Creation of multiple object-related approval processes via policy rules HOWTO

Mechanisms described here are part of midPoint 3.7. However, they are considered **EXPERIMENTAL** and may (and probably will) change in future midPoint releases.

Imagine that you want to split approval of an object modification into more independent approval processes, each with its own approval schema. For example, you need to approve any change of user's costCenter attribute separately from other changes for that user.

Something like this:

Rule	Description
1	Each modification of a user record should be approved by his line manager.
2	Each modification of user's costCenter should be approved by his line manager and accounting department representative.

So, the behavior for typical scenarios will be:

Scenario	Description	Behavior
1	User's administrativeStatus is changed.	One process is created, asking line manager for his approval.
2	User's costCenter is changed (e.g. new value is added).	One process is created, asking line manager and accounting department representative for their approvals.
3	User's costCenter and administrativeStatus are changed.	Two processes are created: 1. approval of administrativeStatus change: line manager is asked 2. approval of costCenter change (add/delete/replace of one or more values): line manager and accounting department representative are asked

The implementation is done using the following policy rules:

Implementation of rule 1 (general modification)

```
<globalPolicyRule>
    <name>user-modification</name>
    <policyConstraints>
        <modification>
            <operation>modify</operation>
        </modification>
    </policyConstraints>
    <policyActions>
        <approval>
            <name>line-manager-approval</name>
            <compositionStrategy>
                <order>10</order>
            </compositionStrategy>
                        <approverExpression> ... </approverExpression>                <!-- selection of line
manager is here -->
        </approval>
    </policyActions>
    <focusSelector>
        <type>UserType</type>
    </focusSelector>
</globalPolicyRule>
```

Implementation of rule 2 (costCenter modification)

```
<globalPolicyRule>
   <name>modification-of-cost-center</name>
   <policyConstraints>
      <modification>
         <item>costCenter</item>
      </modification>
   </policyConstraints>
   <policyActions>
      <approval>
         <processSpecification>
            <deltaFrom>
               <item>costCenter</item>
            </deltaFrom>
            <includeAction>line-manager-approval</includeAction>
         </processSpecification>
         <compositionStrategy>
            <order>20</order>
         </compositionStrategy>
         <approverRef oid="..." type="UserType"/>  <!-- accounting department representative -->
      </approval>
   </policyActions>
   <focusSelector>
      <type>UserType</type>
   </focusSelector>
</globalPolicyRule>
```

Note that the approval action in the second policy rule refers to the first rule (includeAction: line-manager-approval). It is done to avoid duplication of code for line manager approval instruction. The includeAction is currently limited to those actions that are triggered together with the referring one. This might change in the future - the whole multi-process approvals is considered an experimental feature in midPoint 3.7.

"Per value" approvals

Now let's assume that we want to approve each cost center being added or removed individually, e.g. by a manager of given cost center. (Another, perhaps more realistic, example is approving inducements added or removed from a role by their owners or approvers.)

The rules would look like this:

Rule	Description
1	Each modification of a user record should be approved by his line manager.
2	Each modification of user's costCenter should be approved by his line manager and the manager of the cost center being added or removed.

So, the behavior for typical scenarios will be:

Scenario	Description	Behavior
1	User's administrativeStatus is changed.	One process is created, asking line manager for his approval.
2	User's costCenter is changed: single new value is added.	One process is created, asking line manager and cost center manager for their approvals.
3	User's costCenter is changed: one value (CC1) is removed and two new values (CC2, CC3) are added.	Three processes are created: 1. removal of CC1: asking line manager and CC1 manager for their approvals, 2. addition of CC2: asking line manager and CC2 manager for their approvals, 3. addition of CC3: asking line manager and CC3 manager for their approvals.

3	User's costCenter and administrativeStatus are changed. Value of CC3 is removed and value of CC4 is added.	Three processes are created: 1. approval of administrativeStatus change: line manager is asked, 2. approval of removal of CC3: line manager and CC3 manager are asked, 3. approval of addition of CC4: line manager and CC4 manager are asked,

As for the implementation, rule 1 is the same as in previous example. Rule 2 is slightly modified:

Implementation of rule 2 (costCenter modification)

```
<globalPolicyRule>
   <name>modification-of-cost-center</name>
   <policyConstraints>
      <modification>
         <item>costCenter</item>
      </modification>
   </policyConstraints>
   <policyActions>
      <approval>
         <processSpecification>
            <deltaFrom>
               <itemValue>costCenter</itemValue>
            </deltaFrom>
            <includeAction>line-manager-approval</includeAction>
         </processSpecification>
         <compositionStrategy>
            <order>20</order>
         </compositionStrategy>
         <approverExpression> ... </approverExpression>  <!-- deriving manager for the CC being added or 
removed -->
      </approval>
   </policyActions>
   <focusSelector>
      <type>UserType</type>
   </focusSelector>
</globalPolicyRule>
```

<item>costCenter</item> was changed to <itemValue>costCenter</itemValue> meaning that we are no more interested in change of costCenter as such but in each value of costCenter that is being added or deleted. Also, fixed approverRef is replaced by approverExpression. Details of the expression are left as an exercise for the reader. 🙂