put into service.
- Consider versioning policy changes, especially when they trigger new versions of the source code.

M Managers: what you need to do

- Make sure that services only rely on versions of codebases of equal or greater maturity than the service. For example, don't use a beta version of a codebase in a production service.

D Developers and designers: what you need to do

- Make sure that the codebase versioning approach is followed for all releases.

Further reading

- Semantic Versioning Specification used by many codebases to label versions. https://semver.org/
- Software release life cycle https://en.wikipedia.org/wiki/Software_release_life_cycle
- Service Design and Delivery Process by the Australian Digital Transformation Agency. https://www.dta.gov.au/help-and-advice/build-and-improve-services/service-design-and-delivery-process
- Service Manual on Agile Delivery by the UK Government Digital Service. https://www.gov.uk/service-manual/agile-delivery

Contributing to this standard

🙇 Thank you for contributing!

We understand that a standard like this can only be set in collaboration with as many public technologists, policy makers and interested folk as possible. Thus we appreciate your input, enjoy feedback and welcome improvements to this project and are very open to collaboration.

We love issues and pull requests from everyone. If you're not comfortable with GitHub, you can email use your feedback at info@publiccode.net.

Problems, suggestions and questions in issues

A high-level overview of the development that we already have sketched out can be seen in the roadmap. Please help development by reporting problems, suggesting changes and asking questions. To do this, you can create a GitHub issue for this project in the GitHub Issues for the Standard for Public Code.

https://docs.github.com/en/issues/tracking-your-work-with-issues/creating-an-issue

https://github.com/publiccodenet/standard/issues

Or, sign up to the mailing list and send an email to standard@lists.publiccode.net.

https://lists.publiccode.net/mailman/postorius/lists/standard.lists.publiccode.net/

You don't need to change any of our code or documentation to be a contributor!

Documentation and code in pull requests

If you want to add to the documentation or code of one of our projects you should make a pull request.

If you never used GitHub, get up to speed with Understanding the GitHub flow or follow one of the great free interactive courses in GitHub Skills on working with

https://docs.github.com/en/get-started/quickstart/github-flow

https://skills.github.com/

GitHub and working with MarkDown, the syntax this project's documentation is in.

This project is licensed Creative Commons Zero v1.0 Universal, which essentially means that the project, along with your contributions is in the public domain in whatever jurisdiction possible, and everyone can do whatever they want with it.

1. Make your changes

Contributions should follow the requirements set out in the criteria of the Standard for Public code itself. Reviewers will also be ensuring that contributions are aligned with the values of public code. Furthermore, they will review that the contribution conforms to the standards `70` and remains coherent with the overall work.

This project uses the GitFlow branching model and workflow. When you've forked this repository, please make sure to create a feature branch following the GitFlow model.

https://nvie.com/posts/a-successful-git-branching-model/

Add your changes in commits with a message that explains them. If more than one type of change is needed, group logically related changes into separate commits. For example, white-space fixes could be a separate commit from text content changes. When adding new files, select file formats that are easily viewed in a `diff`, for instance, `.svg` is preferable to a binary image. Document choices or decisions you make in the commit message, this will enable everyone to be informed of your choices in the future.

https://robots.thoughtbot.com/5-useful-tips-for-a-better-commit-message

If you are adding code, make sure you've added and updated the relevant documentation and tests before you submit your pull request. Make sure to write tests that show the behavior of the newly added or changed code.

Applicable policy

Currently, the Standard for Public Code is not implementing any specific public policy.

Style

The Standard for Public Code aims to use plain English and we have chosen American English for spelling. Text content should typically follow one line per sentence, with no line-wrap, in order to make `diff` output easier to view. However, we want to emphasize that it is more important that you make your contribution than worry about spelling and typography. We will help you get it right in our review process and we also have a separate quality check before making a new release.

Standards to follow

These are the standards that the Standard for Public Code uses. Please make sure that your contributions are aligned with them so that they can be merged more easily.

