Persona Configuration

This feature is available in midPoint 3.6 or later.

Introduction

Please see Personas page for an introduction.

Persona Construction

Personas are managed by using persona constructions. Persona construction is a mechanism that is very similar to ordinary ordinary construction used to manage resource objects. The persona constructions can be placed in assignments/inducements in a way which is almost identical to ordinary constructions. However, the persona constructions control the way how personas are created.

Persona construction looks like this:

```
...
<inducement>
    <personaConstruction>
        <targetType>UserType</targetType>
        <targetSubtype>admin</targetSubtype>
        <objectMappingRef oid="894ea1a8-2c0a-11e7-a950-ff2047b0c053"/>
    </personaConstruction>
</inducement>
...
```

Meaning of individual setting is as follows:

- `targetType`: Type of the persona object. Usually UserType.
- `targetSubtype`: Subtype of the persona. In case of UserType personas this setting specifies the value of `employeeType` property of the resulting persona.
- `objectMappingRef`: Reference to the object template which will be used to map data from the focus (source user) to the persona (see below).

Type and subtype together form unique key persona key. I.e. construction with the same type/subtype combinations are evaluated as if they specify the same persona. In fact, the subtype is very important for any persona. Each persona much have a subtype value. This value is used to distinguish persona types.

Subtype

Subtype was replaced by Archetypes in midPoint 4.0. However, full support for archetyped personas was not implemented yet.

Object Mapping - Object Templates

The mapping between the source user (focus) and persona is implemented by a use of object template. The usual use of object template is such that the same object is both the source and the target of object template mapping. However, it is all different when working with personas. In such case the focus object (source user) is the source of the object template mappings and the persona is the target. Therefore the object template can be used to map data from the source user to personas. When object template is used in this way we refer to it as *object mapping*.

The object template is used without any major change in the syntax:

```
<objectTemplate oid="894ea1a8-2c0a-11e7-a950-ff2047b0c053">
    ...
    <item>
        <ref>name</ref>
        <mapping>
            <source>
                <path>name</path>
            </source>
            <expression>
                <script>
                    <code>'a-' + name</code>
                </script>
            </expression>
        </mapping>
    </item>
    <item>
        <ref>givenName</ref>
        <mapping>
            <source>
                <path>givenName</path>
            </source>
        </mapping>
    </item>
    <item>
        <ref>familyName</ref>
        <mapping>
            <source>
                <path>familyName</path>
            </source>
        </mapping>
    </item>
    <!-- Full name will be computed using ordinary user template. No need to map it here. -->
    <mapping>
        <!-- initial persona password mapping -->
        <strength>weak</strength>
        <source>
            <path>credentials/password/value</path>
        </source>
        <target>
            <path>credentials/password/value</path>
        </target>
    </mapping>
</objectTemplate>
```

The template above is responsible for mapping focus to the 'admin' persona. It will create a username for the persona using the `a-` prefix. It will also map givenName and familyName. There is also a weak mapping for the password, which is used to set an initial password for the persona. This mapping will make sure that the `name`, `givenName`, `familyName` and password are set up when a new admin persona is created. It will also make sure that any update for `name`, `givenName` and `familyName` in the source user are properly propagated to the persona.

When working with object templates as object mappings please keep in mind that source and target are different objects. They in fact may even be a different object types, because the persona may be different object type that its source.

Persona Assignments and Roles

Persona constructions are usually placed in inducements that are placed in *persona roles*. Persona roles are just ordinary roles, but instead or granting accounts and groups they grant new personas. A persona role may simply look like this:

```
<role>
    <name>Persona: admin</name>
    <inducement>
        <personaConstruction>
            <targetType>UserType</targetType>
            <targetSubtype>admin</targetSubtype>
            <objectMappingRef oid="894ea1a8-2c0a-11e7-a950-ff2047b0c053"/>
        </personaConstruction>
    </inducement>
</role>
```

When this role assigned to a user then a new 'admin' persona will be created for that user. When this role is unassigned then the persona is deleted. This all works in almost the same way as ordinary roles and constructions work.

Persona Links

MidPoint keeps track about persona ownership by using persona links. These links are simple `personaRef` object references:

```
<user oid="df39166a-30cf-11e7-9aa3-03298e38b048">
    ...
    <employeeType>physical</emloyeeType>
    ...
    <personaRef oid="e59a75d0-30cf-11e7-a5e2-a71b5b1d913a" type="UserType"/>
    ...
</user>

<user oid="e59a75d0-30cf-11e7-a5e2-a71b5b1d913a">
    ...
    <employeeType>admin</emloyeeType>
    ...
</user>
```

Persona links are automatically created when a new persona is created. And they are automatically deleted when a persona is deleted.

