

# Support Guidelines

- [Introduction](#)
- [Issue Tracker Is The Law](#)
- [Priorities](#)
- [Development Cycle](#)
- [Deployment Lifecycle](#)
- [Cooperation](#)
- [See Also](#)

## Introduction

Simply speaking, there are two broad categories of support provided for midPoint:

- **MidPoint subscription** is a commercial (paid) support for midPoint product provided by Evolveum.
- **Community support** is free support on "best effort" basis. This support is provided only by the means of [public mailing lists](#). This is community service, which means it is provided to the community by community. It is not provided by Evolveum. Evolveum only maintains the means of communication (mailing lists) and participates in the service.

However, when it comes to [reporting a bug](#) or [creating feature request](#) the same basic principles apply. This page is summarizing the principles.

## Issue Tracker Is The Law

MidPoint development team is guided by the information in our [issue tracking](#). The content of issue tracker is the plan. Developers work on the issues tracked there. Strictly speaking, they should work **only** on the issues files in the issues tracker. Therefore, if you want your bug to get fixed or your feature to get implemented then it has to go into issue tracker. For midPoint subscribers this is often done by the subscriber or partner, as this the best way to maintain communication fidelity. However it is also a common practice that midPoint developer creates an issue for the subscriber. Regardless of the actual method, all bug reports, feature requests, improvement suggestions and other development tasks must be recorded in the issue tracker to make sure they are addressed.

## Priorities

From the perspective of a support process there is one thing that is more important than anything else: priority. MidPoint development team is guided by issue priorities. MidPoint [Development Process](#) is time-based, with two releases every year. Releases are planned to accommodate estimated amount of support requests and subscriber features. Release dates are always slightly adjusted, but they are not meant to move in any dramatic way. There is no way around the [iron triangle](#). We do not want to compromise on quality and customers are not going to pay more than negotiated amount. Therefore if release dates won't change, then it is the scope that has to change. Low-priority items are moved out from milestones and releases. This usually affects community issues, as subscriber issues are prioritized. But this principle is also important to estimate when a particular issue will be addressed (including subscriber issues).

Following table summarizes the priority system used in midPoint development and support:

Priority	Who can report	Usually used for	Typical reaction time (but no guarantees)	Usage Guidelines	Remarks
<b>Blocker</b>	subscriber	Issues that block everything and everybody. Production environment is down. Financial or reputation damage imminent. <a href="#">Security issues</a> .	<b>Immediate</b> (hours)	This priority is <b>reserved for issues that affect production environment</b> . Do not abuse this priority for issues that only affect testing or development environments. This priority is not applicable for deployment projects that have not yet reached production phase. Exception to this rule are possible only in very severe cases and only for platform subscribers.	<b>Do not abuse this priority.</b> Developers will drop everything and work on this issue. We will be very upset if this priority is abused. It is better to start on lower priorities and escalate as necessary.
<b>Critical</b>	subscriber	Dangerous situation. Production environment is affected in a non-trivial way. Risk of financial or reputation damage.	<b>Quick</b> (hours, days) Must be addressed within development milestone.	This priority is <b>recommended for issues that affect production environment</b> . Do not report issues with this priority that affect only testing or development environment, unless such issues are extremely serious. Use of this priority for deployment project is justified only if the issue blocks entire deployment team and the engineers are not able to proceed with any deployment tasks. Risk of project delays does <b>not</b> justify use of this priority. Deployment project should plan for sufficient reserves for issues discovered during the deployment.	Sometimes used by the development team to mark the most important features in a release. MidPoint won't be released unless all critical issues are addressed. May cause milestone delays. May cause release delays.

<b>Major</b>	subscriber	Uncomfortable situation. Development or project is affected in a non-trivial way. Risk of financial or reputation damage if the issue is not addressed eventually.	<b>Appropriate</b> (weeks, months) Usually addressed within development milestone.	This is the <b>highest priority to use for issues that affect testing and development environments</b> , which is also the highest priority for deployment projects (given the exceptions mentioned above).	Usually addressed within development milestone, but may be postponed to a final stabilization phase (after feature freeze) if needed. Major issues may block releases. But in rare cases even major issues can be postponed to the next release, but that needs to be negotiated with the subscriber. May cause release delays.
<b>Minor</b>	subscriber, community	Minor nuisance. Makes life a bit harder, but not a big problem. Negligible risk of financial or reputation damage.	<b>Eventual</b> (months, years) Usually addressed within a release cycle. But no guarantees.	This is the highest priority that a community issues can have. Higher priorities are possible only if the issue is selected by the development team or endorsed by a subscriber.	Addressed during the final stabilization phase (after feature freeze). Development team will try to address as many minor issues as can fit into the development cycle. But it is almost certain that some of them will be left unsolved and moved out. Minor issue will never block a release.
<b>Trivial</b>	subscriber, community	Cosmetic issue. Not really important. Just an idea.	<b>Unpredictable.</b> No guarantees. No promises. No estimations.		Can be freely moved in the plan. The most likely candidate to be moved when the time runs out.

