Guidelines for external contributions for the Eclipse mdmbl project

openMDM(R) Eclipse Working Group

Document history:

Author	Date	Version	Description
Angelika Wittek	01.05.2018	0.1	initial version
Angelika Wittek	02.05.2018	0.2	Review comments from S. Ebeling added
Angelika Wittek	02.05.2018	0.3	Review comments from S. Wartini added
Angelika Wittek	02.05.2018	1.0	Initial official version
Angelika Wittek	27.08.2018	1.1	Approach 1 added, review comments from S. Ebeling added

Table of contents

Note: this document is written in Google Docs, location:
https://docs.google.com/document/d/1cFiYue7bYxLp5-rXHNj79xQWDne16mOESYNy9r5bQ/edit?usp=sharing

1 Introduction

In this document we describe two approaches how external contributions / development can be integrated into the openMDM application.

The openMDM Working Group strongly recommends to follow the first approach. The second approach was defined for existing contributions, started before the first approach was defined.

First approach (recommended):
The external development is integrated to the openMDM development from the beginning on. Starting with the coordination of the new features in the committees, integrated milestone planning with the product manager and development in the Eclipse Infrastructure. This approach is described in this chapter.

Second approach (not recommended):
The last months have shown that there are several code extensions based on the Eclipse mdmbl code that should to be contributed to the mdmbl codebase. The development is done outside the EWG and the Eclipse Infrastructure. The code is contributed as once, when the external development is finished. Here we describe a way how these contributions can be contributed to the codebase.

For every contribution these guidelines have to be followed, in every case the AC/QC and the openMDM(R) product manager (PM) will support the contributors and specify and/or modify the guidelines if appropriate.

In the openMDM(R) EWG the development of external contributions are normally assigned from a customer to a contractor (in this case the contributor), we define these two stakeholder here for the following steps.

Please note: Modifying the mdmbl code and transferring to other parties breaks the EPL. The code has to be put under the EPL and has to be available for the public.

2 Approach 1:
External contribution - the integrated way

For a successful joined work we will first define the stakeholder involved, defining the roles. Then defining the responsibilities and the process.

2.1 Stakeholder and Roles

Stakeholer

1. The customer
2. The contractor (in this case the contributor)
3. The openMDM(R) EWG

Roles:

1. The Product Owner (PO) assigned to a dedicated person from the customer
2. The Key Developer (KD) assigned to a dedicated person from the contributor
3. The Product Manager (PM) from the openMDM(R) EWG

During the whole lifecycle of this development these three persons (contribution project committee - CPC) have regular exchange and are responsible to inform the committees.

2.2 Life cycle of a contribution project

A contribution project starts with the announcement of the PO to the Steering committee. It finishes with the successful integration of the code into the platform and is part of a milestone release.

2.3 Steps for the contribution project

The following steps have to be processed:

1. The PO announces a new contribution project and presents the features in the SC and the AC (“one pager presentation”)
    a. The SC can reject it by voting against with a powerful decline comment
    b. The AC/QC will provide support, if requested by the SC
2. The SC / AC has to check the features, e.g.:
    a. Are there overlappings with other requirements?
    b. Are the changes applicable for all sorts of ASAM ODS compatible data sources?
3. The KD will provide the user stories in the openMDM(R) JIRA REQU-Project
4. The PM is responsible for the milestone release planning:

    a. The CPC is defining the user stories that have to go together into a milestone release (user story package).
    b. The PM will assign the user stories in JIRA to the planned milestones
5. The PM will provide Eclipse Git Repository Branches for the development
6. The PM creates for every user story at least one Bugzilla issue
7. The KD has to announce and get the approval for API changes from the AC for every user story, before the development starts.
    a. The KD adds the approved API changes to the dedicated Bugzilla issue
8. The contributor seeks for setting up own committer, if he has no. See here. As long as there is no:
    a. The PM is responsible that the code reviews are done (2-3 working days)
    b. The PM supports with all issues only committer can do.
9. The contributor is developing in the Eclipse Infrastructure:
    a. Continuous development in the Eclipse Git Repositories
    b. Setting up continuous builds and static code analysis (Jenkins and Sonar)
    c. Using the Eclipse Code Review System (Gerrit)
    d. Reporting and tracking Bugs in the Eclipse Bugzilla Issue Tracker
    e. Following the Eclipse Development Process
    f. For every milestone release: merges from the master branches to the contribution branches
    g. Active communication via the dev mailing list
10. The contributor is following the EWG guidelines:
    a. Quality rules (to be defined from AC/QC)
    b. JUnit tests for new code have to be provided (test coverage > 70%), for code extensions Junit test have to be added / extended
    c. The approval for the usage of new 3rd party libraries have to be requested from the AC before usage, see also 11.b
11. The contributor is following the Eclipse Legal Process, see also (Introduction to the Eclipse IP management for openMDM)
    a. Providing legal documentation (license/copyright header, notice files)
    b. Executing IP checks for 3rd party libraries (new ones and version changes) before usage. The PM will support.
12. During development from the contactor:
    a. For new detected, existing Bugs a new Bugzilla Bug has to be created. The Bug should be fixed, if possible. If there are any reasons against, this issue has to be discussed with the PM.
13. The contributor is providing documentation:
    a. Core changes, implementation details and important information about a user story has to be added to the dedicated Bugzilla Issue
    b. For Bugs that are fixed in the contribution: the Bugzilla Bugs has to be referenced from the dedicated Bugzilla Bug
14. The development of the user story package is finished by the contributor:
    a. The PM is responsible to review and merge the code to the dev branches, according to the milestone release plan
    b. If any issues come up, the PM will request changes

