Notifications

Introduction

This mechanism is used to notify users about relevant changes in midPoint and/or connected resources. For example, a user (or user's boss, or the person who requested the operation, the security manager, etc) may be notified when one of user's accounts is created, modified, or removed. Or, when the midPoint user record is created, when the password is changed, or when he has a new work item to process. There are many such situations imaginable.

Currently there are the following basic kinds of notifications:

1. **User notifications.** These are related to midPoint user record, e.g. its creation, modification or removal.
2. **Resource object notifications.** These are related to objects on resources, e.g. creation, modification, or removal of accounts, groups, etc.
3. **Workflow notifications.** These are generated e.g. when a work item is created or completed, or when a workflow process instance is started or finished.
4. **Access certification notifications.** They are sent e.g. when a campaign is started, closed or about-to-be-closed, or when a response is requested from a reviewer.
5. **Task notifications.** These are created when a task is started or finished. They are useful e.g. to notify in cases when a recurring task (like a reconciliation or live sync) ends with a failure.
6. **Custom notifications.** These can be used for any other purposes.

Some simple examples

Configuration of notifications is currently done within SystemConfiguration object. Some examples are shown below:

```
<notificationConfiguration>
    <handler>

        <!-- this event handler sends accounts passwords (when created or changed) via
mail to the account owner email address, if known -->
        <accountPasswordNotifier>
            <recipientExpression>
                <script><code>requestee.getEmailAddress()</code></script>
            </recipientExpression>
            <transport>mail</transport>
        </accountPasswordNotifier>

        <!-- this event handler sends accounts passwords (when created or changed) via
mail to the account owner email address stored in schema extension attribute
otherMailbox, if known -->
        <accountPasswordNotifier>
            <recipientExpression>
                <script><code>basic.getExtensionPropertyValue(requestee,
'http://evolveum.com/my', 'otherMailbox')</code></script>
            </recipientExpression>
            <transport>mail</transport>
        </accountPasswordNotifier>

        <!-- this event handler sends accounts passwords (when created or changed) via
SMS to the account owner telephone number, if known -->
        <accountPasswordNotifier>
            <recipientExpression>
                <script><code>requestee.getTelephoneNumber()</code></script>
            </recipientExpression>
            <transport>sms:default</transport>
        </accountPasswordNotifier>
```

```
        <!-- this event handler sends *user* passwords (when created or changed) via
SMS to the user's telephone number, if known -->
        <userPasswordNotifier>
            <recipientExpression>
                <script><code>requestee.getTelephoneNumber()</code></script>
            </recipientExpression>
            <transport>sms</transport>
        </userPasswordNotifier>

     <!-- this event handler sends notifications about successful account creation to
the account owner mail, if known -->
        <simpleResourceObjectNotifier>
            <status>success</status>  <!-- only successful operations! -->
            <transport>mail</transport>
        </simpleResourceObjectNotifier>

        <simpleUserNotifier>  <!-- sends notification about user operations to the
user's mail address -->
            <transport>mail</transport>
        </simpleUserNotifier>

        <simpleTaskNotifier>
            <transport>mail</transport>
        </simpleTaskNotifier>

    </handler>

    <!-- configurations suitable for testing - they redirect all notifications to log
files; some more real configurations are show below -->
    <mail>
        <redirectToFile>mail-notifications.log</redirectToFile>
    </mail>
    <sms>
        <redirectToFile>sms-notifications.log</redirectToFile>
```

```
    </sms>
</notificationConfiguration>
```

