Multiaccounts HOWTO

MidPoint 4.0 and later

This feature is available only in midPoint 4.0 and later.

Introduction

MidPoint allows to use multiple accounts on a particular resource, provided that they have unique kind and intent combination. However, there are cases that this is not quite enough. There is often a case that a particular resource can have more than one resource with that kind/intent combination. An example can be an HR system that keeps a single record for every employment contract for a particular user. Therefore there may be multiple contracts during employee lifecycle, some of them valid, other already expired. We cannot use *intent* to distinguish between them because all the intents have to be strictly defined in resource schema handling. But in this case we do not know how many contracts the user can have, therefore there is no good way to define the intents.

MidPoint 4.0 introduced a new concept of tag to distinguish such accounts. There may be as many accounts as needed as long as they have unique (resource, kind, intent, tag) combination (also known as *discriminator*). Unlike intent, tags are dynamically generated and they do not have to be defined beforehand.

Resource Configuration

The use of "multiaccounts" feature is disabled by default. The ability to use allow multiple accounts and use tags needs to be explicitly enabled in resource schema handling by using `multiplicity` element:

```
<resource>
    ...
    <schemaHandling>
        <objectType>
            ...
            <multiplicity>
                                <maxOccurs>unbounded</maxOccurs>
                        </multiplicity>
            ...
        </objectType>
    <schemaHandling>
    ...
</resource>
```

This setting allows multiple resource objects to exists (and be linked to a focus) for that particular object type. Each of the objects needs to be distinguished by a unique tag value.

Generating Tag Values

Value of "tag" is used to distinguish resource objects that have the same kind/intent combination. Similarly to kind and intent, the tag is stored in shadow object. Tags are generated only for those shadows where tag is needed. Which means that tags are generated only when the "multiaccounts" configuration is enabled and only for those object types where it is needed. The tag values are generated at the same time as kind and intent. Which means that the tag is generated in synchronization code for the inbound direction.

The default tag value is OID of the shadow. This is a convenient value to use, as it is permanent and globally unique. Therefore it should work well even if account ownership is changed.

Tag Expression

While shadow OIDs make very good tag values, there are some drawbacks.

Firstly, OIDs are ugly. In fact, OIDs are ugly by purpose. And that does not cause much trouble for ordinary objects as for them OIDs can be easily translated to user-friendly names. But we do not have that option for tags. Generating nicer tag values may help a lot with visibility and diagnostics.

Secondly, OIDs are always unique. Which may look like a good thing, but there are situation when it is not. Even though we allow multiple accounts to be linked the same focus, that does not mean we want unlimited and uncontrolled number of such accounts. There may still be duplicates and conflicts. We may want to detect duplicates by generating tags based on account properties.

Fortunately, there is an easy way how to generate tag values by using an expression:

```
<resource>
    ...
    <schemaHandling>
        <objectType>
            ...
            <multiplicity>
                                <maxOccurs>unbounded</maxOccurs>
                <tag>
                                        <expression>
                                                <script>
                                                        <code>basic.getAttributeValue(projection, "contractId");
</code>
                                                </script>
                                        </expression>
                                </tag>
                        </multiplicity>
            ...
        </objectType>
    <schemaHandling>
    ...
</resource>
```

The expression will get the shadow filled with all the attribute values set in variable `projection`. Responsibility of the expression is to return a tag value. The expression is invoked only when tag value is not know. Once the tag is set, it is considered to be fixed. Therefore choose the expression algorithm wisely. Use of account attributes that have unique value and do not change is a good idea. Contract identifiers or even permanent account identifiers are a good choice.

Ordinal tags

Avoid the temptation to use ordinal values as tags (e.g. "#1", "#2" and so on). Such tags may look good, but they are very problematic in practice. First problem is how to assign them, as they depend on other accounts that are linked to the user. Then there is problem of numbering "gaps" when a particular account is deleted. But perhaps the worst problem is when account ownership needs to be changed. Then the tags need to be recomputed. Having tags that change all the time create a fragile configuration and makes it difficult to diagnose the situation. It is always better to find some kind of identifier in the account data to be used as a tag.

MidPoint lacks convenient support for tags as ordinary numbers. This is somehow done by purpose, to avoid the temptation. However, we might be willing to implement it given sufficient funding.

Inbound Mappings

Since midPoint 4.0, inbound mappings work across all the accounts. They merge the output values together. This works also with all the "tagged" accounts. E.g. the common case is that each of the accounts will produce one assignment representing employment contract. This is usually implemented in an inbound mapping. Therefore there is just one inbound mapping that produces one assignment. But there may be several "tagged" accounts for which this mapping is applied. All such assignments from all the accounts are merged together and set to the user.

Mapping Range

One of the difficult parts of setting up inbound mappings is the problem of a range. E.g. typical situation is that some assignments are managed automatically by the means of inbound mappings. But other assignments are managed manually. It is important to distinguish those assignments and properly set range of inbound mappings. Otherwise inbound mappings may overwrite assignments that are assigned manually or managed by other means.

See Inbound Mapping and Mapping pages for the details.

Limitations

This "multiaccounts" feature is not implemented completely. The implementation is currently limited:

- Multiple resource objects are currently supported **only in inbound direction**. I.e. it works only for authoritative source resources. This feature will not work in the outbound direction. It may not not work even if inbound and outbound mappings are combined in a single resource.
- GUI support is very limited.
- Migration between single-account and multi-account setups is not supported. The shadows must be created in an appropriate setup (e.g. with tag or without tag).
- Tags cannot change. Once set, the tag is considered to be fixed. It is not updated when resource object is renamed. It is not updated when owner is changed.

All those limitations can be removed with appropriate platform subscription.

See Also

- Kind, Intent and ObjectClass
- Inbound Mapping