Personas and Authorization

User that has linked personas is considered to be owner of the personas for the purposes of authorizations. Therefore following authorization can be used to allow users to see their personas:

```
     <authorization>
        <name>self-persona-read</name>
        <description>
            Allow to read all the personas of currently logged-in user.
        </description>
        <action>http://midpoint.evolveum.com/xml/ns/public/security/authorization-model-3#read</action>
        <object>
            <type>UserType</type>
            <owner>
                <special>self</special>
            </owner>
        </object>
    </authorization>
```

Since midPoint 3.6 this authorization is part of the default *End User* role.

However, the situation is more complicated when it comes to persona modifications. Personas are usually assigned in a form of roles. Therefore there is no need for any special authorization for the assignment itself (authorization request phase). However, personas are quite special when it comes to execution. Assignment of a new persona means that a new user needs to be created. The authorization for this operation is evaluated in the usual way - the user who started the operation needs to be authorized for all the effects of the operation. Which is especially important in the case, when a user requested persona role for himself. Then the requesting user must have authorizations to create new users (personas). MidPoint is implemented in such a way, that request-phase authorization to create users is not needed as this is all considered to be just an effect of persona role assignment. However, execution-phase authorization is required.

However, the execution-phase authorizations to create new users are **not** part of the default *End User* role. Blank authorization to create any kind of user may just be too dangerous. This is an execution-phase authorization so in theory the request-level authorization should prevent security breach. However, even very generous execution-phase authorizations may be dangerous in case of construction and mapping misconfiguration. And a broad authorization for all users might pose risk even for privileged users. Therefore we have decided **not** to put such a broad authorization in the end user role by default. The end user role needs to be customized for a specific deployment that is using personas. We recommend adding authorizations that are constrained to specific persona types that the users may request:

```
<authorization>
    <name>auth-persona-execute-add</name>
    <action>http://midpoint.evolveum.com/xml/ns/public/security/authorization-model-3#add</action>
    <phase>execution</phase>
    <object>
        <type>UserType</type>
        <filter>
            <q:equal>
                <q:path>employeeType</q:path>
                <q:value>admin</q:value>
            </q:equal>
        </filter>
        <!-- owner constraint cannot be here, the link does not exists when the persona is added -->
    </object>
</authorization>
<authorization>
    <name>auth-persona-execute-modify-delete</name>
    <action>http://midpoint.evolveum.com/xml/ns/public/security/authorization-model-3#modify</action>
    <action>http://midpoint.evolveum.com/xml/ns/public/security/authorization-model-3#delete</action>
    <phase>execution</phase>
    <object>
        <type>UserType</type>
        <filter>
            <q:equal>
                <q:path>employeeType</q:path>
                <q:value>admin</q:value>
            </q:equal>
        </filter>
        <owner>
            <special>self</special>
        </owner>
    </object>
</authorization>
```

It is also a good idea to constraint these authorizations even further by only allowing those items that are used in the object mapping (object template).

Miscellaneous

Since midPoint 3.7 password policy can be used to enforce different passwords on linked personas.

Limitations

The implementation of personas in midPoint 3.6 is limited. The persona functionality is perfectly usable for most use-cases. However advanced use cases may not be supported. Currently known limitations include:

- Approvals: The operation that automatically provisions, deprovisions or updates a persona must not be subject to approvals. This means that the automatic operations on personas must all be completely automatic and synchronous. It is OK to map change of names or other properties from source focus to persona. It is also OK to use the object mapping to create assignments as long as they are are not subject to approval. But it is not OK to use persona object mapping to create an assignment that is subject to approval. In that case the system will behave in unpredictable way. The workaround is to automatically assign only those roles that are not subject to approval. Then let the user log in with the persona credentials and request additional roles for the persona. Once the persona is provisioned then the request-approval process works without limitations.
- Error handling: If more than one persona is provisioned at the same time then an error in one persona may cause the other persona not to be provisioned.
- Construction merging: Currently only one persona construction is supported for each persona. MidPoint cannot currently merge two persona constructions and apply them both. Attempt to assign two persona constructions that refer to the same persona at the same time will result in an error.
- User-user personas only: Currently only the user-user scenario is tested. This means that that both the focus (source) and the persona must be of UserType. Other combinations may work under some circumstances, but they are not tested and currently not supported.

Incomplete feature

This is an incomplete feature of midPoint and/or of other related components. We are perfectly capable to implement, fix and finish the feature, just the funding for the work is needed. Please consider the possibility for sponsoring development of this feature. If you are midPoint platform subscriber and this feature is withing the goals of your deployment you may be able to use your subscription to endorse implementation of this feature.

See Also

- Personas