- IETF RFC 2119 - for requirement level keywords https://tools.ietf.org/html/rfc2119
- Web Content Accessibility Guidelines 2.1 - for readability https://www.w3.org/TR/WCAG21/#readable

2. Pull request

When submitting the pull request, please accompany it with a description of the problem you are trying to solve and the issue number that this pull request fixes. It is preferred for each pull request to address a single issue where possible. In some cases a single set of changes may address multiple issues, in which case be sure to list all issue numbers fixed.

3. Improve

All contributions have to be reviewed by someone. The Foundation for Public Code has committed to make sure that maintainers are available to review contributions with an aim to provide feedback within two business days.

It could be that your contribution can be merged immediately by a maintainer. However, usually, a new pull request needs some improvements before it can be merged. Other contributors (or helper robots) might have feedback. If this is the case the reviewing maintainer will help you improve your documentation and code.

If your documentation and code have passed human review, it is merged.

4. Celebrate

Your ideas, documentation and code have become an integral part of this project. You are the open source hero we need!

In fact, feel free to open a pull request to add your name to the `AUTHORS` file and get eternal attribution.

Translations in other languages

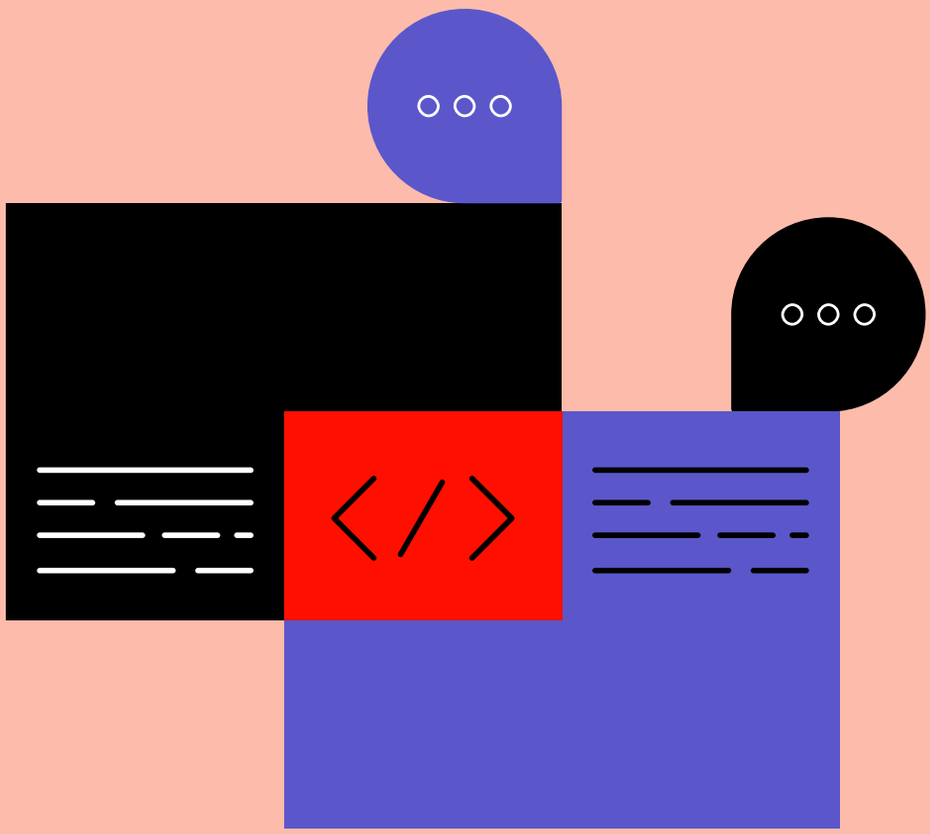
While the Standard does not have any official translations, you can help maintain existing and add new community translations of the Standard.

https://github.com/publiccodenet/community-translations-standard

Releases

We have dedicated documentation for creating new releases and ordering printed standards.

For more information on how to use and contribute to this project, please read the `README`.