General Notifier Example

```
<notificationConfiguration>
. . .
    <handler>
     <generalNotifier>
         <name>Notify system administrator for organization change - general</name>
            <expressionFilter>      <!-- Filter only changes of OrgType objects -->
                <script>
                    <code>
                    import com.evolveum.midpoint.notifications.api.events.ModelEvent
                    import
com.evolveum.midpoint.xml.ns._public.common.common_3.OrgType

                   (event instanceof ModelEvent &amp;&amp; event.getFocusContext() !=
null &amp;&amp;
OrgType.class.isAssignableFrom(event.getFocusContext().getObjectTypeClass()))
                    </code>
                </script>
            </expressionFilter>
            <recipientExpression>    <!-- Send notifications to iam@localhost -->
                <value>iam@localhost</value>
            </recipientExpression>
            <subjectExpression>      <!-- Subject expression is defined here -->
                <script>
                    <code>
                    tmpObject = 'Organization - GeneralNotifier - '
                    if (event.isSuccess())
                        tmpText = "[IDM] SUCCESS: " + tmpObject +
event?.getChangeType() + " operation succeeded"
                    else if (event.isFailure())
                        tmpText = "[IDM] ERROR: " + tmpObject + event?.getChangeType()
+ " operation failed"
                    else  tmpText = "[IDM] IN PROGRESS: " + tmpObject +
event?.getChangeType() + " operation in progress"
                    return tmpText
                </code>
               </script>
            </subjectExpression>
            <bodyExpression>         <!-- Body expression is defined here -->
                <script>
                   <code>            <!-- Some imports needed to have all necessary
objects and variables -->
                    import
com.evolveum.midpoint.notifications.impl.notifiers.GeneralNotifier;
                    import com.evolveum.midpoint.notifications.api.events.ModelEvent;
                    import com.evolveum.midpoint.prism.delta.ObjectDelta;
                    delta = ObjectDelta.summarize(((ModelEvent)
event).getFocusDeltas());
                    hiddenPaths = GeneralNotifier.getAuxiliaryPaths()
                    body = ''
                    attemptedTo = event.isSuccess() ? "" : "(attempted to be) ";
```

```
                    if (delta.isAdd()) {
                        body = "The object was " + attemptedTo + "created with the
following data:\n";
                        body += textFormatter.formatObject(delta.getObjectToAdd(),
hiddenPaths, false);
                    } else if (delta.isModify()) {
                        body = "The object was " + attemptedTo + "modified. Modified
attributes are:\n";
                        body += textFormatter.formatObjectModificationDelta(delta,
hiddenPaths, false);
                    } else if (delta.isDelete()) {
                        body = "The object was " + attemptedTo + "removed.\n\n";
                    }
                    </code>
                </script>
            </bodyExpression>
            <transport>mail</transport>
        </generalNotifier>
    </handler>
    <!-- configurations suitable for testing - they redirect all notifications to log
files; some more real configurations are show below -->
    <mail>
         <redirectToFile>mail-notifications.log</redirectToFile>
    </mail>
    <sms>
```

```
        <redirectToFile>sms-notifications.log</redirectToFile>
    </sms>
</notificationConfiguration>
```

A bit of theory related to event handling

When something potentially of interest occurs, an **event** is generated. This event is then processed by one or more **event handlers**, as prescribed in <notificationConfiguration> section.

There are the following kinds of event handlers.

Notifiers

A notifier is a module that transforms an event to zero, one or more notifications. Examples are:

Notifier	Type	Description
simpleUserNotifier	User notification	Generates notifications about user records.
simpleResourceObjectNotifier	Resource object notification	Generates notifications about resource objects (e.g. accounts).
userPasswordNotifier	User notification	Generates notifications about user passwords.
accountPasswordNotifier	Account notification	Generates notifications about account passwords.
simpleWorkflowNotifier	Workflow notification	Generates notifications about start/completion of work items (i.e. user tasks) and about start/completion of workflow process instances.
simpleCampaignNotifier, simpleCampaignStageNotifier	Certification notification	Generates notifications about certification campaigns.
simpleTaskNotifier	Task notification	Generates notifications about tasks.
generalNotifier		This is a general purpose notifier that is driven by expressions, which transform an event into a notification.

Filters

A filter is a component that does not generate notifications, but passes through (or filters out) defined subsets of notifications. There are currently the following kinds of filters:

Filter	Description
category	passes one or more defined categories of events: resource object-related events ("resourceObjectEvent"), user-related events ("modelEvent"), work item-related events ("workItemEvent"), workflow-related events ("workflowProcessEvent"), or any workflow-related events ("workflowEvent" - currently this means "either workItemEvent or workflowProcessEvent")
status	passes or filters events based on their status: success, alsoSuccess, failure, onlyFailure, inProgress (with a bit different sets of acceptable values and their semantics for account/user/workflow events)
operation	passes or filters events based on the kind of operation that was executed or attempted to: add, modify, or delete
expression	a generic filter that evaluates an expression and passes/filters event based on the result
objectKind	passes or filters events based on the kind of resource object they deal with (account, entitlement, generic)
objectIntent	passes or filters events based on the intent of resource object they deal with

A filter can be specified as part of a notifier - in that case it defines which classes of events get processed by that notifier. Or a filter can be a standalone part of a handler chain (see below).

Special kinds of handlers

There are two special kinds of handlers:

Name	Description
Handler chain (chained)	Contains sequentially ordered event handlers (notifiers, filters, forks, other chains). Processing within a chain continues until an event is filtered out. (Please note that filtering is done only by filters; all notifiers simply pass all events along.)
Fork (forked)	Splits processing of an event into more handlers "in parallel". (We expect this kind of handler will be used only occassionally, however, we have implemented in for completeness.)

