Secure by Design Artefact Library - GitHub Management Process

**Table of Contents**

[Background 2](#_pdhy5ohw5i2)

[Purpose 2](#_nqxe14k1vhgx)

[Types of artefacts 2](#_sgs3dc6sbg0t)

[Key Roles 3](#_rxowvtuac9ai)

[Artefact development process 5](#_jbvvmhxrmr25)

[Request 5](#_xafs1hmnmjyi)

[Review 5](#_wb5q0vq5001c)

[Create (or change) 6](#_9ult21yqkmw)

[Integrate (Beta) 6](#_pioty0yd7345)

[Publish (Live) 6](#_ba7qmfasuny5)

[Annex 7](#_uztqgl93f2qo)

[Annex 1 - Github terms and definitions 7](#_ya80bqaxc1mk)

[Annex 2 - Detailed artefact development process 8](#_ytqabsos8spj)

[Annex 3 - Github Instructions 9](#_4q1xfn8pyz5l)

[Request a new artefact or update to an existing artefact 9](#_2k4rxw24xl3q)

[Review the new request 9](#_ffp2uxn7u53e)

[Create Artefact 10](#_tt5t5g7k5e61)

[Integrate Artefact 12](#_ajli06b29869)

[Publish Artefact 13](#_s35mnp5e4r31)

# 

# Background

The Secure by Design (SbD) Artefact Library provides a centralised repository of proven solutions to common cyber security problems that support the implementation of SbD principles across government. The library is hosted on Github and provides essential resources for:

* **Guidance**: Enabling teams, regardless of their cyber expertise, to embed security into project delivery by providing practical tools and frameworks.
* **Standardisation**: Offering consistent approaches and best practices for addressing security risks across various departments.
* **Efficiency**: Reducing duplication by reusing existing security solutions and materials.

It is an open repository, accessible to government, vendors and private sector. The resources are aimed mainly at technical and security architects, developers, DevSecOps, but could be useful to anyone in the digital delivery teams.

# Purpose

The purpose of this document is to outline the management processes of the Secure by Design artefacts library hosted on the Cabinet Office Github. This includes key steps in relation to content management, triage and approvals, feedback and improvements, monitoring & reporting, and user access and permissions.

# Types of artefacts

The table below provides a brief description of the various artefact types available for access or request through the artefact library. This list will be revisited and updated on regular intervals.

| **Type** | **Description/Comments** |
| --- | --- |
| Best Practice Guidance | Widely accepted guidelines or standards that represent the most efficient and effective way to accomplish a task or achieve a specific outcome. |
| Checklists | Lists of items or tasks to be completed, often used to ensure that all necessary steps in a process are followed. |
| Security Blueprints | High-level, strategic framework that guides the design, implementation and management of security measures within an organisation. It provides a high-level overview of security goals, principles, and architectures. |
| Security Designs | High-level considerations / decisions about how security will be integrated into the entire system. This includes the selection of appropriate security controls (like firewalls, encryption, and authentication systems), how these controls interact, and how they fit into the overall system architecture. |
| Security Documentation | Written materials related to the security measures, policies, procedures, and protocols of a system or organisation. |
| Security Patterns | Reusable solution to a common security problem within a specific context in software or system design. Security patterns are like design patterns, providing developers with tested solutions for specific security challenges they might face during software development. They are typically focused on a single aspect of security and are detailed. |
| Security Requirements | Criteria that a system must meet to ensure its protection from threats and vulnerabilities. |
| Software Code | Instructions and statements in a programming language that define the functionality and behaviour of a software application. |
| Templates | Pre-formatted documents or forms that provide a consistent structure for capturing and presenting information. |
| Threat Models | Representations of potential security threats and vulnerabilities, used to identify and prioritise risks to a system. |
| Use Cases | Scenarios in which a system or application is used, detailing the interactions between users and the system to achieve a particular goal. |

# 

# Key Roles

The following table contains a list of roles and indicative permissions involved in managing the artefact library process.