Code of Conduct

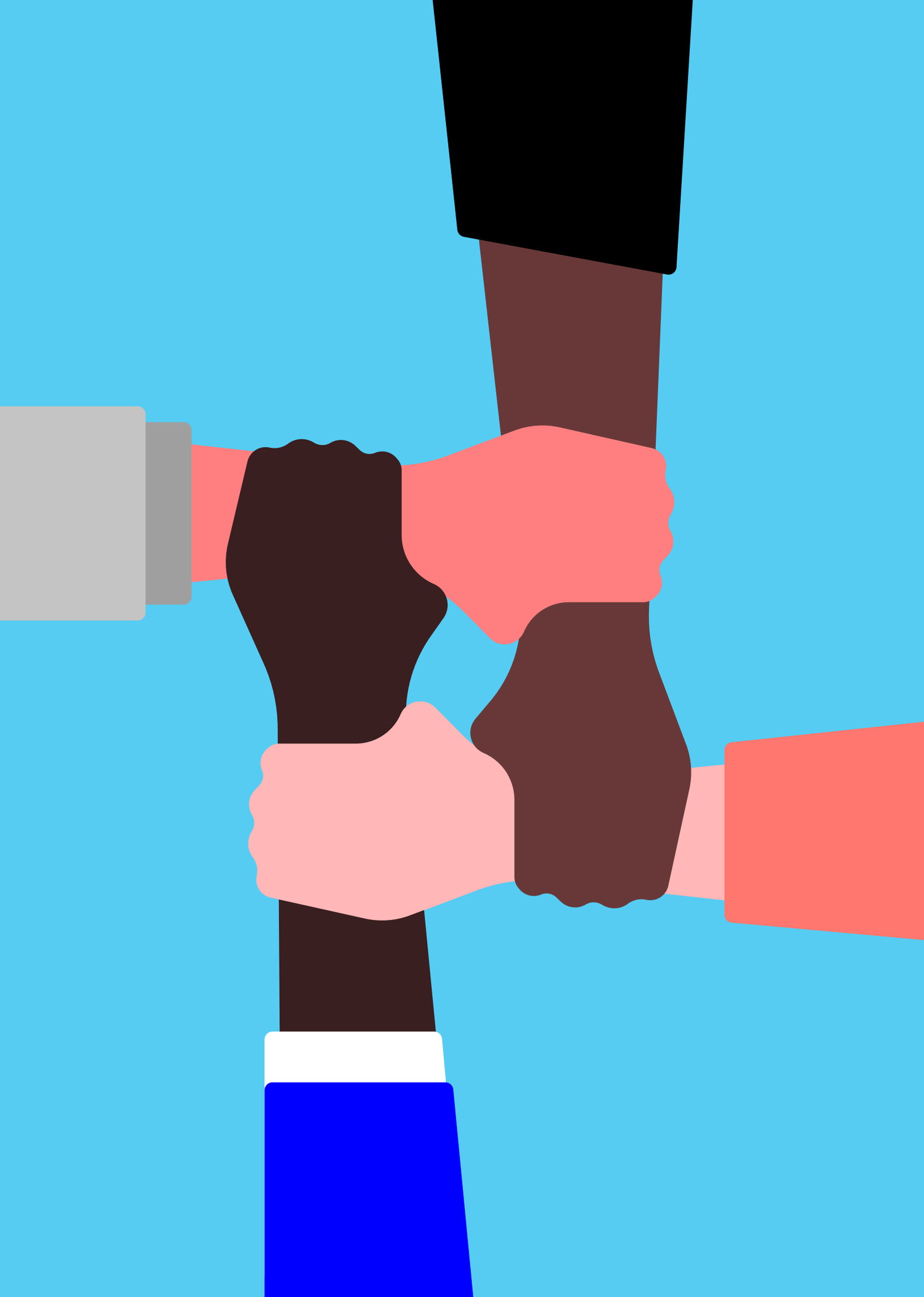
Many community members are from civic or professional environments with behavioral codes yet some individuals are not. This document expresses expectations of all community members and all interactions regardless of communication channel.

