Bugfixing and Support

MidPoint is designed to be efficiently deployed and maintained. Therefore we pay considerable attention to the code quality and testing. However nobody is perfect and as all humans our time is limited. Therefore we want to make sure that we spend our development, testing and bugfixing time efficiently. This page describes the procedures that we follow to do so.

Testing and Bugfixing New Features

The development plan for new midPoint features is specified in Roadmap. Planned roadmap features are transformed to code in each midPoint development cycle. The tests for the new features are developed together with the feature itself. These are usually unit Tests and integration Tests. However initially the tests usually cover only basic scenarios and also mostly only positive test cases. It is responsibility of the engineer that requested the feature to lead the testing of such feature. We expect that the requesting engineer will use the midPoint development version and try to use the feature in real situations. The engineer will undoubtedly find some issues which should be reported back to midPoint team (e.g. in a form of bugreport as specified below). MidPoint team will use this report to improve the tests and to fix the problem. The fix is committed to the development version and it can be re-tested. This process may repeat several times until the feature works as expected.

Bugs and Bugfixes

Bugs happen. Although we work very hard to avoid it would be naive to expect that there are no bugs. Therefore we pay a great attention to the bugfixing process to make sure that any bug is fixed quickly and reliably. The process goes roughly like this:

1. MidPoint user **finds a bug**
2. MidPoint user (or support engineer) files a **bug report** in our issue tracking system (see Creating a Bug Report)
3. The issues is **assigned** to midPoint engineer
4. MidPoint engineer tries to **replicate** the bug. The ideal way is to create a unit or integration test that replicates the bug. This test becomes part of midPoint testing suite.
5. MidPoint engineer **fixes** the bug. The bug is always fixed in the development version of midpoint if possible (see Release Process).
6. The bugfix is **backported** to the support branch if needed. However there is an overhead in backporting the bugfixes. Therefore this happens only for severe bugs that are either critical for midPoint quality (e.g. security issues) or bugfixes that are important for customers with support subscription. Normal community-reported bugfixes are placed only in the development version. See also Support Guidelines for more details.

It is important to emphasize that bugfixing in midPoint is strongly based on replication of the bugs. This is a way how we maintain midPoint quality (see below). Therefore it is important to create good bug reports.

Why?

Because we need to keep midPoint practical, efficient and maintainable.

We avoid spending too much time on testing during the development of a new feature. It is a *new* feature after all. Who knows exactly how it should work? What exactly it should work? Most feature requests are in a very vague form and therefore creating a new feature includes a lot of experimentation. Writing a test based on vague definition is a plain waste of time. Therefore we minimize that only to basic tests that helps the developer to implement the feature. We rather prefer testing by using real-life scenarios. The engineer that requested the feature is responsible for testing it in such real scenarios. Therefore the requesting engineer should check out midPoint Development Snapshot and he should try to use the new feature. Then he finds out what exactly are the circumstances for which the feature works and for which it fails. He can then report these to the midPoint team. The midPoint developer now has the data that he needs to create a good test. A test that reflects reality. Therefore **the developer creates the test first** even before he starts to fix the issue. He can see the test fail, therefore he can be reasonably sure that the test works. **Then the developer fixes the problem**. He can see now that the test passes therefore he can be reasonably sure that the problem was fixed. The fix and the test are committed to **midPoint development version**. The reporting engineer can use the development version to check whether the issues was really fixed and report further problems until the feature works. The tests that were created using this process remain as a part of part of midPoint automated test suite. This is the way how we build a test suite for each feature. This process makes sure that are tests reflect reality as much as possible and that the feature really works in real world scenarios. This also makes the development efficient because we do not create tests for theoretical cases that are unlikely to ever be used.

We highly emphasise **replication** of the problems in tests. Our developers know that they have to try to replicate each reported bug first. If the issue cannot be replicated how would the developer know that he really fixed the issue? Replicating the case saves time and trouble for everybody. We use automated unit and integration tests to replicate the bugs if possible. It still holds that the developer should **create a test first**, see how it fails, **then fix the bug** and see how the test passes. The test will remain in midPoint automated test suite and it will be executed every day. Therefore it is used to avoid regressions - a situation when the bug reappears because something in midPoint implementation was changed. We use this approach to avoid fixing the same bug over and over again.

Most bugfixes and almost all new features will end up only in the development version of midPoint. This is the most efficient way from the development perspective. We generally want to integrate midPoint features very early (which really means "all the time") to make sure that midPoint is a nice integrated system and not just a bunch of unrelated components. Therefore we try to avoid long unsynchronized branches in the development and we try to keep midPoint development mostly linear. Therefore new features and community-reported bugs will be available in production quality only in the next midPoint release. Of course the bugfixes and features are always available in the development branch. Which usually works fine for most cases. However as this branch haven't been through a full quality assurance (testing) cycle yet we cannot guarantee that everything works perfectly. Although it may work for you this branch is not intended for production use. *Hic sunt leones.* You can of course use it as you wish to but you have been warned.

Some bugfixes are **backported** to the support version (see Release Process). But backporting is a tricky process. Sometimes it is easy but it may be quite painful at other times. And it is always an overhead. Strictly speaking the support version of midPoint is always a development dead end. The code that goes to this branch will be obsolete when a new midPoint version is released. It will be thrown away. Therefore we really want to avoid backporting unless there is a very good reason to do it. And there are usually only two good reasons to do this: danger and money. We will backport a bugfix if it is dangerous not to backport it. E.g. we will backport the fix for a security issues or for issues that can potentially destroy midPoint data. But we will **not** backport other fixes - unless there is someone paying for it. Therefore a customer with support subscription has a right to request that a bugfix for his issue should be backported to support version. But even this does not happen automatically. This has to be explicitly requested. We do this because we want to keep the overhead as low as possible. Keeping the overhead low allows us to focus on new development rather than to spend time on things that will be thrown away in few months anyway.

See Also

- Development Process
- Release Process
- Creating a Bug Report
- Support Guidelines