Archetype Configuration

MidPoint 4.0 and later

This feature is available only in midPoint 4.0 and later.

Introduction

Simply speaking *archetype* is a well-defined object subtype. Employee, Contractor, Project, Workgroup, Application, ... each of them is an archetype.

Please see Archetypes page for a more in-depth introduction to archetype concepts.

Archetype Definition

Archetype definition is a special midPoint object (ArchetypeType):

```
<archetype oid="7135e68c-ee53-11e8-8025-170b77da3fd6">
   <name>employee</name>
   ...
   <archetypePolicy>
       <!-- definition of archetype icon, color and labels will be here -->
   </archetypePolicy>
   ...
</archetype>
```

Archetypes define character and behavior of "archetyped" objects in `archetypePolicy` section. This part of archetype definition specifies what icons, color and labels to use, which templates to apply and so on.

Archetypes work like abstract roles, therefore all objects that need to be "archetyped" can be simply assigned to an archetype. Being a midPoint object, archetype has an natural persistent identifier: OID. OID of the archetype definition is also identifier of archetype itself. Each object of that archetype will contain archetype definition OID in `archetypeRef` reference:

```
<user>
   <name>foo</name>
   <archetypeRef oid="7135e68c-ee53-11e8-8025-170b77da3fd6" /> <!-- "employee" archetype -->
   ...
   <assignment>
       <targetRef oid="83614aaa-ee57-11e8-b770-7f3cb1d00719" type="RoleType"/> <!-- some role, that includes
another role, that includes "employee" archetype -->
   </assignment>
</user>
```

In the above example, user was created with the archetype assignment, but without the `archetypeRef` reference. The `archetypeRef` reference is computed by midPoint and stored into the object. This reference is needed for easy and efficient resolution of an archetype, because archetype may be assigned indirectly. The `archetypeRef` reference can also be used to search for all objects of a given archetype.

Archetypes as Abstract Roles

Archetype definitions are abstract roles, which means they essentially work as a role. To apply an archetype to an object simply assign archetype to that object as you would normally assign a role. From that point on archetype applies to that object. Archetypes being abstract roles makes them really powerful. That means archetypes may work as ordinary roles or metaroles:

```
<archetype oid="7135e68c-ee53-11e8-8025-170b77da3fd6">
   <name>employee</name>
   ...
   <inducement>
       <!-- whatever roles, rules and privileges that an employee needs to have -->
   </inducement>
</archetype>
```

Being an abstract role, archetype may also be assigned indirectly. E.g. `basic employee` role may contain inducement for `employee` archetype, making sure that all holders of `basic employee` role will have appropriate archetype.

Icons, Colors, Labels

Archetype policy can be used to set custom icon, color and label for "archetyped" objects:

```
<archetype oid="7135e68c-ee53-11e8-8025-170b77da3fd6">
   <name>employee</name>
   ...
   <archetypePolicy>
        <display>
            <label>Employee</label>
            <pluralLabel>Employees</pluralLabel>
            <icon>
                <cssClass>fa fa-user</cssClass>
                <color>blue</color>
            </icon>
        </display>
    </archetypePolicy>
    ...
</archetype>
```

The example above is setting custom icon for `employee` archetype. It also specifies that whenever an employee icon can be used in color then blue color should be used. There is also a label that should be used to display objects of that particular archetype. And also a plural form of the label used when objects of that archetype are listed.

Archetypes And Views

If an archetype is assigned to an object, midPoint honor that archetype definition. E.g. it will display those objects with appropriate icon and color. But otherwise archetype definitions and object lists are not displayed in the user interface by default - except for places such as Repository Objects. This behavior has several reasons: there may be archetype definitions that are just being prepared, definitions that should not be publicly visible, or there may be just too many definitions to be displayed in the menu. Therefore, by default, there will be no additional menu items or "shortcut" buttons for archetypes.