Be here to collaborate.

Be considerate, respectful, and patient.

Strive to be as constructive as possible.

To raise a concern, please email directors@publiccode.net.



Governance

This standard lies at the core of the codebase stewardship provided by the Foundation for Public Code. We decide if a codebase is ready for community co-development based on this document.

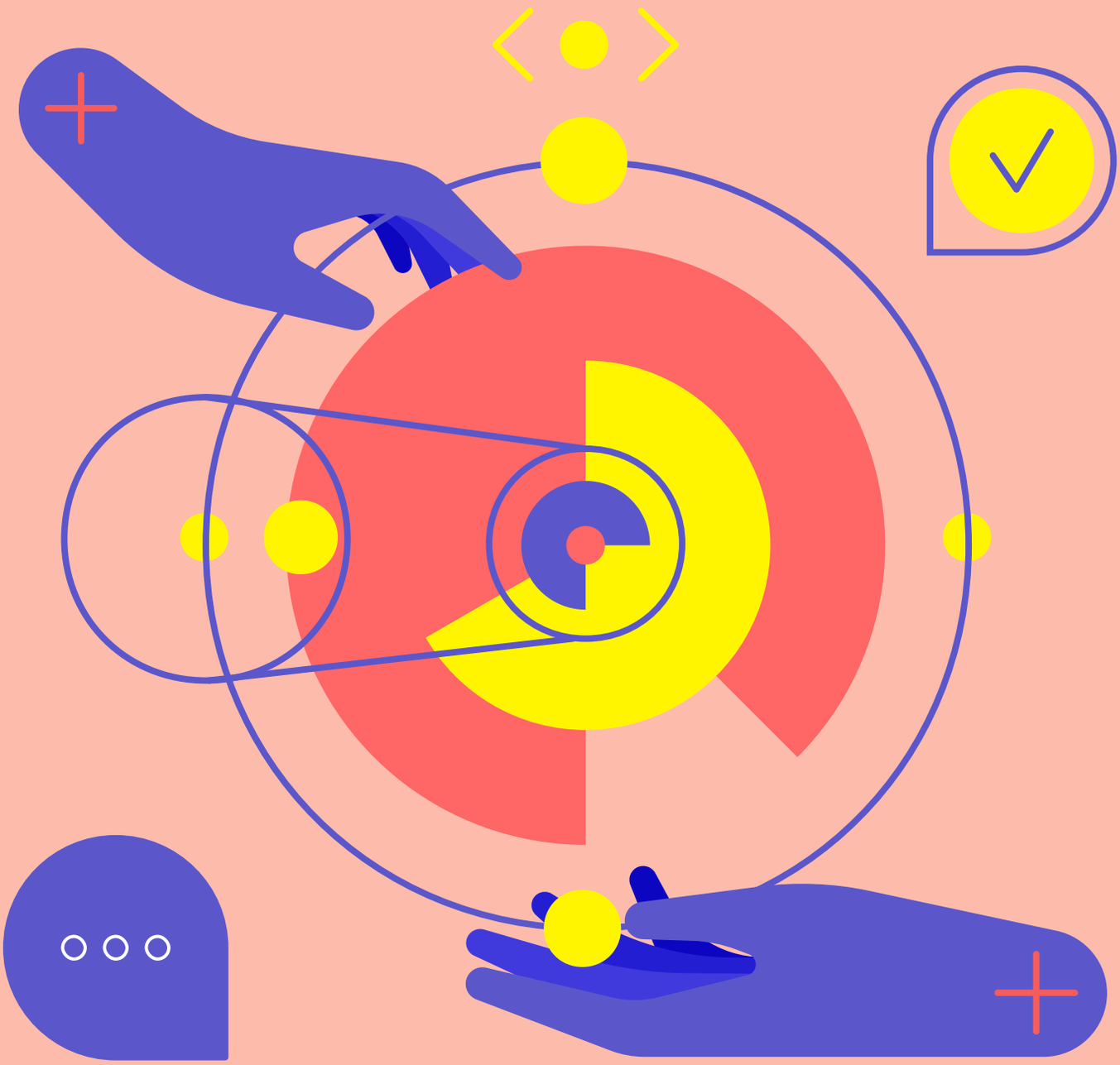
https://publiccode.net/

The standard is maintained by Foundation for Public Code staff.