| **Role** | **Suggested Members** | **Description** | **GitHub permission** |
| --- | --- | --- | --- |
| Admin | CDDO SbD Team | Manages settings, assigns permissions to users, and manages branch protection rules. | Admin |
| Approver | CDDO SbD Team + Contributors | Approvers   1. Reviews incoming requests (issues) and provides approval to fulfil the request. 2. Reviews newly created artefacts to check whether these meet required standards ensuring the quality based on the design specification and provides approval for them to be published | Write |
| Collaborator | CDDO SbD Team | Commits changes directly, creates branches, pushes content and opens / merges pull requests. | Write |
| Contributor | Experts selected based on their knowledge and experience | Creates, edits, or submits content to the library by:   * Submitting pull requests with changes to the code or documentation. * Opening and managing issues to report bugs or suggest improvements. * Reviewing or commenting on existing pull requests or issues. | Write |
| Maintainer | CDDO SbD Product Manager | Maintain the repository and oversee the publications by setting the project direction and managing the contributors community. | Maintain |
| Owner | CDDO | Adds / removes members, manages settings and enforces policies. | Admin |
| Requester | Everyone | As the SbD artefact library is a public open source repository, by default, everyone will have view access to the repository and ability to raise new requests (issues) and comment on issues. | Read |
| Triager | CDDO SbD Team | Manages incoming requests, applies labels, closes tickets, and assigns work to collaborators and contributors. | Triage |

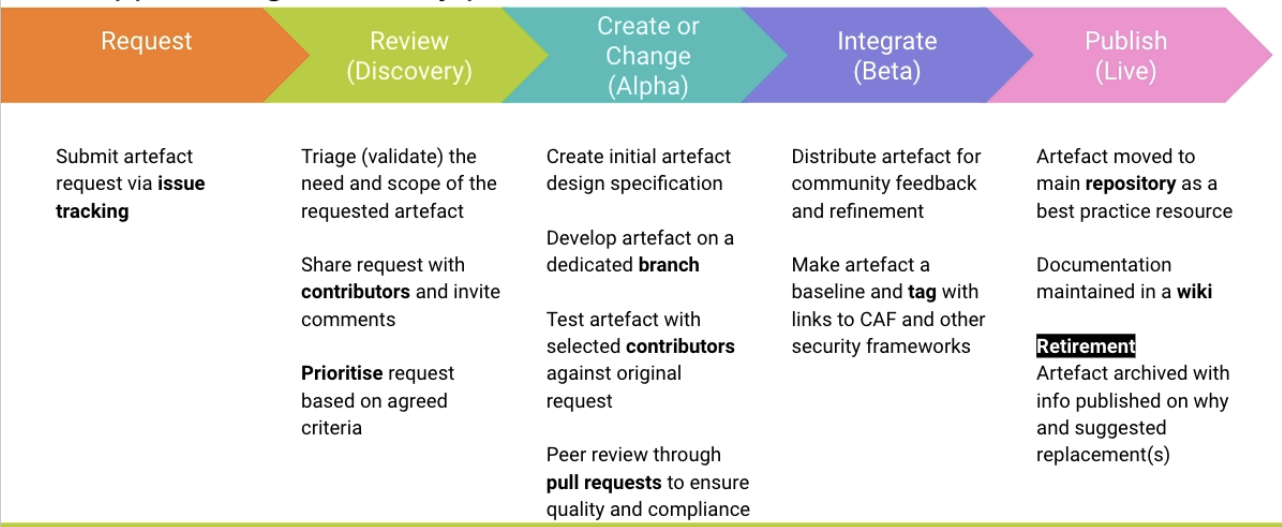
# 

# Artefact development process

The following diagram outlines the end-to-end process for the development of an artefact in GitHub, showcasing the various stages from request to publication. It visually represents the collaborative and agile nature of artefact development using GitHub’s version control system. Refer to Annex 1 for Github terms and Annex 2 for the detailed steps in Github.

## Principles

1. Short increments periods for contributors to avoid clashes



## Request

This sub-process outlines the steps for submitting a request to either create a new artefact or modify an existing one within the GitHub repository.

The process begins when a Requester identifies the need for a new artefact or a change to an existing artefact and creates a new 'issue' on GitHub. The Requester uses the appropriate template, specific to the type of artefact being requested, to ensure all necessary details are captured. This step ensures a standardised and efficient way to manage requests, enabling the repository maintainers and approvers to effectively review and prioritise the work.

## **Review**

This sub-process includes the steps for reviewing and approving new requests submitted via GitHub.

Once a request has been initiated and submitted as a GitHub issue, a notification is automatically sent to users that are ‘watching’ the SbD artefact library repository. This will include all SbD team members that have a responsibility of Collaborator in the library. The notification will trigger the review process and the issue will be assigned to a **Triager**. The Requester will receive an acknowledgment of their submission within 3 working days, and a final decision—whether the request is accepted or rejected—within 10 working days.

