Unique midPoint User Name

Similar to an unique account username, unique midPoint user names may be configured. For example, you may need to generate your user names based on Given Name and Family Name attributes until the user name is unique:

- `john.smith` - found existing user, trying again
- `john.smith1` - found existing user, trying again
- `john.smith2` - no existing user, use this for the new user

The requirement also can be based on different methods than just sequential number increments, for example random number:

- `john.smith` - found existing user, trying again
- `john.smith109` - no existing user, use this for the new user

MidPoint has a built-in functionality to support this method. The feature is called *iteration* and it can be enabled in Object Template. The following steps are required to configure this feature:

- Enable the iteration by setting the maximum number of iteration attempts. This is done by setting the `iteration` property in the Object Template.
- Use of the `tokenExpression` in the Object Template to specify algorithm for the iterations
- Use one of the iteration variables (`iteration` or `iterationToken`) in (user name) mapping.

When enabled the midPoint will do the following when a new user is created or existing user is renamed:

1. MidPoint sets iteration variables to initial values (see below)
2. MidPoint evaluates the mappings
3. MidPoint checks if mapping results for user name are unique
    a. If yes: we have the final values, midPoint continues with other mappings/provisioning
    b. If no: iteration variables are changed to the next iteration and the process is retried until the maximum number of iterations is reached

Iteration Variables

There are two iteration variables that can be used in mappings:

- Variable `iteration`: Numeric variable contains the number of current iteration. It starts with `0` and increments on every iteration.
- Variable `iterationToken`: String variable that contains the portion of the identifier that is changed in each iteration. It can be derived from the iteration number using a special expression. A default value is supplied if no expression is used. See the example below.

Iteration number	The value of `iteration` variable	The default value of `iterationToken` variable
0	0	"" (empty string)
1	1	"1"
2	2	"2"

The iteration variables can be used in Object Template mappings. In this example, the mapping will be applied only for employees (employeeType == "EMPLOYEE"):

```
<objectTemplate>
. . .
    <mapping>
        <name>My Object Template: Name for employees</name>
        <source>
            <path>givenName</path>
        </source>
        <source>
            <path>familyName</path>
        </source>
        <source>
            <path>employeeType</path>
        </source>
        <expression>
            <script>
                <code>
                tmpGivenName = basic.norm(basic.stringify(givenName))?.tr(' ', '.')
                tmpFamilyName = basic.norm(basic.stringify(familyName))?.tr(' ', '.')
                return tmpGivenName + '.' + tmpFamilyName + iterationToken
                </code>
            </script>
        </expression>
        <target>
            <path>name</path> <!-- midPoint user name -->
        </target>
        <condition>
             <script>
                <code>givenName != null &amp;&amp; familyName != null &amp;&amp; employeeType == 'EMPLOYEE'<
/code>
            </script>
        </condition>
    </mapping>
. . .
</objectTemplate>
```

This example is the most basic use of `iterationToken` variable. The effect of this mapping is to suffix the username. Default value of `iterationToken` variable is used in this example. This token has an empty value on the first iteration. Therefore if the `name` of the user object is unique then an account without any suffix is created (e.g. `jack`). However if the name is not unique then a suffix is appended to the username until an unique value is found (e.g. `jack3`).

Iteration Token Expression (Sequential, Zero Padded)

The expression that transforms numeric value of variable `iteration` to the string value of variable `iterationToken` is configurable. In the following example, iterator is increasing sequentially and zero-padded:

```
<objectTemplate>
. . .
    <iteration>
        <maxIterations>999</maxIterations>
        <tokenExpression>
            <script>
                <code>
                if (iteration == 0) {
                    return "";
                } else {
                    return sprintf("%03d", iteration);
                }
                </code>
            </script>
        </tokenExpression>
    </iteration>
. . .
</objectTemplate>
```

This expression will result in the following username sequence:

- `john.smith`
- `john.smith001`
- `john.smith002`
- ...

Iteration Token Expression (Random, Zero Padded)

In the following example, iterator is random (1-999 to have at most three digits) and zero-padded:

```
<objectTemplate>
. . .
    <iteration>
        <maxIterations>999</maxIterations>
        <tokenExpression>
            <script>
                <code>
                if (iteration == 0) {
                return "";
                } else {
                    rnd = new Random().nextInt(999) + 1
                    return "." + sprintf("%03d", rnd);
                }
                </code>
            </script>
        </tokenExpression>
    </iteration>
. . .
</objectTemplate>
```

This expression might result in the following username sequence:

- `john.smith`
- `john.smith.56`
- `john.smith.381`
- ...