NRA maintenance procedures

This document outlines feedback and maintenance procedures of the NAP reference architecture (NRA).

**History**

|  |  |  |
| --- | --- | --- |
| Version | Who | Note |
| 0.1 | Petr Bureš | First draft based on NRA maintainers internal discussion |
| 0.2 | PB+BW | Revision of the document |
| 0.3 | WG2.3 | Internal revision of the document. |
| 20241009 | WG2.3 | Resolution at the meeting |
| 0.5 | Petr Bureš | Revised and harmonised with MS Word template |
| 0.7 | BW+SL+PB | Revision of the comments |

# Actors and responsibilities

## Maintainers

Typically, a smaller group (5-10 members) is ideal. Reason: This ensures focused and efficient handling of core updates and critical issues. A smaller group can communicate more effectively and make quicker decisions.

### Responsibilities:

* Core Updates: Implement and test major updates and changes.
* Issue Resolution: Address critical issues and bugs.
* Documentation: Maintain detailed documentation for all changes and updates.

### Involvement:

* Regular Meetings: Weekly or bi-weekly meetings to discuss progress and upcoming tasks.
* Direct Communication: Maintain open channels for immediate issue resolution.

## Commenters and Supporters

Medium-sized group (20-50 members). Reason: This group provides a broad range of feedback and support without becoming too unwieldy. They can cover various aspects of testing and troubleshooting, offering diverse perspectives.

### Responsibilities:

* Feedback: Provide constructive feedback on updates and changes.
* Testing: Participate in beta testing to identify potential issues.
* Support: Assist in troubleshooting and provide support to the community.

### Involvement:

* Feedback Sessions: Monthly sessions to gather feedback and suggestions.
* Beta Testing: Organized testing phases before major releases.

## Community

Larger group (100+ members). Reason: The community should be large enough to provide a wide array of suggestions and report issues from different user experiences. A larger community ensures that updates and changes are well-rounded and cater to a broader audience.

### Responsibilities:

* Engagement: Stay informed about updates and changes.
* Reporting: Report issues and bugs encountered.
* Suggestions: Offer suggestions for improvements and new features.

### Involvement:

* Surveys and Polls: Regular surveys to gather community input.
* Forums and Discussions: Active participation in forums and discussion boards.

# General aspects of Maintenance

## Workflow Integration

1. Issue Tracking: Use a centralized issue tracking system where all members can report and track issues.
2. Version Control: Implement a version control system to manage updates and changes.
3. Communication Platforms: Utilize platforms like Slack, Discord, or dedicated forums for seamless communication.
4. Documentation: Ensure all documentation is accessible to everyone, with clear guidelines on how to contribute.

## Regular Updates and Transparency

* Update Announcements: Regularly announce updates and changes to keep everyone informed.
* Transparency Reports: Publish reports on the progress and status of ongoing tasks and issues.

## Training and Onboarding

* Training Sessions: Conduct training sessions for new maintainers and active community members.
* Onboarding Guides: Provide comprehensive guides for new members to understand their roles and responsibilities.

# Workflow Integration

## Issue Tracking

**Purpose:** To centralize the reporting and tracking of issues, bugs, and feature requests.

**Tools**: Use platforms like Jira, GitHub Issues, or Trello.

**Process**:

* Reporting: Community members report issues through the chosen platform.
* Prioritization: Maintainers prioritize issues based on severity and impact.
* Resolution: Issues are assigned to maintainers for resolution, with progress tracked and updates provided.

**Benefits:**

* Transparency: Everyone can see the status of reported issues.
* Efficiency: Streamlines the process of identifying and resolving issues.

## Version Control

**Purpose**: To manage and track changes to the codebase or project files.

**Tools**: Use Git-based platforms GitHub, GitLab, or Bitbucket..

**Process**:

* Branching: Create branches for different features or fixes.
* Pull Requests: Commenters and supporters review and test changes before they are merged.
* Merging: Maintainers merge approved changes into the main branch.

**Benefits**:

* Collaboration: Facilitates collaborative development and code review.
* History: Maintains a history of changes, making it easier to revert if needed.

## Communication Platforms

**Purpose**: To ensure seamless communication among all groups.

**Tools**: Use platforms like Slack, Discord, or dedicated forums.

**Process**:

* Channels: Create specific channels for different topics (e.g., updates, issues, general discussion).
* Announcements: Regularly post updates and announcements.
* Discussion: Encourage active participation and discussion among community members.