During the review, a **Triager** evaluates the request based on predefined criteria and procedures to determine whether the artefact should be created or changed.

The process may also involve tagging relevant **Contributors** for feedback and suggestions, ensuring a collaborative and comprehensive review before final approval.

Following the Triager's assessment, an **Approver** makes the final decision on whether to move forward with the creation or modification of the artefact.

## **Create (or change)**

This sub-process outlines the steps involved in the creation or modification of artefacts.

Once a request has been reviewed and approved, the **Triager** assigns the responsibility for developing or amending the artefact to a **Contributor**. The Contributors develop the artefact specification and are tasked with drafting or updating the artefact based on the specific requirements of the request.

Following the initial creation or amendment, all artefacts must undergo a peer review process to ensure quality, accuracy, and compliance with agreed document specification / standards. During this review, **Collaborators** facilitate peer feedback, ensuring that the artefact is refined and aligned with best practices.

## **Integrate** (Beta)

This sub-process outlines the steps required for integrating the artefact into the artefact library once it has passed through community peer review, testing, and final approval.

The first step is the formal announcement of the artefact to the community for review. The **Collaborator** makes the announcement. This provides stakeholders with visibility of the newly added artefact and invites further feedback.

Next, feedback is gathered from the community, which may involve additional refinement or updates to the artefact to ensure it continues to meet the needs of the users.

After incorporating any necessary adjustments, the artefact is tagged with appropriate metadata, such as platform-specific tags (e.g., AWS, Azure) or cybersecurity frameworks (e.g. CAF, NIST, ISO), allowing users to quickly find artefacts relevant to their requirements. The **Collaborator** plays a critical role in facilitating these tasks, ensuring that all relevant documentation and release notes are updated accordingly.

Finally, the artefact must be formally approved by the designated **Approver** before it can be published in the library and made available for use by the wider community.

## **Publish** (Live)

This sub-process outlines the steps required for publishing the artefact in the library, specifically into the main branch for broad access and use and an announcement is made to the community.

Throughout this process, the **Collaborator** ensures that the artefact is properly categorised, tagged with relevant metadata (e.g., related to specific platforms like AWS, security standards, or frameworks), and that its documentation is fully up-to-date. The **Maintainer** may also assist in ongoing management of the artefact, ensuring it remains available and that any future updates or revisions are promptly applied.

# 

# Annex

## Annex 1 - Github terms and definitions

| **Term** | **Definition** |
| --- | --- |
| Project | The whole artefact library is a project and there is a Project management tool that helps the team organise tasks, issues and pull requests (PRs) and organise work similar to tools like Jira and Asana. |
| Issue | Each "issue" represents a piece of work that can be tracked, discussed, and linked to commits or PRs. |
| Prioritise | Refers to the process of determining the order in which issues or PR should be addressed based on their importance or urgency. |
| Repository | Central space where the artefacts (documentation) and other resources are stored which tracks the history of all changes made to the files. |
| Branches | Collaborators create branches to work on different features without affecting the main project. Each branch is a parallel version of the main branch. Once individual branches are complete, they can be merged back into the main branch. |
| Pull Requests (PRs) | Feature that enables maintainers and collaborators to propose changes to the main project. Other collaborators can review the changes, discuss, and approve them before merging into the main branch. |
| Commits | Each time a collaborator makes a change, they "commit" their changes to the repository. Each commit has a message describing the changes and a unique ID, making it easy to track and review updates. |
| Merge and Conflicts | When changes are ready to be integrated into the main branch, GitHub merges the code. If there are overlapping edits, a conflict may occur, which developers need to resolve manually. |
| Wiki | Place where project documentation can be easily created, organised, and maintained. It provides a collaborative space for teams to write content. Changes to the wiki are tracked, enabling users to revert to earlier versions if needed. It allows content to be broken down into categories, sections, and pages, making it easy to navigate and find information. |
| Retirement | Repository or project or branch is marked as deprecated, meaning no new updates or support will be provided, but it remains available for historic references. |
| Watching | Feature that allows users to receive notifications about the activity in a repository. When you choose to "watch" a repository, GitHub sends you updates via email or in your GitHub notifications about key events like new issues, pull requests, and discussions. |

## Annex 2 - Detailed artefact development process

The following flowchart shows the key steps of the artefact creation process.