We welcome contributions, such as suggestions for changes or general feedback, from anyone.

Because of the important role that the Standard for Public Code has in our core process we require the highest standards from the Standard.

We will try to respond promptly to all pull requests. The pull request is an opportunity to work together to improve our methods and the Standard. We may not accept all changes, but we will explain our logic.



Version history

Version 0.7.0

May 31st 2023: 📑 the fifteenth draft adds new requirements for documenting review funding and clarifies review process requirement.

- Add requirement to document who is expected to cover the cost of reviewing contributions.
- Add requirement to have a short description of the codebase.
- Change the focus of contributions adhering to standards to focus on the review of contributions.
- Relaxed MUST requirements to SHOULD in Make the codebase findable.
- Review template now in HTML format.
- Introduction converted to foreword.
- Improved contributing guidelines.
- Improved documentation of scripts.

Version 0.6.0

April 20th 2023: 🔀 the fourteenth draft adds new requirements for portability and tests and an introduction to each criterion.

- New requirement in Create reusable and portable code about the development being a collaboration between multiple parties.
- New requirement in Create reusable and portable code about being dependent on a single vendor.
- New requirement in Use continuous integration about publishing results for automated tests.
- Differentiating the two requirements about security to clearly be about providing a method and having documentation about it.
- Rephrased requirements to focus on the codebase rather than contributor behavior.

- Removed the sections Why this is important and What this does not do and replaced with an introduction in each criterion.
- Added general What this does not do section in the introduction of the Standard.
- Added guidance for public policy makers about related policies and license compatibility.
- Added guidance for developers and designers about version controlling files.
- Clarified guidance for developers and designers about prompt responses and search engine optimization.
- Added Further reading about accessibility.
- Aligned criteria URLs with criteria names.
- Improved navigation in the web version.
- Moved tools in Further reading sections to the community implementation guide.
- Moved compliance or certification process to publiccode.net.
- Change format of the review template to make it easier to update after a new release.
- Improved the text on the landing page and added links to related resources.
- Added spell checker automated test.
- Made minor changes to text for clarity and consistency.
- Moved SPDX headers to yaml header.

Version 0.5.0

January 25th 2023: 🎨 the thirteenth draft focuses on documenting style guidelines.

- Adjust the coding style requirement to focus on the codebase using a style guide rather than contributor behavior.
- Moved requirement for codebase names to Make the codebase findable from Use plain English.
- Moved requirement about testing the code by using examples to Use continuous integration from Document the code.
- Split requirement about machine testable standards to clarify that open is more important than testable.

- Adjust how to test findability requirements to be less reliant on search engine algorithms.
- Made minor changes to text for clarity and consistency.

Version 0.4.1

December 5th 2022: 📝 the twelfth draft clarifies document maturity.

- Document maturity focuses on whether or not versions of the codebase are ready to use.
- Document maturity no longer requires specific labels for codebases that are not ready to use.
- Audit flow image now generated from an easier to translate format.
- Improved guidance on How to test.
- Add publiccode.yml file.
- Add review template.
- Consistently link glossary terms.
- Add practices and standards to follow in CONTRIBUTING.
- Add Matti Schneider to Authors.
- Add remaining SPDX headers to files.
- Made additional minor changes to text for clarity.
- Some hyperlinks updated.
- Moved examples to the Community implementation guide.

https://publiccodenet.github.io/community-implementation-guide-standard/

Version 0.4.0

September 7th 2022: 🔭 the eleventh draft adds a new findability criterion.