Archetypes can be added to menus and other places of midPoint interface by using object views. All that is needed is simple definition of a view in admin GUI configuration structure:

```
...
<adminGuiConfiguration>
    <objectCollectionViews>
        <objectCollectionView>
            <identifier>empls-view</identifier>
            <type>UserType</type>
            <collection>
                <collectionRef oid="7135e68c-ee53-11e8-8025-170b77da3fd6" type="ArchetypeType"/> <!--
"employee" archetype -->
            </collection>
        </objectCollectionView>
    </objectCollectionViews>
</adminGuiConfiguration>
...
```

This can be defined in global system configuration or in a role. Such view should create "Employees" link in the menu in the "Users" section.

The view specifies a way how a collection of objects is displayed. Creating appropriate menu item is just one of the aspects. The view may also define how the actual list of objects looks like. The view specifies the columns, defaults search settings and so on. However, when the view is bound to an archetype, it can do even a bit more. Such view can automatically render buttons to create new objects.

Assignment Relation

Archetypes can define possible object types and relations that can be used in assignments to "archetyped" objects, e.g.:

- Projects are organizational units that can have members, managers and owners.
- Business roles have members, owners and approvers. Owners and approvers must be employees.
- Departments are organizational units that may contain other organizational units (but only if they are of Section archetype), users and business roles.

There is an *assignment relation* mechanism that can be used for that purpose. The assignment relation specification can be used to limit possible assignment holder object types and assignment relations. This is perhaps best illustrated using an example of a business role:

```
 <archetype oid="018e7340-199a-11e9-ad93-2b136d1c7ecf">
    <name>Business Role</name>
    ...
    <inducement>
        <assignmentRelation>
            <description>Any user can have business role (can be a member).</description>
            <holderType>UserType</holderType>
            <relation>org:default</relation>
        </assignmentRelation>
        <assignmentRelation>
            <description>Only employees may be owners/approvers for business role.</description>
            <holderType>UserType</holderType>
            <holderArchetypeRef oid="7135e68c-ee53-11e8-8025-170b77da3fd6"/> <!-- Employee archetype -->
            <relation>org:approver</relation>
            <relation>org:owner</relation>
        </assignmentRelation>
    </inducement>
    ...
</archetype>
```

First `assignmentRelation` in the above example specifies that any user can be assigned to the business role with default relation. Second `assignmentRelation` specifies rules for `approver` and `owner` relations. Only an employee can be owner or approver of the business role.

Please note that in this case `assignmentRelation` specifications are placed in the **inducement** of the archetype, not assignment. We want to apply `assignmentRelation` to "archetyped" objects. And that is exactly what inducements do. But archetype definition are itself first-class midPoint objects - and they are also abstract roles. Therefore archetype definition can have assignments pointing to it, such as owner of an archetype definition. Therefore the `assignmentRelation` statements in the assignment also make sense if we want to control what objects can be assigned to the archetype definition. But placing `assignmentRelation` in inducement is the usual case.

Open and Closed Assignment Relations

MidPoint 4.1 and later

This feature is available only in midPoint 4.1 and later.

Assignment relation specifies which objects can be assignment to other objects. But how to interpret the situation when there is no assignment relation specified? This may mean two different things:

- **Open** approach: Assignment relation is not used at all. Any assignments to any objects is possible. User interface will render a button that allows to assign any combination of target object and relation. This is the behavior compatible with midPoint 3.9 and earlier. And we still want to maintain that compatibility. This the only option for midPoint 4.0 and it is also the default behavior of later midPoint versions.
- **Closed** approach: No assignment relations are possible. Only those relations that are explicitly specified should be allowed. This option is ideal for systems that have archetype configuration finished and cleaned up.

User interface will always render buttons that allow assignment of specific object types given by assignment relation. E.g. a button to "assign business role" will always be there (assuming that there is a "business role" archetype with appropriate assignmentRelation). The difference between open and closed approach is that in the open mode the "generic" assignment button will be rendered in addition to other buttons.