## 

## Annex 3 - Github Instructions

### Request a new artefact or update to an existing artefact

The process of creating a new artefact in the SbD artefact library, or updating an existing one starts with a request from a user. The request will be made through the GitHub repository.

1. **Start**:
   * The requester initiates the process of requesting a new artefact, or an update to an existing artefact.
2. **Requester Navigates to Repository:**
   * The requester goes to the [Secure by Design GitHub repository](https://github.com/co-cddo/SbD).
3. **Requester Clicks on "Issues" Tab:**
   * The requester selects the "Issues" tab to open a new issue.
4. **Requester Clicks on "New Issue":**
   * The requester initiates the creation of a new issue.
5. **Requester Selects "Artefact Request" Template:**
   * The requester chooses the predefined "Artefact Request" issue template.
6. **Requester Fills in Template Details:**
   * The requester completes the template, providing all necessary information about the artefact request.
7. **Requester Clicks "Submit New Issue":**
   * The requester submits the issue by clicking the "Submit new issue" button.
8. **Issue Created and Notification Sent to Reviewers:**
   * The issue is created in the repository, and reviewers are notified of the new request.
9. **End**:
   * The request process is complete.

### Review the new request

After the request has been initiated and submitted, a notification is sent to the SbD team for review. The request is reviewed and a decision is made as to whether to proceed with the creation of the artefact, or not.

1. **Start:**
   * The review and initial approval process begins.
2. **SbD Reviewers Receive Notification of New Issue:**
   * Reviewers are notified of the new artefact request issue.
3. **SbD Reviewers Open and Read Issue Details:**
   * Reviewers open the issue and read the provided details.
4. **SbD Reviewers Discuss Validity and Scope in Issue Comments:**
   * Reviewers discuss the validity and scope of the request using issue comments.
5. **Request More Info:**
   * Reviewers may request additional information from the requester if needed.
6. **Requester Provides More Info:**
   * The requester provides the requested additional information.
7. **Issue Validated?:**
   * Decision point to determine if the issue is validated.
   * If no, the request is declined, and the process is terminated.
   * If yes, relevant contributors will be tagged for feedback and input.
8. **Tag Contributors for Feedback:**
   * Reviewers tag relevant contributors to provide input and feedback on the issue.
9. **Collect Feedback and Comments:**
   * The request is given a prioritisation and added to the queue.
   * The request is assigned to a group for creation.

### Create Artefact

The creation or amendment of SbD artefacts will be assigned to security architects or stakeholders that have volunteered to assist in this area. All newly created or modified artefacts will go through peer review and testing before they are approved for deployment.

**1. Start process for the creation of the Artefact**

1. **Create a Dedicated Branch**
   * Navigate to the GitHub repository.
   * Create a new branch named after the artefact or issue number (e.g., artefact-123).
2. **Create the Artefact**
   * Create the artefact according to the request.
   * Commit changes to the dedicated branch as required.
3. **Open a Pull Request (PR)**
   * Once the artefact has been created, open a PR to merge the dedicated branch into the main branch.
     + Go to the repository on GitHub.
     + Click on "Pull Requests."
     + Click on "New Pull Request."
     + Select the dedicated branch and the main branch.
     + Provide a detailed description of the changes and link to the issue.
     + Submit the PR for review.

**2. Peer Review of the Artefact**

1. **Initial Peer Review**
   * Notify reviewers by tagging them in the PR description or comments.
   * Reviewers conduct a review, providing feedback and suggestions for improvement.
   * Address any feedback by making necessary changes and updating the PR.
   * Reviewers approve the PR if it meets quality and the request specification.
2. **Merge Changes**
   * Once approved, the PR is merged into the dedicated branch (not the main branch yet).
     + Click "Merge pull request."
     + Delete the branch after merging if desired.

**3. Testing the Artefact with Selected Users**

1. **Select Test Users**
   * Identify and notify selected contributors to test the artefact.
2. **Testing Process**
   * Provide test users with instructions and access to the dedicated branch.
   * Collect feedback on the functionality, usability, and any issues encountered during testing.
     + Create issues for any bugs or improvements needed based on feedback.
3. **Address Feedback**
   * Implement changes based on test user feedback.
   * Update the PR with the changes and request another review if necessary.