**Benefits**:

* Engagement: Keeps everyone informed and engaged.
* Collaboration: Facilitates real-time collaboration and support.

## Documentation

**Purpose**: To provide clear and accessible guidelines and information.

**Tools**: Use platforms like Confluence, GitHub Wiki, or Notion.

**Process**:

* Creation: Maintainers create and update documentation for all changes and updates.
* Access: Ensure documentation is easily accessible to all community members.
* Feedback: Allow community members to suggest improvements to documentation.

**Benefits**:

* Clarity: Provides clear guidelines and instructions.
* Support: Helps community members understand how to contribute and resolve issues.

# Versioning

For new releases of the NRA the following scheme will be used:

* New versions will be released periodically (e.g. quarterly) fixing several issues, improvements and bugs at once, i.e. not every issue resolution will trigger new version release.
* The versioning scheme will follow semantic versioning ([Semantic Versioning)](https://semver.org/)
  + Given a version number MAJOR.MINOR.PATCH, increment the:
    - MAJOR version when you make incompatible API changes
    - MINOR version when you add functionality in a backward compatible manner
    - PATCH version when you make backward compatible bug fixes
  + Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format. (alpha [a], beta [b], release candidate [rc])
* For more information see: <https://semver.org/>

The versioning scheme applies to architectural views (i.e., motivational layer, user needs, functional view, physical view, organizational view, etc.) and to a file itself. Only views that changed will have the version increased. Version on the level of architectural objects is date of change of the object (e.g date modified)

# Communication of releases

How do stakeholders know there has been a major release (mailling list?). Options:

* Follow GitHub repository
* LinkedIn Channel NAPCORE: News entry
* E-Mail NAPCORE mailing list (full list)

# Workflow for Handling Issues

* Reporting: Issues are reported through the issue tracking system.
* Categorization: Issues are categorized as General, Editorial, or Technical.
* Assignment: Issues are assigned to the appropriate team (maintainers, editors, or community managers).
* Resolution: The assigned team works on resolving the issue.
* Feedback: The resolution is communicated back to the reporter, and feedback is gathered if necessary.

## Reporting and categorization

Based on the community feedback, internal update processes, model issues handling create issue on GitHub (<https://github.com/NAPCORE/NAP-Reference-Architecture/issues>).

Template of the issue shall address (in reference to MS file issue):

* **Issue name:** composed of the problem description and scope identification
  + **[View / Object Type / Object Name]** – as inserted by the user, see description above
  + **Short description**
* **Label assigned by user:** 
  + **Issue type** (General/ Editorial / Technical)
    - General: These are broad issues that don't fall into specific categories like editorial or technical. They often relate to overall user experience, community guidelines, or general feedback
    - Editorial: These issues pertain to content accuracy, clarity, and presentation. They often involve written content, documentation, or any editorial material
    - Technical: These issues relate to the functionality and performance of the platform or service. They often involve bugs, glitches, or technical malfunctions.
  + **Scope** (bug, enhancement, question)
* **Contents of the issue:**
  + **Comments –** initial comments from the filing user
  + **Proposed Change** **–** initial proposal from the filing user.
* Content of the issue follow-up
  + **Observation**: a sequence of comments to the issue ending with a resolution.

## Assignment

1. If the issue is received as file, NRA maintainers create a GitHub issue and attach the submitted file with comments to it.
2. NRA maintainers review the issue and
   1. assign / reassign labels
      1. **Scope**: bug, enhancement, question
      2. **Severity**: minor, major
      3. **Status**: needs review, in review, reviewed, in processing, done
      4. **Milestone**: version in which it will be implemented
   2. Assign resolution team
   3. assess the impact: whether the issue potential resolution triggers a substantial change on MS implement or operate their NAP. If a substantial change is foreseen, the resolution may need to involve a SCOM decision.

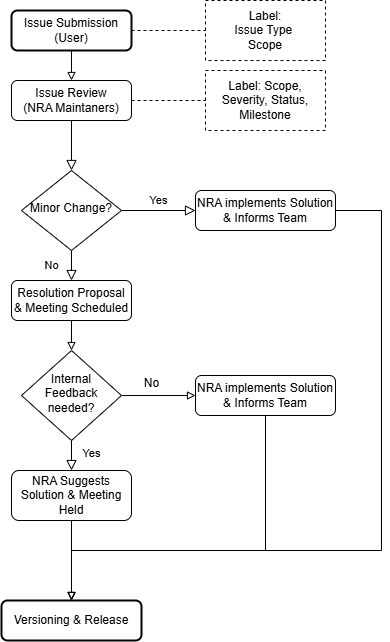
## Resolution and feedback

In case of **minor** - nonbreaking change:

1. NRA maintainers propose and implement a feedback resolution.
2. NRA maintainers inform NRA team about the implemented resolution (or include it in a change log).