The open/closed approach can be specified in archetype policy:

```
<archetype oid="7135e68c-ee53-11e8-8025-170b77da3fd6">
   <name>employee</name>
   ...
   <archetypePolicy>
        ...
        <assignmentHolderRelationApproach>closed</assignmentHolderRelationApproach>
        ...
    </archetypePolicy>
    ...
</archetype>
```

The configuration above essentially means that whenever an employee is edited in midPoint user interface the "generic" assignment button will **not** be rendered. Only the buttons given by explicit assignmentRelations are rendered.

Both open and closed mode are still limited by authorizations, of course.

This setting controls behavior of midPoint user interface. E.g. setting the approach to "closed" will hide the button that controls generic assignment in user's the "Assignments" tab. But it will not disable similar button in the "Members" tab of the role. This setting is only about controlling uni-directional behavior of GUI. It does not constraint the entire assignment model. That will be too complex to implement (at least for now).

The assignmentHolderRelationApproach controls the "holder" side of the relation. A similar property that can limit the "target" side of the relation (e.g. buttons in the "Members" tab) is planned for the future.

For midPoint 4.1 this can be configured only on per-archetype basis. There is no global setting that can set open/close as a default for an entire system. The plan is to implement this later together with "inheritance" of object policy configurations in system configuration objects. E.g. object policy for UserType inheriting from object policy for ObjectType - and user archetypes inheriting from the User Type policy.

Order Constraints

Assignment relation applies only to assignments by default. Therefore it controls when an assignment can be made. It does not apply to inducements - yet. In later midPoint versions there will be an element that can specify "order constraints". In that case assignment relation could specify properties of inducements, including high-order inducements. However, the implementation in midPoint 4.0 is limited to assignments.

Default Definition

Archetype definitions can specify details of behavior for "archetyped" objects. But there are also objects that do not have any archetype. We may want to specify behavior for those objects as well. And in fact this was possible in midPoint for ages, even though it was not explicitly denoted as having anything to do with archetypes. There is `defaultObjectPolicyConfiguration` container in system configuration object:

```
<systemConfiguration>
   ...
   <defaultObjectPolicyConfiguration>
      <type>UserType</type>
      <objectTemplateRef oid="10000000-0000-0000-0000-000000000222"/>
   </defaultObjectPolicyConfiguration>
   ...
</systemConfiguration>
```

In fact, the data structure of `defaultObjectPolicyConfiguration` is almost identical to the structure of `archetypePolicy` in the archetype definition. And it also works in almost the same way. This is the definition that is applied to non-archetyped objects of that particular type. And parts of that definition may also apply to archetyped objects, as this definition is merged with `archetypePolicy`. Of course, `archetypePolicy` will override any aspects of the default specification. But the aspects that are not defined in `archetypePolicy` are taken from the default global policy.

In midPoint 4.0 there may be some limitations, e.g. changing global object icons or colors in the system configuration may not work properly. Those things were hardcoded in midPoint user interface for years. And as this is mostly a user interface functionality it is not easy to hunt down and fix all the issues. But those things should steadily improve in following midPoint versions.

Pre-Defined Archetypes

MidPoint is designed to fit in many environments and those environments may be unlike each other is a very significant way. However, there are still few things that most of the environments have in common. There are types of objects that are used in almost any midPoint deployment. Therefore midPoint 4.0 has a few default archetypes that can be used as starting point for further midPoint configuration:

Archetype	Member Object Type	Member Objects	Description
System User	User	administrator	Archetype for system users, i.e. non-person users that are needed for system to work. This may be (root-like) system administrator, application users and so on.
System Role	Role	superuser, approver, reviewer, delegator	Archetype for roles that are essential from the system point of view. Those are usually roles for the most powerful system administrators, roles for internal usage in the system (e.g. by tasks) and so on.
Business Role	Role	*none*	Archetype for roles that have meaning from the business perspective. Business roles are usually assigned directly to users, often by using request-and-approve processes. Business roles are usually composed from smaller roles.
Manual Provisioning Case	Case	*none, assigned dynamically*	Archetype for cases that describe manual provisioning operations.
Operation Request	Case	*none, assigned dynamically*	Archetype for cases that describe operation requests, e.g. role assignment requests.
Approval Case	Case	*none, assigned dynamically*	Archetype for approval cases, e.g. role assignment approval.