**4. Final Peer Review to Ensure Quality and Compliance**

1. **Request Final Peer Review**
   * Notify reviewers for a final peer review of the artefact.
2. **Final Peer Review Process**
   * Reviewers perform a thorough review, ensuring all feedback has been addressed and the artefact meets all quality and compliance standards.
   * Address any additional feedback or changes requested by reviewers.
3. **Approval and Merge to Main Branch**
   * Once the final review is approved, merge the dedicated branch into the main branch.
     + Click "Merge pull request."
     + Ensure all automated tests pass before merging.
4. **Documentation and Close Issue**
   * Update documentation related to the artefact in the repository's wiki.
   * Close the original issue related to the artefact request.

### Integrate Artefact

Artefacts that have gone through peer review, testing and approval will be integrated into the library.

1. **Announce the Artefact for Community Review**
   * Create a discussion thread in the repository to announce the artefact is ready for community feedback.
   * Tag relevant community contributors and interested parties to notify them of the feedback request.
2. **Collect Feedback and Refine the Artefact**
   * Monitor the issue or discussion thread for feedback from the community.
   * Create issues for any bugs, suggestions, or improvements reported by the community.
   * Address feedback by making necessary changes to the artefact in a new branch or the existing dedicated branch.
   * Once the feedback has been addressed, update the pull request or create a new one for final review.
3. **Final Review After Refinement**
   * Request a final review from reviewers.

**2. Make Artefact a Baseline and Tag with Links to Security Frameworks**

1. **Establish the Artefact as a Baseline**
   * Once the final review is approved, merge the dedicated branch into the main branch.
     + Ensure all automated tests pass before merging.
     + Use a meaningful commit message to indicate the artefact is now a baseline.
2. **Tag the Artefact with Relevant Frameworks**
   * Create a new tag for the artefact in the repository.
     + Name the tag meaningfully (e.g., v1.0.0-artifact-123).
3. **Link to Security Frameworks**
   * Update the repository’s documentation (e.g., README, Wiki) to include links to relevant security frameworks such as NIST.
     + Add a section describing the artefact’s alignment with NIST and other frameworks.
4. **Update Release Notes**
   * Create a release in the repository to officially document the baseline artefact.
     + Navigate to the "Releases" tab.
     + Click on "Draft a new release."
     + Select the newly created tag (e.g., v1.0.0-artifact-123).
     + Provide release notes that include a summary of the artefact, its purpose, and links to the security frameworks.
5. **Communicate the Baseline Establishment**
   * Announce the baseline establishment to the community.

### Publish Artefact

After a successful initial integration, the artefact is published and an announcement is made to the community.

**1. Prepare the Artefact for Final Integration**

1. **Review the Final Artefact**
   * Ensure all stakeholder and collaborator feedback has been reviewed and addressed appropriately and the artefact has been updated and approved.
   * Conduct a final review to confirm the artefact meets all quality and compliance standards.
2. **Finalise Documentation**
   * Update the artefact’s documentation in the repository’s Wiki or documentation directory.
     + Include detailed usage instructions, examples, and any necessary configuration information.
     + Provide links to relevant security frameworks (e.g., NIST) and baseline information.

**2. Move the Artefact to the Main Repository**

1. **Merge the Artefact Branch**
   * Ensure all changes are committed and pushed to the dedicated branch.
   * Merge the dedicated branch into the main branch.
     + Provide a detailed description of the artefact and the changes made.
     + Link to the issue that tracks the artefact’s development and integration..
     + Ensure all tests pass before merging.

**3. Create and Document the Release**

1. **Tag the Release**
   * Create a new tag for the artefact in the main branch.
     + Use a meaningful tag name (e.g., v1.0.0-artefact-123).
2. **Create a Release**
   * Navigate to the "Releases" tab in the GitHub repository.
   * Click on "Draft a new release."
   * Select the newly created tag (e.g., v1.0.0-artefact-123).
   * Provide release notes that include a summary of the artefact, its purpose, and links to relevant security frameworks.
3. **Update the Wiki and Documentation**
   * Ensure the artefact’s documentation in the Wiki is up-to-date.
   * Include links to the release, tag, and any relevant sections of the documentation.

**4. Communicate the Release**

1. **Announce the Release**
   * Create an issue or discussion thread to announce the release of the artefact as a good practice resource.
     + Tag relevant stakeholders and contributors.
2. **Notify Relevant Teams and Contributors**
   * Send notifications about the new release.