- Introduce new criterion: Make the codebase findable.
- Improve How to test section for most criteria.
- New requirement in Welcome contributors about publishing activity statistics.
- Removed redundant requirement about portable and reusable code.
- Expand open license definition to include both OSI and FSF approved licenses.

- Rephrase MAY requirements to use the keyword OPTIONAL for clarity.
- Expressed intent that the Standard for Public Code should meet its own requirements where applicable and added assessment.
- Add SPDX license identifiers to files.
- Introduced new Code of Conduct.
- Clarify distinction between source code and policy text.
- Restructuring of requirements with bullet point lists.
- Acknowledge the importance of codebase modularity for reuse.
- Move requirements related to Findability to the new criterion.
- Clarify the role of non-open standards when used in a codebase.
- Additional guidance about build-time and runtime dependencies.
- Added roadmap for the development of the Standard for Public Code.
- Update structure of Authors file.
- Add Audrey Tang to Authors.
- Added a list of criteria to the print edition.
- Clarify what the standard means with policymakers, managers, developers and designers.
- Made additional minor changes to text for clarity.
- Some hyperlinks updated.

Version 0.3.0

May 23rd 2022: the tenth draft strengthens documentation and localization.

- Requirement for localization made explicit in Create reusable and portable code.
- Documentation of governance changed from a SHOULD to a MUST.
- Replace the very subjective (and hard to test) “contributions MUST be small” with requirement to document expectation in contributing guidelines and focus on a single issue.

- Community translations now linked in the footer.
- Revert “Replace BPMN svg with Mermaid flowchart”.
- Many minor clarifications to language and sentences made more simple.
- Some hyperlinks updated.

Version 0.2.3

March 15th 2022: 📜 the ninth draft allows English summaries for policy lacking an official translation.

- Relax the criterion Use plain English by adding a new requirement allows bundled policy not available in English to have an accompanying summary in English instead of translating the full text.
- Similarly, allow for English summaries for policies not available in English in Bundle policy and code.
- Clarify that term ‘policy’ includes processes which impact development and deployment in Bundle policy and code.
- Emphasize reusability also on parts of the solutions in Create reusable and portable code.
- Expand guidance to Developers and designers in Create reusable and portable code about deploying to proprietary platforms.
- Add nuance to use of non-English terms in what management need to do in Use plain English.
- Change the pull request process diagram to use Mermaid instead of BPMN to make community translations easier.

https://github.com/publiccodenet/community-translations-standard

- Added Maurice Hendriks to AUTHORS.
- Added OpenApi Specification to further reading.
- Made the attributions in further reading sections clearer.
- Made additional minor changes to text for clarity.

Version 0.2.2

November 29th 2021: 🏛️ the eighth draft recognizes that policy which executes as code may not be in English.

- Document exception to “All code MUST be in English” where policy is interpreted as code.
- Add MAY requirement regarding committer email addresses in Maintain version control.
- Expand guidance to Policy Makers in Bundle policy and code.
- Expand guidance to Developers and designers in Use a coherent style.
- Add “Different contexts” to glossary.
- Add Mauko Quiroga and Charlotte Heikendorf to AUTHORS.
- Add Digital Public Goods approval badge.
- Added “next” and “previous” links to criteria pages of web version.
- Add Open Standards principles to further reading.
- Add Definition of plain language to further reading.
- Move the Semantic Versioning Specification further reading reference.
- Clarify that publiccode.yml is one example of a machine-readable metadata description.
- Changed “your codebase” and “your organization” to be less possessive.
- Made additional minor changes to text for clarity.
- Add instructions for creating a print version.

Version 0.2.1

March 1st 2021: 🧽 the seventh draft has minor cleaning up after version 0.2.0.

- New SHOULD requirement on using a distributed version control system and why distributed is important.
- Feedback requirements for rejected contributions are more strict than accepted ones.
- Specify that copyright and license notices should also be machine-readable.