Some of the archetypes are provided as a starting point for system configuration. This is namely the *Business Role* archetype. Feel free to modify those archetypes. Those are provided in the initial objects mostly to keep the terminology of midPoint deployments somehow aligned. This makes communication in midPoint community smoother.

There are also archetypes that are essential for proper midPoint functionality, e.g. the archetypes for cases. While you can still modify those, you should have good understanding of how midPoint works and what effects can those changes may have. Please be careful here.

Authorizations

Archetype can be used as a criterion in authorizations:

```
<authorization>
  <action>...</action>
  <object>
    <archetypeRef oid="00000000-0000-0000-0000-000000000321"/>
  </object>
</authorization>
```

Multiple Archetypes

MidPoint 4.2 and later

This feature is available since midPoint 4.2

MidPoint archetypes are strongly inspired by how LDAP objectClasses can be defined and used. Therefore, midPoint is designed to support three kinds of archetype:

- abstract (not yet supported) - cannot be assigned directly to object, can be extended, can extend another abstract archetype
- structural - only one can be assigned to the object directly, can be extended, can extend other structural or abstract archetypes
- auxiliary (not yet supported)- can be assigned to the object with structural archetype assigned, object can have more then one auxiliary archetypes assigned

For now, only structural archetypes are implemented and supported. Since version 4.2 it is possible to define hierarchy between structural archetypes, so there can be one parent which is extended by its child. To define archetype inheritance, it is needed to point in the child archetype to its parent using *super ArchetypeRef* element, such as in the example bellow

Archetype inheritance

```
<archetype>
        ...
        <superArchetypeRef oid="00000000-0000-0000-0000-000000000521" type="ArchetypeType"/>
        ....
</archetype>
```

Archetype Inheritance

Using archetype inheritance, following practices apply:

- basic archetype attributes, such as name, displayName, ... - those defined in archetype which is assigned directly to the object are used.
- archetypePolicy - all archetype policies defined either in directly assigned archetype, or super archetypes are merged together.
- inducement / assignment - these are applied based on the standard midPoint algorithms. The important think to mention is, that the inheritance relation defined by superArchetypeRef is (4.2) translated to the inducement as well. In other words, example above is in midPoint 4.2 translated to inducement with target oid="00000000-0000-0000-0000-000000000521" while evaluating assignments/inducements.

Archetype policy - merging

There is quite complex algorithm for merging archetype policies across hierarchy. Following examples will show how the merging works. The example bellow shows archetype for basic task. It contains archetype policy defining the details about how the icon should look like, and two GUI virtual containers (sections) used on task details page - Advanced options and Operational attributes (state)

Basic task

```
{
  "@ns" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3",
  "archetype" : {
    "name" : "Basic task",
    "archetypePolicy" : {
      "display" : {
        "label" : "Task",
        "pluralLabel" : "Tasks",
        "icon" : {
          "cssClass" : "fa fa-tasks",
          "color" : "grey"
        }
      },
      "adminGuiConfiguration" : {
        "objectDetails" : {
          "type" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3#TaskType",
          "container" : [ {
            "display" : {
              "label" : "Advanced options"
            },
            "displayOrder" : 150,
            "item" : [ {
              "path" : "cleanupAfterCompletion"
            }, {
              "path" : "threadStopAction"
            }, {
              "path" : "binding"
            }, {
              "path" : "dependent"
            } ]
          }, {
            "display" : {
              "label" : "Operational attributes (state)"
            },
            "displayOrder" : 900,
            "item" : [ {
              "path" : "executionStatus"
            }, {
              "path" : "node"
            }, {
              "path" : "nodeAsObserved"
            }, {
              "path" : "resultStatus"
            }... ]
          } ]
        }
      }
    }
  }
}
```

The next archetype example extends the *Basic task* archetype above. It is a parent archetype for resource related tasks, containing additional information about icon color, attributes which have to be hidden/shown on details page and additional information to GUI virtual containers (sections) on details page.