Times indicated above are the times when a **developer starts working on the issue** after it is reported. Which means actual developer, not some kind of automated response system that acknowledges receipt of the report. The numbers above are our best estimates that we have about the times and we will try hard to fit into that time. But there may be rare cases, such as when the development team is overloaded by support requests or when a developer with a special skill is temporarily unavailable. The times may be a bit longer in such cases.

**Time to fix is not guaranteed** for any issue. We are sorry, but we cannot do that even if we wanted to. Some issues might be fixed in a couple of hours. But it may take days or even weeks to fix issues that are triggered by exotic configurations, issues that appear randomly, issues triggered by external events and so on. It is impossible to predict how long the fix will take, therefore we simply cannot guarantee that. The priority system is the best we can offer.

This may sound harsh, but there is a good reason for that. We work with L3 issues, which means product bugs, feature requests and similar product-based issues. We do not have the option to "hack" or "work around" the issues that affect only specific configuration. However, deployment engineer may still be able to work around the issue on L2 level. E.g. the issue may be avoided by changing the configuration, isolating midPoint from external events, mitigating effect of the issue and so on. This is where issue resolution times might be predictable. You may have such options because you know your environment, configuration and tolerances. But we do not have such privilege. Therefore we cannot guarantee fix times.

Developers are free to increase priority of any subscriber issue. Priorities of non-subscriber issues can be changed by developers in any way they seem appropriate, but priorities of those issues are usually going down. If you do not like this, there is a simple way to improve your chances: [get midPoint subscription](#).

**Security issues are always the highest priority**, no matter who is the reporter. When reporting issue to the issue tracker please clearly indicate that this is an security issues (e.g. use word SECURITY in the title). Appropriate priority will be set be the developer reviewing the issue. If the issue report is sensitive and it may put others at risk then you can use our responsible disclosure mail address [security@evolveum.com](mailto:security@evolveum.com). See [Security Guide](#) page for more details.

**Priority abuse:** Please, do not abuse the priority system. Attempts to abuse priority system may result in decreasing issue priorities (including subscriber priorities) and/or lower success rates during escalations. We will absolutely hate to do that. Therefore please do not force us to do it. If there is some confusion about appropriate priority it is usually better to select lower priority and explain the situation in the comment. Every new issue is reviewed by midPoint team member. The priority sometimes gets increased during this review if the reviewer thinks that a higher priority is appropriate or if there is a risk that the issue may affect larger number of users.

## Development Cycle

Development cycle is the same for every release: There are development milestones. Those are usually three milestones M1, M2 and M3. Each milestone will introduce new functionality. There is a dedicated time for bugfixing at the end of each milestone. That's where major-priority issues are addressed.

Last development milestone is a *feature freeze*. This means that all features planned for the release are done. Feature freeze is followed by a stabilization phase. That is the time of a more intense testing. All development efforts are dedicated to bugfixing. All "red" (blocker, critical, major) issues should be addressed at this time. Some "green" issues (minor, trivial) are likely to be addressed as well, but it is almost certain that not all of them will be solved. Remaining issues will be re-planned when release date comes.

## Milestones

Development milestones were introduced in midPoint 4.0 release. Therefore please allow some time for the development process to adapt to this new regime. Therefore the times and procedures indicated above may slightly vary during the first few releases in the 4.x family. We kindly ask for patience and understanding. We are doing our best, but developers are people too.