- Advice on how to test that notices be machine-readable.
- Clarify guidance for rolling releases.
- Clear up definition of version control in glossary.
- Add further reading encouraging contribution, SPDX, Git and reviewing contributions.
- Add links to videos about the concept of public code.
- Update BPMN link.
- Reduce link duplication.
- Add Alba Roza and Ngô Ngọc Đức Huy to authors.
- Made additional minor changes to text for clarity.

Version 0.2.0

October 26th 2020: 🎊 the sixth draft splits a requirement and adds clarity.

- Split “Welcome contributions” criterion into “Make contributing easy” and “Welcome contributors”.
- Rename criterion “Pay attention to codebase maturity” to “Document codebase maturity”.
- Changed MUST to SHOULD for requirement of codebase in use by multiple parties.
- Add MUST NOT requirement regarding copyright assignment.
- Clarify role of configuration in reusable code requirement.
- Glossary additions: continuous integration, policy, repository, and version control.
- Replace references to ‘cities’ with ‘public organizations’.
- Clarify aspects of sensitive code by separating contributor and reviewer requirements into separate items.
- Expand further reading, and guidance to policy makers, developers and designers.
- Add Felix Faassen and Arnout Engelen to authors.
- Made additional minor changes to text for clarity.

Version 0.1.4

November 27th 2019: 🧹 the fifth draft consists mostly of additional minor fixes.

- Linked License.md file.
- Add Sky Bristol, Marcus Klaas de Vries, and Jan Ainali to authors.
- Made punctuation more consistent, especially for bullet lists.
- Made some minor changes to text for clarity.

Version 0.1.3

October 8th 2019: 🍂 the fourth draft only patches and fixes minor things for the autumn cleaning

- Renamed continuous delivery to continuous integration.
- Referencing accessibility guidelines in the language standard.
- A bunch of style and consistency fixes.

Version 0.1.2

August 22th 2019: 🌠 the third draft focuses on better text and takes community input

- With some great new contributors comes a fresh author list.
- All links are now HTTPS.
- General proofreading, wording clarifications, and smashed typos.
- Updated criteria:
  - Requirement for reuse in different contexts
  - Recommendation for explicit versioning
  - Recommendation for multi party development
  - Recommendation for license headers in files
  - Recommendation for vulnerability reporting
  - Recommendation for explicit documentation of governance

Version 0.1.1

May 9th 2019: 🤔 the second draft fixes a few basic oversights and fixes a lot of typos

- Removed references to the Foundation for Public Code, we're going to have to change the name in becoming an association.
- Updated the introduction.
- Updated the glossary.
- Added the code of conduct.
- We've recommended using the publiccode.yml standard for easier reuse.

Version 0.1.0

April 16th 2019: 🎉 the first draft is ready, it is all brand new and has snazzy new ideas in it

- 14 criteria with their requirements and how to operationalize them.
- An introduction with a high level background, what this standard is, and how the Foundation for Public Code will use it.

This first version was produced together with the Amsterdam University of Applied Sciences and the City of Amsterdam as a part of the Smart Cities? Public Code! project.

https://smartcities.publiccode.net/

This license is the legal contract that allows anyone to do anything they like with the content in this entire document.

CC0 1.0 Universal

```
Creative Commons Legal Code

CC0 1.0 Universal

    CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE
    LEGAL SERVICES. DISTRIBUTION OF THIS DOCUMENT DOES NOT CREATE AN
    ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS
    INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES
    REGARDING THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS
    PROVIDED HEREUNDER, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM
    THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS PROVIDED
    HEREUNDER.

Statement of Purpose

The laws of most jurisdictions throughout the world automatically confer
exclusive Copyright and Related Rights (defined below) upon the creator
and subsequent owner(s) (each and all, an "owner") of an original work of
authorship and/or a database (each, a "Work").

Certain owners wish to permanently relinquish those rights to a Work for
the purpose of contributing to a commons of creative, cultural and
scientific works ("Commons") that the public can reliably and without fear
of later claims of infringement build upon, modify, incorporate in other
works, reuse and redistribute as freely as possible in any form whatsoever
and for any purposes, including without limitation commercial purposes.
These owners may contribute to the Commons to promote the ideal of a free
culture and the further production of creative, cultural and scientific
works, or to gain reputation or greater distribution for their Work in
part through the use and efforts of others.

For these and/or other purposes and motivations, and without any
expectation of additional consideration or compensation, the person
associating CC0 with a Work (the "Affirmer"), to the extent that he or she
is an owner of Copyright and Related Rights in the Work, voluntarily
elects to apply CC0 to the Work and publicly distribute the Work under its
terms, with knowledge of his or her Copyright and Related Rights in the
Work and the meaning and intended legal effect of CC0 on those rights.

1. Copyright and Related Rights. A Work made available under CC0 may be
protected by copyright and related or neighboring rights ("Copyright and
Related Rights"). Copyright and Related Rights include, but are not
limited to, the following:

  i. the right to reproduce, adapt, distribute, perform, display,
```

