Version: 0.2

MCC MCP environments, issue handling and release management

1 SUMMARY

This document describes two things

- The MCP environments operated by partners of the MCC on behalf of the MCC for their purposes.
- The release management procedure for the MCP reference software as developed by partners of the MCC on behalf of the MCC and how the above environments are used in relation to that.

The official MCP reference software is located in GitHub at this location:

github.com/MaritimeConnectivityPlatform

2 MCC MCP ENVIRONMENTS

The MCC (or rather partners of the MCC on behalf of the MCC) will operate 3 environments of the MCP.

- A test environment which will be used for testing new versions of the MCP reference software as developed by the MCC (or rather partners of the MCC on behalf of the MCC).
- A staging environment which will be used for testing new release candidates of the MCP reference software.
- A public test environment for use by anyone for general tests/assessment of the MCP and services. This is mainly for promotional purposes.

2.1 Access to environments, and validation of users

Only MMC members will be granted access to the test and staging environments, and these will only be used for the purposes designated for them.

The pubic test environment will be, as the name indicates, made available to all (relevant) external stakeholders. Users of the public test environment will go through a very simple validation process.

- New users apply for access through the management portal of the public test environment
- Their organisations will be assessed though their website
- The users will be validated through email correspondence where they are requested to use an email address belonging to the official internet domain
- Access to the environments are handled by the MCC secretariat, which may deviate from the above procedure if they deem appropriate

3 RELEASE IDENTIFICATION

There will be major releases, X.0.0, minor release, X.Y.0 and patches, X.Y.Z. X and Y will always be the same for all elements in GitHub. An individual element may be patched independently of the other elements. The versioning numbers will be given as labels in GitHub.

Each element identifies its own version number, and the management portal lists version numbers of all elements.

3.1 Release units

The release units covers all official MCP elements in GitHub, which are:

IdentityRegistry;	MIR - Maritime Identity Registry
MCPKeycloakSpi;	Part of MIR
MCP-PKI;	Part of MIR
mc-serviceregistry;	MSR - Maritime Service Registry
MC_MMS;	Maritime Messaging Service
APIGateway;	Component for the MMS
MNS;	Maritime Naming Service – component of MMS
MCP-Portal;	Maritime Connectivity Platform Management Portal
EndorsementService;	Endorsement Service

4 SOFTWARE ISSUE (BUGS/FEATURES) HANDLING

All issues are handled using Trello boards. Two types of boards will be used:

Backlog: Contains all issues which have not yet been assigned to a specific version. This Trello board is located here: https://trello.com/b/S5APwfnI/backlog.

Version X.Y.Z: Each version will have a Trello board for managing the release of that specific version. A template for these boards is located here: https://trello.com/b/F8PDohFj/release-plan-template

These Trello bards are only accessible by members of the consortium.

New issues (features, changes, bugs) may come from a number of different sources:

From working groups in the consortium

From the public test environment. Either reported by users through the portal or generated automatically from the running instance

From official (approved by the MCC) operators of MCP instances

All issues will be handled by the MCC testbed group, but any issue relating to any other specific group (MIR, MSR, MMS) must be handled in accordance with their decisions if relevant.

All new issues will be entered into the MCC Trello 'backlog' board list 'Incoming Unqualified Issues'.

At the testbed group meetings, the following process is followed:

All items in the 'Incoming Unqualified Issues' are discussed, and ideally, they should be removed from this list during the meeting. Each item is either:

Moved to the 'To delete list', if they are irrelevant, solved, duplicates etc

Moved to 'Qualified Unprioritized Issues' list. When in this list, the description of the issue must have been assessed and altered if needed.

In the second step the issues in the 'Qualified Unprioritized Issues' list is handled. These may:

Stay in the current list, if prioritisation is wished to be postponed.

May be moved to one of the prioritized lists (low, medium, high).

Third step is to go through the prioritized issues. These may either:

Stay in the prioritized list

Be moved to the 'Release plan' Board assigned to a 'Target Issue' list for a specific release version. When moved to the 'Release plan' board, all issued must be assigned a responsible person from the working group

When issues are assigned to a target issue list, then the following aspects must be handled

Do the test cases need to be updated? If so, this work needs to be done together with the implementation.

Is there any impact on any existing documentation? If so, the update of that also needs to be done with the implementation. This would need to be coordinated with other relevant working groups (MIR, MSR, MMS). No changes must be released without updated documentation.

When an issue assigned to a target issue list (a specific version) has been resolved and the fix/feature have been submitted to GitHub, the issue is moved to the ' Issues Fixed, submitted to GitHub' list.

If all issues have been removed from the 'Target Issues' list, the MCDT initiates the release process as described below.

The group may also choose to move an issue back to the 'Backlog' board if release timing constraints require it.

4.1 Release process

When all issues related to a specific target release has been resolved and submitted to GitHub, the release process kicks in.

First the code is frozen by applying a label in GitHub. And the checkbox in the 'Release Procedure' card is checked.

Second, the code is deployed to the staging servers and the 'Deployed to staging' is checked.

Third all the Test cases described in the 'Test cases' list are conducted (on the staging instance obviously). Passed test cases are moved to the 'Test cases succeeded' list. Failed tests are moved back to the 'Target Issues' list with a comment describing the nature of the failure.

When all tests are conducted then, then either situation applies:

1. One or more tests failed, and thus there are one or more issues in the 'Target Issues' list. In this case, all (or some; up the discretion of the working group) of the Test cases are moved back to the 'Test cases' list, and the release process is reset by de-selecting the flags 'Code Freeze, beta tag' and 'Deploy to staging'.
2. Or all tests succeeded and thus both the 'Target issues' and the 'Test cases' lists are empty. Then the release process continues as follows:

The flag 'Test successfully completed' is checked.

A final release label is put on the source code in GitHub and the corresponding checkbox is marked.

Release note (including downtime in relation to deployment) are written and send to working group members and posted to the google groups, and the related checkbox is marked.