In case of **major** – breaking change:

1. NRA maintainers identify a (preliminary) resolution proposal.
2. NRA maintainers schedule a resolution meeting involving the NRA (+ NLKF) team and the commenting party.
3. Workshops/seminars introducing the issue for a very serious change.
4. During the resolution meeting, a decision is made on to address the review feedback.
5. The resolution is implemented either during the meeting or later based on the taken decision.



## Reaction times, issue resolution timeframe

The timeframe is subject to regular period of the NRA maintenance meetings.

For any issue

* Step 1 –2: **one month**, ideally during monthly NRA maintainers meeting

For a minor - nonbreaking change:

* Step 3: **two months**
* Step 4: **one week**

For a major – breaking change:

* Step 3-4: **two months**
* Step 5: **one week**
* Step 6: **two weeks to two months** to implement resolution of major issue based on its complexity.

## Reporting and processing of Issue within EA Model

During the NRA development some issues, comments or reminders are generated on the fly, to describe them as regular issue could be too time demanding. They are described by the issue object in the NRA model and linked to the object they comment on. The process is as follows:

Identification and creation of the issue:

* Create issue object in EA and describe the problem, assign urgency and severity.
* Link created issue object to the object in the view which they describe.
* Move issue to supplements/issue folder for later resolution

Handling the issue:

* Regularly revisit the issues, directly resolve small issues and delete them
* Rewrite complex issues into a template, crate a GitHub issue mirroring this one and link the GitHub issue inside, use GitHub for tracking and resolving the issue, when resolved delete the issue.

# Procedures used during the NAPCORE

Here is feedback procedures used during the project NAPCORE runtime.

## NAPCORE Feedback process

This feedback is intended when an approval and possible intervention / collaboration from another working group is needed. For example, a decision on the standards to be applied for interfaces or processes to be realized by NAP, etc.

The procedure is as follows:

1. The NRA team identifies an issue where approval from a specific WG is needed.
2. NRA maintainers propose a resolution of the identified problem AND the responsible WG.
3. The NRA team sets up a meeting with the responsible WG, where the issue is presented and confirmed/rejected.
4. Based on the decision of the responsible WG, the NRA maintainers process the change to the NRA.

The NAPCORE feedback serves to involve responsible WGs in the important NRA aspects. The actions are needed, for example, for:

* **[WG4.4] Metadata Standards**: Resolving what parts and how it shall be referenced within the NRA.
* **[WG2.2] Data Sets / Service Profiles**: Resolving which profiles are recommended for delegated acts.
* **[WG2.4] Demonstrators**: any geed practice or experience.

## Demonstrator Feedback template

This type of feedback is reserved for demonstrators to bring **back any good practice/information** about what platform has been used, what approach has been taken, how it was organized, etc.

The feedback is organized into sections with specific information:

* **Licenses**: What license (for data/service usage and NAP usage (metadata)) types have been necessary to use in the demonstrator and if there is a lesson learned (e.g., what type is most useful or what licenses are needed for what part of the practice and how they were collected, etc.).
* **Interfaces**: What interfaces have been used with what software solution and what transport protocol, and if there is a lesson learned (e.g., what was found to be useful, most accepted, what software solutions are preferred, what security was deemed sufficient, what procedures, etc.).
* **Specification**: What data/service/metadata specification (standard or protocol specification, profile) has been used and with what outcome, and if there is a lesson learned (e.g., what specification for what data type, how was the implementation realized, shared code, tags used).
* **Functionality**: What functionality is needed at the NAP to make the demonstrator work and what functionality of the demonstrator is seen as beneficial to be adopted by NAPs.
* **Processes**: What processes realized by the demonstrator are relevant for the NAP and could be adopted in full or partially by the NAP. For example, registration of users, and how specifically (compare with current NRA)
* **Quality Parameters/Requirements**: Did the demonstrator yield any specific information about quality parameters of the NAP, or if there are any recommendations on the quality of the NRA service.
* **Additional Info**: Any other good practice that did not fit into previous categories but is beneficial for NAP.

The intention is to identify good practices, resolutions, software code that could be reused by other NAPs.