Mail attachments

We can define attachments for mail. We have two choices, adding attributes to the xml or writing of script.

> Supported since midPoint 4.0.

Example of attachments

```
<notificationConfiguration>
. . .
    <handler>
        <generalNotifier>
        . . .
           <attachment>
             <contentType>image/png</contentType>
                <contentFromFile>/home/user/example.png</contentFromFile>
           </attachment>
           <attachment>
                <contentType>text/plain</contentType>
                <content>Hello world!</content>
                <fileName>hello_world.txt</fileName>
           </attachment>
           <attachment>
                <contentType>image/jpeg</contentType>
                <content xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xsd:base64Binary">iVBORw0KGgoAAAANSUhEUgAAATUAAABLCAMAAAAmnuAzAAAAaVBMVEUAAA
D///////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////8G612AAAAInRSTlMAwIBA8NBwoOBgELAgkO
gw+1Cc3B/3PK9Xh6nWyvO5Uy1z3+GvZwAAB4RJREFUeNrsmO2W2iAQhl8+EwIkTWJ0rbqn5f4vslVTOwimlWN7
utbn52wCzMMwWcSLf4s1XtxPixf3c8CL+5nw4nVC/w6f8eJ+GF68rMUYtgSKUcjDCAYfCGcq4zDDwxIoZkSeQO
D4OBhpuZXmMdbW/4k1Iz03XSfNQ6ypmydx9VTWasGYl0LUD7GG3Ygscv1M1joJVonaQHYPsQY15MNvz2RNKLA6
VIASj7GGd40IzUbZyk/6mawhNIz30jXhYd/QkeHCtp3URgPY+aeyZivGYW1lH2YNo8KJFZOXg6meq9YaZXvXW9
U8zhrGzanOPtGiey5raLgKijdIrSlxDX6TjVpBhS2NPJk1SmyNoRglJ7nCjOs5r/d/zpo40uBRuJ5/JxqwEceQ
6UqtOUZxmOmv7pOr8J1WTa1sD5+mQxDje1C+/7U1zQgeP/A0TPPpqzqckdaA0jFCbpz5cRNPZ1SYsQJnhA0ztS
mstToQDGYqGu2Asd3VKwCuqjp+nM7vzwOu18vW+kDJPqZwwdSBUmtagbmBWFranObbqUDw5zqJJu+KrPlAqDCj
4qyU2gwbuYJpD1W0LB9s55asxbsicgZmNXOKMaxBuTUJI0NEBXRXIdmVWHPR5uJME2e1lRg0hnZoj8+EGNUvWv
O5TqdpcBZDU6RJ/bY1nWQswjV9l0yimgJrUIHgcKKPYutpBaaBcRpCBixa60JmBZbELM6YkEU1v2tNJBmn21DL
kMAXraWk+27StyzGLbBl6CuvcmmhWbKGOpOtTKfswg1sgbW7kE2BNUcjVboYs5YA+JvbWSNDhmovXd5auisimV
LiRFOHW4g/bC2YAmtgaWOLdmJcAxje9uE23Ny25tKQSffJ0ykZow7VfdZEuBdbYs0kjU3QrFYKANTnls7D/cgV
yfPLbWtQSWOrFquPX1do94etyRJrTVKtdMn9oAHg8/6nXN8AWwmIi493vWBNJ1OqpLj90lse5dak5zb5JHgWB1
yBNdBhq6tAjRFHdp8vK2xPSUlSNDWQt5buipgDsRFIqjEJFVijX+A+RGhc/9cmSqyZ68YmaVYtjkyDnqNf5QhA
eyrcuAVrsAv3BXedqUf6liu25tKbjk2f6kusQcbzONpTNMORT2TSyQNzuJHERmot3RUWr0klq6z4DCM+Sq2xjE
iR3iR5kbUqbmyGZjXo2JoDViPH4OPxWd5aekTjVHWSfA5das1mVuXinMut9XFjq+h6WWxN4ci0ZSy+kNU3raXf
zPQ2JcMivMxa/gRkxPCy39fqqLEpujEMR4RktO28TbsNPd2Su/WX0x+y1vooKDK18CGteVrBTZQVA6Df+LSepe
```

```
73XOvtLhwY01pvh5PidjPI3bCR7zJrDSTMoEk7+NDWvrVzrktugmAYftEP8BTEs+tme+D+L7LRtEqCsttOu0ln
eGbyQyCR70GOmbE0G5V1W2KkX5su6ZRRZuHcS8UGMWgz1KcvnJpuTtXTlFxkipo7wTpd1Log/NfW7C4aCXtf+3
Ncy+77VP1FtXbGtGTktROsE0m5mZX/uTWrGDvZUdmzgX1AIF5GLaxVhWK4QC9OsE6rVE6c3tnAv/JIHmqtsHvl
zeqvHjDT3W94GEbdbo2t6sVt71hzBk5pTzzuI5Mwl+Jo0QXzUGvglrabqEaOGV1HdM0tMNM2QCMhrmlxSS9LKT
jWPKdnJ0eqr5LxzlerB1sTh1F1mLlc5Pbh6qkB2vPrOtj93JweWQM3LgX2F4z7xO7BeMY/wRqxe05uF3VOaFS/
3DdfN7xJDIB/tVRTmZ7nUow51jytwrFBZiNJs61alYx3pwxKgYqbT7DmwrCS70a1Dmw6VWYlrzL5tRTJlsCVAN