communicate, and translate a Work;
ii. moral rights retained by the original author(s) and/or performer(s);
iii. publicity and privacy rights pertaining to a person's image or likeness depicted in a Work;
iv. rights protecting against unfair competition in regards to a Work, subject to the limitations in paragraph 4(a), below;
v. rights protecting the extraction, dissemination, use and reuse of data in a Work;
vi. database rights (such as those arising under Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, and under any national implementation thereof, including any amended or successor version of such directive); and
vii. other similar, equivalent or corresponding rights throughout the world based on applicable law or treaty, and any national implementations thereof.

2. Waiver. To the greatest extent permitted by, but not in contravention of, applicable law, Affirmer hereby overtly, fully, permanently, irrevocably and unconditionally waives, abandons, and surrenders all of Affirmer's Copyright and Related Rights and associated claims and causes of action, whether now known or unknown (including existing as well as future claims and causes of action), in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "Waiver"). Affirmer makes the Waiver for the benefit of each member of the public at large and to the detriment of Affirmer's heirs and successors, fully intending that such Waiver shall not be subject to revocation, rescission, cancellation, termination, or any other legal or equitable action to disrupt the quiet enjoyment of the Work by the public as contemplated by Affirmer's express Statement of Purpose.

3. Public License Fallback. Should any part of the Waiver for any reason be judged legally invalid or ineffective under applicable law, then the Waiver shall be preserved to the maximum extent permitted taking into account Affirmer's express Statement of Purpose. In addition, to the extent the Waiver is so judged Affirmer hereby grants to each affected person a royalty-free, non transferable, non sublicensable, non exclusive, irrevocable and unconditional license to exercise Affirmer's Copyright and Related Rights in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "License"). The License shall be deemed effective as of the date CC0 was applied by Affirmer to the Work. Should any part of the License for any reason be judged legally invalid or ineffective under applicable law, such partial invalidity or ineffectiveness shall not invalidate the remainder of the License, and in such case Affirmer hereby affirms that he or she will not (i) exercise any of his or her remaining Copyright and Related Rights in the Work or (ii) assert any associated claims and causes of action with respect to the Work, in either case contrary to Affirmer's

express Statement of Purpose.

4. Limitations and Disclaimers.

a. No trademark or patent rights held by Affirmer are waived, abandoned, surrendered, licensed or otherwise affected by this document.
b. Affirmer offers the Work as-is and makes no representations or warranties of any kind concerning the Work, express, implied, statutory or otherwise, including without limitation warranties of title, merchantability, fitness for a particular purpose, non infringement, or the absence of latent or other defects, accuracy, or the present or absence of errors, whether or not discoverable, all to the greatest extent permissible under applicable law.
c. Affirmer disclaims responsibility for clearing rights of other persons that may apply to the Work or any use thereof, including without limitation any person's Copyright and Related Rights in the Work. Further, Affirmer disclaims responsibility for obtaining any necessary consents, permissions or other rights required for any use of the Work.
d. Affirmer understands and acknowledges that Creative Commons is not a party to this document and has no duty or obligation with respect to this CC0 or use of the Work.