Resource related task (extends Basic task)

```
{
  "@ns" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3",
  "archetype" : {
    "name" : "Resource related task",
     ....
    "archetypePolicy" : {
      "display" : {
        "icon" : {
          "color" : "green"
        }
      },
      "itemConstraint" : [ {
        "path" : "extension",
        "visibility" : "vacant"
      }, {
        "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:objectclass",
        "visibility" : "visible"
      }, {
        "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:kind",
        "visibility" : "visible"
      }.... ],
      "adminGuiConfiguration" : {
        "objectDetails" : {
          "type" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3#TaskType",
          "container" : [ {
            "identifier" : "resourceOptions",
            "display" : {
              "label" : "resourceObjects"
            },
            "item" : [ {
              "path" : "objectRef"
            }, {
              "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:objectclass"
            }, {
              "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:kind"
            } ]
          }, {
            "identifier" : "resourceOperationOptions",
            "display" : {
              "label" : "operationOptions"
            },
            "item" : [ {
              "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:dryRun"
            } ]
          } ]
        }
      }
    },
    "superArchetypeRef" : {
      "oid" : "00000000-0000-0000-0000-000000000511",
      "type" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3#ArchetypeType"
    }
  }
}
```

The last archetype example is extension of *Resource related task*. This archetype describe additional details for Reconciliation tasks.

Reconciliation task (extends Resource related task)

```
{
  "@ns" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3",
  "archetype" : {
    "oid" : "00000000-0000-0000-0000-000000000541",
    "name" : "Reconciliation task",
    ....
    "archetypePolicy" : {
      "display" : {
        "label" : "Reconciliation task",
        "pluralLabel" : "Reconciliation tasks",
        "icon" : {
          "cssClass" : "fa fa-exchange"
        }
      },
      "itemConstraint" : [ {
        "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:objectQuery",
        "visibility" : "visible"
      }, {
        "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:finishOperationsOnly",
        "visibility" : "visible"
      } ],
      "adminGuiConfiguration" : {
        "objectDetails" : {
          "type" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3#TaskType",
          "container" : [ {
            "identifier" : "resourceOptions",
            "display" : {
              "label" : "ReconciliationTask.resourceObjects"
            },
            "item" : {
              "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:objectQuery"
            }
          }, {
            "identifier" : "resourceOperationOptions",
            "display" : {
              "label" : "ReconciliationTask.reconciliationOptions"
            },
            "item" : {
              "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:finishOperationsOnly"
            }
          } ]
        }
      }
    },
    "superArchetypeRef" : {
      "oid" : "00000000-0000-0000-0000-000000000521",
      "type" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3#ArchetypeType"
    }
  }
}
```

Examples above show a hierarchy for task archetypes: *Basic task* archetype is extended by *Resource related task* which is extended by *Reconciliation task* archetype. After assigning *Reconciliation task* archetype to a task, the merged archetype policy then will be:

Reconciliation task archetype when merged

```
{
  "@ns" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3",
  "archetype" : {
    "name" : "Basic task",
    "archetypePolicy" : {
```

```
      "display" : {
        "label" : "Reconciliation task",
        "pluralLabel" : "Reconciliation tasks",
        "icon" : {
          "cssClass" : "fa fa-exchange",
          "color" : "green"
        }
      },
      "itemConstraint" : [ {
        "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:objectQuery",
        "visibility" : "visible"
      }, {
        "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:finishOperationsOnly",
        "visibility" : "visible"
      }, {
        "path" : "extension",
        "visibility" : "vacant"
      }, {
        "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:objectclass",
        "visibility" : "visible"
      }, {
        "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:kind",
        "visibility" : "visible"
      }.... ],
      "adminGuiConfiguration" : {
        "objectDetails" : {
          "type" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3#TaskType",
          "container" : [ {
            "identifier" : "resourceOptions",
            "display" : {
              "label" : "ReconciliationTask.resourceObjects"
            },
            "item" : [ {
              "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:objectQuery"
            }, {
              "path" : "objectRef"
            }, {
              "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:objectclass"
            }, {
              "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:kind"
            } ]
          }, {
            "identifier" : "resourceOperationOptions",
            "display" : {
              "label" : "ReconciliationTask.reconciliationOptions"
            },
            "item" : [ {
              "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:finishOperationsOnly"
            }, {
              "path" : "declare namespace mext='http://midpoint.evolveum.com/xml/ns/public/model/extension-3';
extension/mext:dryRun"
            } ]
          }, {
            "display" : {
              "label" : "Advanced options"
            },
            "displayOrder" : 150,
            "item" : [ {
              "path" : "cleanupAfterCompletion"
            }, {
              "path" : "threadStopAction"
            }, {
              "path" : "binding"
```

```
            }, {
              "path" : "dependent"
            } ]
          }, {
            "display" : {
              "label" : "Operational attributes (state)"
            },
            "displayOrder" : 900,
            "item" : [ {
              "path" : "executionStatus"
            }, {
              "path" : "node"
            }, {
              "path" : "nodeAsObserved"
            }, {
              "path" : "resultStatus"
            }... ]
          } ]
        }
      }
    },
    "superArchetypeRef" : {
      "oid" : "00000000-0000-0000-0000-000000000521",
      "type" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3#ArchetypeType"
    }
  }
}
```

Limitations

Following limitations for archetype functionality apply in midPoint 4.x:

- Only User, Role, Org and Service archetypes are supported in 4.0. Archetypes are designed to work with many object types, including tasks and resources. And in theory, archetypes may be applied to archetypes themselves, creating meta-archetypes. But all of that is not fully supported yet. Some of that extra functionality may work, but it is not tested properly. Therefore use it at your own risk only.
- Performance limitation: Do not create too many archetypes. They all need to be cached in RAM. Tens or even hundreds are perfectly fine. Thousands or more may be a problem.
- AssignmentRelation works only in archetypes. While theoretically assignmentRelation can be placed in any assignment/inducement, this is not yet supported. It must be a first-order inducement (inducement order must be 1). Assignment relation in metaroles or other mechanism that requires higher-order inducement or inducement chaining are not supported yet.
- assignment/inducement that contains assignmentRelation must be always active (non-conditional, no activation)
- AssignmentRelation in archetype assignment is not fully supported yet.
- AssignmentRelation must be (almost) fully specified to work well in midPoint 4.0. Only the archetype definition may be missing. Object type and relation must always be specified. Full support for wildcard assignmentRelations is planned for later midPoint versions.
- AssignmentRelation only applies to limit the assignments between objects. It does not support limitations of inducements yet. I.e. there is no support for *order constraints* in the assignment relation specification. That is planned for later midPoint versions.
- AssignmentRelation does not limit the assignments that can be created - yet. The default behaviour of assignments is *open* (see above). Assignment relation is used in midPoint 4.0 mostly to render special button for user convenience.
- Archetype assignments are not supposed to change during the lifetime of an object. Archetypes should be set once, ideally at the beginning of lifecycle (object created from GUI with an archetype, archetype set by inbound mapping, etc.). The archetype should be fixed and it should never change. That is expected behavior for midPoint 4.0. This behavior can can change in later midPoint versions. However, there still will be some limitations about archetype change. Change of archetype may mean change of object schema, therefore this will always be a sensitive operation.
- There is no user interface to conveniently manage archetype definitions yet.
- Archetypes of archetypes (meta-archetypes) are not supported yet.
- Archetype colors are not applied in the user interface consistently. E.g. the color of "summary panel" on user details page will be red, regardless of the archetype, as red is currently color associated with users. This is planned to be improved later.
- User type colors from 3.x are currently hardcoded in the system. Therefore users are displayed in the menu as red. The plan is to be able to configure this behavior in the future.

Please see Archetype Improvements (Planned Feature) for future plans regarding archetype functionality development.

See Also

- Archetypes
- Object Collections and Views
- Archetype Improvements (Planned Feature)