D0jjVPqwhsyLtbsyiKToyR/SuJcXi4tfQoqra79tOzOWaq5mdPNHCsebpo5kj11zF/QmvZ4bDTjAAIfZMLZlz0
KCaJfJlBPdZS/7lz9P5Ja/qE1pAfRdUrADXwXQLGhScVy0BA3cCx5mmV9GASP2y9jJ7QWnUYVSOAVwBSxsblxP
U3QALU+6wh9x/WZ9xnYyF6Qmugo38J+w5YgjzNkTGrnKTLh71SDw4+wLHm6aISH9UWAYf//lH+aGvyMKqBQ7dL
5Iqn1zp017sV+SQzKNaeaoLfWkbOk+GvqzsAlnQvrYwebS02h2fpVGssFDJJs9npSHM9L5dLuSRXuvVYc1slwQ
6FJOPAUxxpS0pUVo0fYg3JYVSt1sPLwJqIMdYQKeK8U2+K3njOLuRMmxHvWaucfYdLlkq7FkymxW2+pZ6i7MZR
9PesFbGPEjblYQ5GkwyiFlPfiqGORDRxnU/12M5ZQohIvw14l9giwzHeMoXI+dXnkp3FKwWwpqxJ7p1/JUUbMf
4RY8dh0yfYEFqARgQcWurGG1FYYUsO1Qi4DKbpN4kKV1qhOBYSiYBLy0wj8BP6laTXpLfQSfeplR56zHCgvVzp
Gis95QjssKhSxhBp6gzxETdwBLwQdogGBHwo7DGFt6N4abBLmEe9EAIfIQz84aVYHyQ8a0+ARuC3aRMEwkuxPo
X2jMBv8zVsAv6AtkUgsPEDb0ZOzzalzlUAAAAASUVORK5CYII=</content>
                <fileName>evolveum_logo.jpg</fileName>
           </attachment>
           <attachmentExpression>
                <script xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="c:ScriptExpressionEvaluatorType">
                    <code>
                       import
com.evolveum.midpoint.xml.ns._public.common.common_3.NotificationMessageAttachmentType
;

                       NotificationMessageAttachmentType attachment = new
NotificationMessageAttachmentType();
                       attachment.setContentType("text/html");
                       attachment.setContent("&lt;!DOCTYPE
html&gt;&lt;html&gt;&lt;body&gt;Hello World!&lt;/body&gt;&lt;/html&gt;");
                       attachment.setFileName("hello_world.html");
                       return attachment;
                    </code>
                </script>
            </attachmentExpression>
            <transport>mail</transport>
        </generalNotifier>
    </handler>
</notificationConfiguration>
```

Attributes

If we use attachment, then we have two choices, insert the content to the xml or use file.

Name	Required	Description
contentType	true	Content type for the attachment, e.g. image/png or text/html.
content	choice	Content of the attachment.
contentFromFile	choice	Path of the file that provides the content for this attachment.
fileName	false	File name for the attachment. E.g. in the case of the mail transport it should be put into Content-Disposition header field. If omitted and if contentFromFile is used, it is derived from the name of that file. If it omitted with used content, it's value is 'attachment'.

Executing the handlers

When an event is created, all the handlers defined at the level of <notificationConfiguration> are executed.

Some more advanced examples:

TODO

For instructions how to configure notifiers, filters, and transports please see administrator's guide.

To send HTML emails, set <contentType>text/html; charset=UTF-8</contentType> and put to <bodyExpression> escaped HTML in single quote.

Custom notifications

These are briefly described in the Sending custom notifications HOWTO.

See Also

- Configuring notifications
- Using Velocity Templates for Notifications