## Deployment Lifecycle

MidPoint deployment projects have a lifecycle of their own. There is usually some exploration/preparation phase. Then there is development and testing. Then the project is deployed to production. Then further iterations follow. Characteristics of individual deployment phases considerably vary when it comes to impact and criticality of the issues. We have tuned our support model to provide the right balance for the needs of individual deployment phases. The guidelines are summarized in the following table:

Phase	Maximum issue priority	Description
Exploration /preparation	Minor	This phase is usually not covered by any support or subscription. Therefore the maximum priority of an issue is minor, as that is the maximum priority of a community issue.  In case that this phase is indeed covered by support or subscription, then the same rules as for deployment/testing phase apply.
Deployment and testing	Major	Configuration is customized and tested, data migration is prepared and so on. But everything happens in the "lab" (development and/or testing environment). Production systems are not affected. MidPoint is not yet in the production, therefore the potential for any real harm is reduced. Therefore the maximum priority is reduced. This allows midPoint team to focus on more serious issues.  Please, be patient in this phase. Plan sufficient reserves in your project. Your issues may have lower priority now. But once you go to production you will surely appreciate that your issues will have a higher priority than the issues of new deployment projects.
Production	Blocker	MidPoint is running with real data. The deployment is supposed to be stable. MidPoint should not misbehave at this point. If midPoint happens to misbehave and there is a potential for harm, then such issue has to be fixed as soon as possible.  This also applies to <a href="#">security issues</a> regardless of project phase.
Follow-up iterations	Major	MidPoint deployment is in production - and the entry above applies to such production operation. However, there is often a new iteration started in parallel. Improved midPoint configuration is prepared in development environment, it is moved to testing after that, update of production environment is planned. Such follow-up iterations are handled in a same way as an original deployment project. While issues from the production environment can still be raised to the highest priorities, the issues that originate from development or testing environment have reduced priority.

Those guidelines reflect our philosophy that prefers production deployments. While many products and programs are designed to support deployment projects where most of the money is generated, midPoint is quite different. We believe that the most important thing is to keep real (production) midPoint deployments running. The projects that are in the process of deploying midPoint can wait a bit. But production deployments that work with real data cannot wait. They need to have absolute priority.

This may seem harsh for deployment projects. But experienced deployment engineers and managers are already well-equipped for this. The project should proceed in smaller steps, testing the configuration is several iterations (prototyping), the project should have several alternative paths and the project plan should contain sufficient reserves for deployment issues. This is a best practice in the entire IT industry and IDM deployment projects are no exceptions here.

We would also like to emphasize the importance of deployment environments. MidPoint is especially designed for deployments that follow the "three environment model":

Environment	Purpose	Maximum issue priority
Development	Develop new configuration, configuration changes, prepare customizations, etc. MidPoint is connected to development instances of source/target systems. Some systems may be simulated.  Works with dummy data. Amount of data is usually reduced.	Major
Testing	Test configurations after the development is done. Validate the system before deployment to production. MidPoint is connected to testing instances of source/target systems.  Works with simulated data, copy of real production data or anonymized production data. Same amount of data as in production environment.	Major

Production	MidPoint running with real source and target systems. Works with real data.	Blocker
------------	--	---------

Preparing and maintaining those three environments is strongly recommended for all deployment projects. Any significant change to the configuration should proceed from development environment to testing environment and only then it should be deployed to production.

There may be cases when those environments are not used and changes are done directly in production environment. Or cases, where development and testing environments are ineffective because the environment, configuration and data do not reflect the reality of production environment. We do not guarantee that our support services will be ideal fit for such environments. Both midPoint and our services were designed with proper engineering principles in mind and they were designed for deployments where such engineering principles are honored. In case that you are doing configuration changes directly in production environment, you should follow the priority limitations of development/testing environments. In such cases your first reaction to problems should be to roll back the configuration changes and revert to stable configuration. Which solves the problem. The sole fact that this problem was observed in production environment does not justify high priority of the issue.

## Cooperation

Most issues cannot be properly addressed unless there is a good cooperation between issue reporter and developer. The developer often needs additional data for the issue. Our usual strategy for all issues is to follow [test-driven bugfixing](#) approach. Therefore we try to reproduce the issue in a controlled environment. Additional data are often needed to achieve that. We expect that it is a responsibility of the reporter to respond to requests for additional data. The usual communication is carried out by the means of comments in the issue tracking system.

We reserve the right to close the issue if the reporter does not respond to communication.

Those guidelines are designed to benefit the entire midPoint community. We do not look well at those that abuse those guidelines. MidPoint development and support team has finite resources. The abusers may get momentary advantage for themselves, but this approach distracts midPoint team from the tasks that are really important. Therefore we reserve to lower the priorities for reporters that repeatedly abuse those guidelines. This is the best approach for the entire midPoint community.

## See Also

- [Subscriptions and Sponsoring](#)
- [Creating a Bug Report](#)
- [Security Guide](#)