Manual Resource Configuration

MidPoint 3.6 and later

This feature is partially implemented in midPoint 3.6. The completion work needs your support. See Manual Resource and ITSM Integration.

Introduction

See Manual Resource and ITSM Integration for an introduction to the topic.

Pure Manual Resources

Pure manual resources are configured as (almost) ordinary single-connector resources. Internal manual connector implementation or an ITSM Plugin should be used in place of an ordinary identity connector. Following example shows the use of internal manual connector implementation:

```
 <resource>
    <name>Manual Resource</name>
    <connectorRef type="ConnectorType">
        <filter>
            <q:equal>
                <q:path>c:connectorType</q:path>
                <q:value>ManualConnector</q:value>
            </q:equal>
        </filter>
    </connectorRef>

    <connectorConfiguration>
        <conf:defaultAssignee>administrator</conf:defaultAssignee> <!-- EXPERIMENTAL !!! -->
    </connectorConfiguration>

    <!-- Static schema definition -->
    <schema>
        <definition>
             <xsd:schema ...>
                ...
                <xsd:complexType name="AccountObjectClass">
                   ...
                </xsd:complexType>
             </xsd:schema>
          </definition>
    </schema>

</resource>
```

Pure manual resource cannot automatically discover the schema. Therefore it needs to have a static schema definition. Apart from that almost all other midPoint features can be used in an almost normal way. E.g. there may be quite ordinary schemaHandling section. Outbound mappings in this section will influence the values that will appear in tickets (cases) that the system administrators receive.

Semi-Manual Resources

Semi-manual resources are built on the combination of a manual connector and an ordinary connector. It is using the multi-connector resource mechanism. The primary connector is the manual connector. The ordinary connector is configured as an additional connector. However, all the capabilities are disables - except for *read* and *schema* capabilities.

The following example shows combination of midPoint built-in manual connector and a CSV connector:

```
 <resource>
    <name>Semi-Manual CSV Resource</name>

    <connectorRef type="ConnectorType">
        <filter>
            <q:equal>
                <q:path>c:connectorType</q:path>
                <q:value>ManualConnector</q:value>
            </q:equal>
        </filter>
    </connectorRef>

    <connectorConfiguration>
        <conf:defaultAssignee>administrator</conf:defaultAssignee> <!-- EXPERIMENTAL !!! -->
    </connectorConfiguration>

    <additionalConnector>
        <name>csv</name>
        <connectorRef type="ConnectorType">
            <filter>
                <q:equal>
                    <q:path>c:connectorType</q:path>
                    <q:value>com.evolveum.polygon.connector.csv.CsvConnector</q:value>
                </q:equal>
            </filter>
        </connectorRef>
        <connectorConfiguration>
            <icfc:configurationProperties>
                <csvconf:filePath>/var/opt/midpoint/resources/semi-manual.csv</csvconf:filePath>
                ...
            </icfc:configurationProperties>
        </connectorConfiguration>
        <capabilities>
            <configured>
                <cap:liveSync>
                    <cap:enabled>false</cap:enabled>
                </cap:liveSync>
                <cap:create>
                    <cap:enabled>false</cap:enabled>
                </cap:create>
                <cap:update>
                    <cap:enabled>false</cap:enabled>
                </cap:update>
                <cap:delete>
                    <cap:enabled>false</cap:enabled>
                </cap:delete>
                <cap:script>
                    <cap:enabled>false</cap:enabled>
                </cap:script>
                <cap:activation>
                    <cap:status>
                        <cap:attribute>ri:disabled</cap:attribute>
                        <cap:enableValue>false</cap:enableValue>
                        <cap:disableValue>true</cap:disableValue>
                    </cap:status>
                </cap:activation>
            </configured>
        </capabilities>
    </additionalConnector>

    <!-- Schema definition comes from the CSV file -->

    <consistency>
        <pendingOperationGracePeriod>PT15M</pendingOperationGracePeriod>
    </consistency>

</resource>
```

There is no need to specify static schema in this case. The CSV connector can be used to automatically generate the schema from the CSV file header. Also please note how the simulated activation capability is configured in the CSV connector capabilities.

Grace Period

The basic principle of the manual and semi-manual connector operation is that midPoint remembers the deltas of operations in progress. The deltas are kept in the shadow objects. When the operation is finished (case/ticket is closed) then the operation is applied to the cached values and the delta is not needed any more. Therefore such deltas are deleted by default. However there are two cases when such deltas may be needed:

- Inspection of operation status and outcome. There might be failure or a warning. If the deltas of finished operations are immediately deleted then there is no convenient way how to get the error or warning messages.
- Latency of semi-manual resources. E.g. if the CSV file used for semi-manual resource is updated on midnight, then none of the operations that were completed during the day are reflected to the CSV file yet. Therefore midPoint needs to keep the deltas of finished operations to correctly present the expected state of the account.

The time interval for which midPoint keeps deltas of finished operations is called *grace period*. It can be configured in the `consistency` part of the resource definition (see above).

Grace Period Implementation

This is how the grace period is really implemented: Provisioning component always applies any deltas, even if they are in grace period. And the connector always creates a case for the changes. But when midPoint projector component reads the account, it indicates that it wants "future point in time" read. In that case provisioning will take the value from CSV, apply all the pending deltas and return that value. This is the value that it should look like when the pending changes are applied. Therefore the reconciliation part of the projector will not compute any reconciliation changes. But when the grace period expires, provisioning component stops to pretend that they were applied. And if the changes are not already in the CSV then the reconciliation detects that, new modifications are executed and new case is created.

ITSM Plugin

Both manual and semi-manual resources are often used with ITSM integration plugins. See ITSM Plugin page for more details.

Refresh Task

Manual resources depend on information from the cases/tickets to detect when an operation is completed. Current implementation assumes that midPoint is always the active party (client): initiating operations and polling for status changes. Therefore to make the manual connectors work there is a need for a task, that will scan the status of all pending operations. Shadow refresh task will do that:

```
<task>
    <name>Shadow refresh</name>
    ...
    <handlerUri>http://midpoint.evolveum.com/xml/ns/public/model/shadowRefresh/handler-3</handlerUri>
    <recurrence>recurring</recurrence>
    <schedule>
        <interval>10</interval>
    </schedule>
</task>
```

Pending Deltas and Cached Data

The general principle is that all midPoint connectors must be able to read the data. But in the (pure) manual case there is no way to read the data from the resource. Therefore in that case midPoint relies on attribute caching. Which means that the pending delta corresponding to the closed ticket is applied to the data cached in the shadow. That is how the resource attributes are supposed to look after the ticket is closed. And as the ticket was closed we assume that the operation was executed successfully.

The semi-manual (manual+CSV) case is similar. However, in this case we have a way how to read the data from resource (although there may be a delay). Therefore in this case the delta from the closed ticket is NOT applied to the CSV data. But we still need to address the delay, e.g. the CSV file may be updated few days after the change was made on resource. Therefore the deltas for closed tickets are still kept in the shadow as pending deltas. While the deltas are in the shadow midPoint will pretend that the changes were applied. MidPoint will take those deltas in consideration during reconciliation process which means that midPoint will not try to "fix" value inconsistencies. When those deltas expire (after "grace period") then midPoint stops pretending that the operation was done. The values should already be in the CSV by that time and everything should be OK. But if the values do not appear in the CSV file then the reconciliation process will notice the inconsistency. The result is that reconciliation will try to fix the problem. Which means a new ticket will be created.

See Also

- Manual Resource and ITSM Integration
- Multi-Connector Resource
- ITSM Plugin