    c. The KD is responsible to fix the requested changes from PM following the release plan.
15. The PM is responsible for the milestone release:
    a. Updating the technical documentation and zhe Eclipse mdmbl project page
    b. After the release, the contribution branches for this user story package will be deleted.

2.4 About Contributor and Committer in the Eclipse Ecosystem

- Everybody who signs an ECA can contribute to the code review system
- Only committer have write access to the Eclipse Git Repositories
- To become a committer he has to be nominated by an existing active committer and elected.
- There are only three requirements around nominating and electing:
    - Define Trust
    - Employment Neutral
    - Public and Archival Election
- Committer have to sign and to follow the Eclipse Committer Due Diligence Guidelines

From the Eclipse website:
(...) Becoming a committer is a privilege that is earned by contributing and showing discipline and good judgment. It is a responsibility that should be neither given nor taken lightly, nor is it a right based on employment by an Eclipse member company or any company employing existing committers.

It is the clear goal that every contributor should have its own committer.

Is is also a clear decision from the EWG, that if a committer is not following the above processes, he will lose his committer status.

3 Approach 2:
External contributions - block contribution

This approach is deprecated. It was defined because of existing situations before approach 1 was defined. This approach brings the risk that the whole contribution is rejected by the EWG.

The decision about an external contribution becoming a part of the mdmbl codebase consists of two parts. First is the process - the openMDM(R) EWG has to decide if the contribution contains extensions / features that are valuable for the codebase and have to be added to the mdmbl project. The second part is the technical part and consists of the code quality checks and the way of merging it into the codebase.

3.1 Organizational Steps

1. The contributor and/or the customer provide a description of the contribution (business aspects and technical overview including API changes)
2. The AC validates the technical and functional content and provides an actionable proposal to the SC.
    a. AC needs to take special care if the new functionality works for existing subsystems and components as well as known products (Example: PAK Adapter).
    b. The AC shall approve such contributions unless severe reasons speak against approval.
3. The SC approves or rejects the task for contribution, in which case the next step will not be started.
4. In case of approval, SC advises the Product Manager (PM) to continue with the technical steps listed below and approves cost / expenses for the PM and the development team.

3.2 Technical steps

1. The PM, the AC/QC and the contractor clarify the contribution rules (IP management, test coverage, code style, code comments, Key Performance Indicator (KPI) of static code analysis, ...).
2. The contributor checks the code in accordance to the rules and applies changes as needed.
3. The contributor provides information and documentation for the contribution (e.g.API Changes, text for the release notes, ...).
4. The contributor merges a defined code version into their contribution. The PM defines the version of the mdmbl code.
5. Test cases of the mdmbl code as well as the contributed code shall execute without errors.

6. The dev team creates a new branch for the contribution in the Eclipse Infrastructure (“new_branch”).
7. The contributor contributes the code to the “new_branch” via Gerrit - no committer rights are necessary, only a valid Contributor License Agreement (CLA) is required:
    1. The dev team reviews the code.
    2. If there are review comments, the contributor applies changes as needed.
8. If the contribution contains more than 1000 lines of new/changed code an IP Check from Eclipse is necessary and will be started and monitored by the PM.
    1. If IP issues come up the contributor applies changes as needed.
9. All checks complete: a committer from the dev team approves the contribution in Gerrit. This causes the push of the contribution to the “new_branch”.
10. The PM and dev team merge the new branch into the main mdmbl code base at their discretion.
11. Final step: The contribution will be included in the next version of the mdmbl code.

3.3 Rejection of an external contribution

If the request for an external contribution to the mdmbl project codebase is rejected, the contributor and the customer are responsible on their own to follow the EPL. E.g. the extended / modified mdmbl code has to be licensed under the EPL and made available to the open.
For all licensing issues follow the EPL 1.0:
https://www.eclipse.org/legal/epl-v10.html

4 Links

https://www.eclipse.org/legal/EclipseLegalProcessPoster.pdf
https://www.eclipse.org/projects/handbook/
https://www.eclipse.org/legal/committerguidelines.php
https://www.eclipse.org/legal/ECA.php
https://wiki.eclipse.org/Development_Resources/HOWTO/Nominating_and_Electing_a_New_Committer
https://wiki.eclipse.org/images/5/50/OpenMDM_IP_management_and_Software_Licensing.pdf