Complete Relativity

Planned feature

This page describes a feature planned for future midPoint versions.

This feature is roughly designed and it was evaluated as feasible. However, there is currently no specific plan when it will be implemented because there is no funding for this development yet. In case that you are interested in supporting development of this feature, please consider activating midPoint Platform subscription.

Introduction

MidPoint is designed and build with relativity in mind. This means that midPoint always thinks in a notion of relative changes (deltas). However, midPoint computational core (a.k.a clockwork and projector) often require knowledge of the absolute state of a resource object such as account or group. This is not part of the original design and strictly speaking it should happen only when objects are *reconciled*. But midPoint implementation evolved from simple algorithms to a more complex one. Features such as provisioning dependencies or consistency mechanism forced us to do some compromises. Those were compromises that do **not** invalidate the original design. The design is still valid. Those compromises were made to make the implementation feasible in given financial and scheduling constraints.

Mappings and expressions also suffer from a similar problem. While mappings are designed to be fully relative, current implementation is evaluating mappings too often. E.g. some mappings are evaluated even if we do not need the result and the result is in fact discarded later. Some mappings are triggering load of a full resource object (e.g. account) even if this is not strictly necessary. This does not affect correctness of the computation. In fact it makes the computation more reliable - and that is also one of the reasons that we have made this compromise. But performance may suffer. This may affect especially deployments with large identity population and many small changes.

Openness

This information is provided under the openness paradigm that guides the development of midPoint. MidPoint is a great software product. But it is also a practical software product developed in a pragmatic way. Therefore midPoint is not perfect. The goal of midPoint is to fulfill practical goals and solve problems that affect midPoint subscribers right now. Therefore it is perhaps completely understandable that midPoint has its drawbacks.

In the spirit of openness that is embedded in midPoint development culture we do not try to cover midPoint's drawbacks. **We try hard to communicate all midPoint limitations and drawbacks very openly.** Even if midPoint has some issues, as far as we are aware those issues do not invalidate midPoint architecture and design. As far as we know it is perfectly feasible to address all those issues. And that is exactly what midPoint platform subscription can be used for.

This problem did not affect midPoint deployments too much when ConnId connectors were the only mechanism to detect changes. Due to limitations of ConnId that are inherited from the past the connector cannot detect fully relativistic changes anyway. The connector usually returns complete new state of a resource object (e.g. account). Therefore there is no penalty in using that absolute state in the computations in this case. However, the requirements may gradually change when new asynchronous method of change detection are introduced, such as messaging resources.

Implementation

The plan is to improve implementation of Clockwork and Projector to support completely relativistic mode. This will make the algorithm more complex and it is very likely to trigger several refactoring waves over existing code. But according to our evaluations this approach is completely feasible. It is also the right thing to do to align the implementation with the original desig and the ideal of relativity.

See Also

- Relativity
- Mapping Relativity
- Messaging Resources